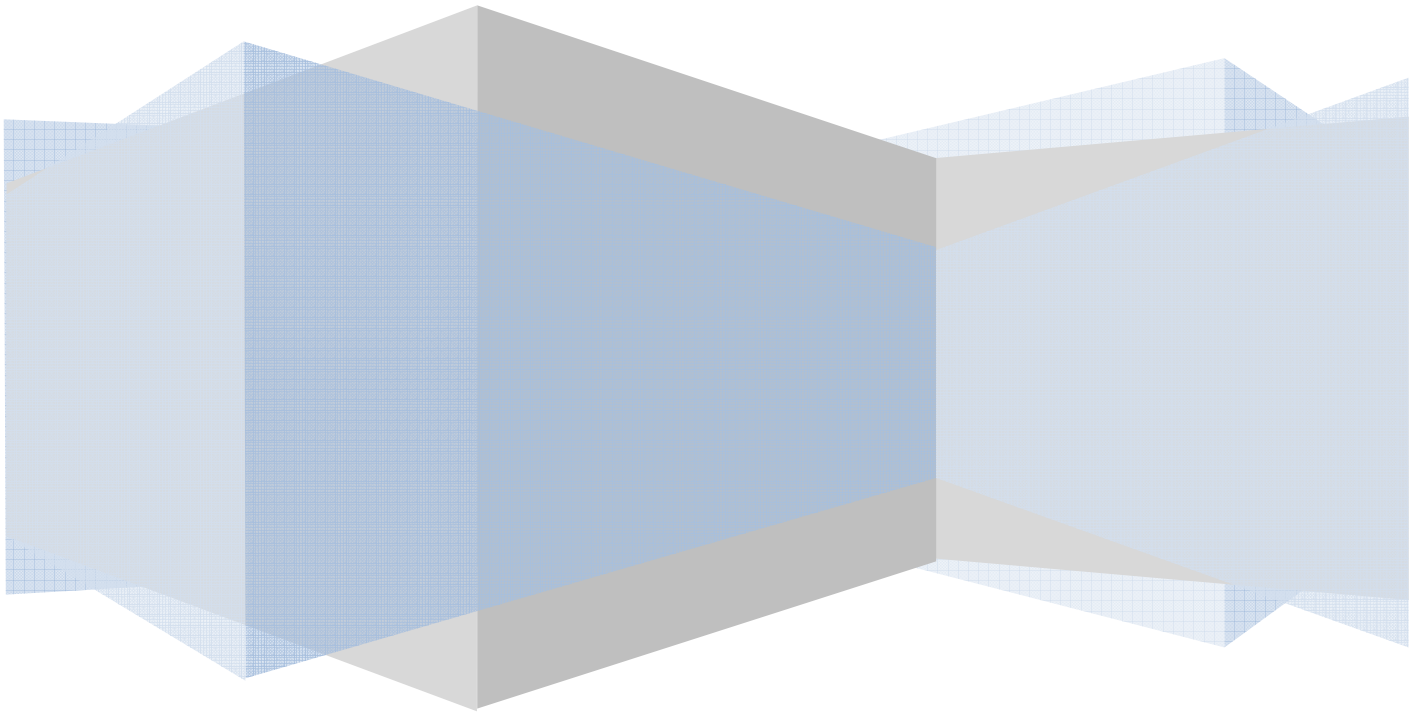
Жовтоводська гуманітарна гімназія

«Основи алгоритмізації та програмування мовою Паскаль»

Розробки уроків за програмою курсу
9 клас

Укладач: І.М.Бахтин



Передмова

Конспекти уроків розроблені згідно програми курсу «Основи алгоритмізації та програмування» 9 клас. Містять необхідний об'єм теоретичного матеріалу, докладно розібрані приклади та завдання для самостійної роботи.

Використані міжпредметні зв'язки (задачі з математичним, фізичним, економічним змістом), наводяться приклади уривків літературних творів, певним чином пов'язаних з навчальними темами, розглядається велика кількість цікавих, із «розважальним» змістом задач. Передбачена наявність самостійних робіт, інформаційних диктантів, практичних робіт, тематичних атестацій. Система методичних рекомендацій дає можливість оптимізувати навчальний процес, вносити коригування в залежності від специфіки класу.

Програма курсу «Основи алгоритмізації та програмування. 9 клас»

1. Пояснювальна записка

Сучасний світ ставить до школярів зовсім інші, ніж у минулому столітті, вимоги. Сьогодні потрібні не "гвинтики" для єдиного суспільного механізму, а творчі особистості, здатні самостійно приймати рішення. Ознакою часу, в якому ми живемо, є лавинне нагромадження інформації і бурхливий розвиток мікроелектронної техніки. Наша цивілізація нестримно прямує до комп'ютерної ери. Відбувається перехід до *інформаційних технологій*, тобто до широкого застосування комп'ютерів і програмного забезпечення у виробництві, управлінні, науці, освіті, медицині, торгівлі, банківській справі тощо. В інформаційному суспільстві щораз більше людей працює в галузях опрацювання інформації, а не у сфері матеріального виробництва. Сьогодні комп'ютери допомагають нам у різних сферах діяльності. Ключем до оволодіння багатьма сучасними перспективними професіями є комп'ютерна обізнаність. Комп'ютерна обізнаність – це уміння спілкуватися з комп'ютером. Таке спілкування передбачає знання апаратної частини, системного і прикладного програмного забезпечення, а також наявність навичок розв'язування типових задач навчального характеру. Програма курсу складається з розділу „Алгоритмізація і програмування". Цей розділ інформатики варто усіляко підтримувати і розвивати. Саме у цьому напрямку наші учні можуть отримати знання, що у перспективі зроблять їх конкурентноздатними на відповідному сегменті світового ринку праці. Певно не всі учні стануть програмістами, але вивчаючи саме цей розділ, вони мають змогу доторкнутися до найбільш високооплачуваної професії у світі, спробувати у ній свої сили. А для усіх інших: майбутніх домогосподарок, виконавців і керівників усіх ланок – будуть корисними методологічні засади цього розділу курсу, що ведуть до розвитку алгоритмічного мислення, яке слід застосовувати як під час переходу вулиці та керування транспортним засобом, так і в управлінні державою тощо. Сьогодні вже мало хто сумнівається в доцільності розширення і зміцнення позицій інформатики у школі. Адже розв'язування на уроках математики логарифмічних та тригонометричних рівнянь чи задач про кількість електронів, що пролітають між двома пластинами на уроках фізики, не дають жодних життєвих переваг для майбутньої професійної діяльності порівняно з задачами опрацювання і пошуку інформації, які розв'язують на уроках інформатики. Саме тут є можливість навчитися складну задачу зводити до послідовності простіших задач і конструювати алгоритм з окремих частин, не кидатися в житті у різні сторони та назад, а йти лише уперед до досягнення мети, чітко пояснювати свої дії оточуючим і працювати в колективі над великими проектами. Саме тут учні усвідомлюють, що алгоритм – не розкіш, а засіб досягнення мети.

Відомо, що вік восьмикласника є кризовим у плані психічного й фізіологічного розвитку молодої людини. Зокрема, відбувається перехід від предметно-образного (конкретно-образного, наочно-образного) до абстрактно-логічного (вербально-логічного, понятєвого, дискурсивного, теоретичного) мислення. Тому 8 клас найбільш придатний для початку систематичного і цілеспрямованого вивчення розділу «Основи алгоритмізації та програмування», адже з пропедевтичним курсом та основами комп'ютерних технологій учні вже ознайомлені з 5 по 7 клас.

До ***базових знань***, які повинні опанувати учні за цим курсом, віднесено: уміння формулювати базові задачі в термінах інформатики та планувати основні етапи їх розв'язання у визначеному програмному середовищі; знання базових структур Навчальної Алгоритмічної Мови, мови програмування Паскаль та розуміння їх особливостей.

До ***базових умінь і основних навичок***, які мають бути сформовані в учнів впродовж вивчення курсу, належать навички роботи в програмному середовищі; уміння складати прості алгоритми, реалізовувати їх у вигляді програм мовою НАМ та мовою Паскаль у середовищі програмування Turbo Pascal 7.0; уміння налагоджувати програми та перевіряти правильність їх роботи; уміння раціонально обирати та використовувати прикладне програмне забезпечення для виконання простих навчально–пізнавальних завдань.

Головна мета курсу – оволодіння учнями технікою доказових міркувань при аналізі алгоритмів і побудові планів розв'язування задач.

У результаті навчання за даним курсом

учні повинні знати:

- базові алгоритмічні структури, поняття математичної моделі задачі, алгоритму, програми;
- основні конструкції мови програмування Turbo Pascal 7.0;
- алгоритми розв'язування базових задач;
- прийоми ефективного використання комп'ютерних ресурсів при розв'язування базових задач;

учні повинні вміти

- виявляти й усувати неоднозначність в умовах задач;
- будувати математичну модель задачі;
- розробляти алгоритми та складати програми для розв'язування задач, як базових, так і однакових за рівнем складності;
- налагоджувати та тестувати програми.

2. Основні складові змісту курсу

«Основи алгоритмізації та програмування. 9 клас»

Навчально-методичне забезпечення курсу становлять такі програмні засоби:

1. Діалоговий інтерпретатор (компілятор) мови програмування Turbo Pascal 7.0 – для реалізації та налагодження програм на комп'ютері.

№	Тема	Кількість годин
1	Основи алгоритмізації	4 (8)
2	Основні поняття мови Паскаль	8 (16)
3	Організація розгалужень	7 (14)
4	Організація циклів	14 (28)
5	Підсумкове заняття	1 (2)
6	Резерв часу	1 (2)
	Усього	35 (70)

3. Зміст навчального матеріалу та вимоги до навчальних досягнень учнів

Зміст навчального матеріалу, кількість годин	**Вимоги до рівня загальноосвітньої підготовки учня**
Тема 1. Основи алгоритмізації, 4 (8) o Поняття алгоритму. Властивості алгоритмів. Основні базові структури алгоритмів. Виконавець алгоритму. Система команд виконавця o Мови програмування та комп'ютерні програми o Навчальна Алгоритмічна Мова (НАМ) o Службові слова o Загальна структура програми, записаної НАМ o Величини. Змінні та сталі величини o Ім'я величини та правила його записування o Типи змінних величин, визначені у НАМ (цілі, дійсні, літерні, логічні). Правила записування розділу опису сталих та змінних величин у НАМ o Аргументи, результати, проміжні величини. Оператор присвоєння	**Учні повинні знати:** o Поняття алгоритму та його властивостей o Поняття виконавця алгоритму та системи команд виконавця o Поняття мови програмування та комп'ютерної програми o Поняття службових слів, величин, змінних і сталих величин o Арифметичні дії та їх позначення, які використовують в НАМ o Процедури введення та виведення змінних в НАМ o Властивості алгоритмів o Основні базові структури алгоритмів o Загальну структуру програми, записаної НАМ o Характеристики величин o Види і типи величин o Призначення розділу опису величин у тексті програми o Призначення та дію оператора присвоєння o Поняття математичної моделі процесу, етапи розв'язування задач на комп'ютері o Призначення та використання впорядкування тексту програми o Правила записування числа у звичайному вигляді, якщо у програмі використовувався стандартний вигляд **Учні повинні вміти:** o Складати та виконувати алгоритми для розв'язування задач з повсякденної людської діяльності, визначати однозначність, закінченість, масовість алгоритму

o Процедури введення та виведення значень величин o Лінійні алгоритми o Поняття математичної моделі o Етапи розв'язування задач за допомогою комп'ютера. Метод покрокової деталізації	o Визначати цілі, яких може досягти виконавець на підставі його припустимих дій; складати алгоритми розв'язання нечислових задач для конкретного виконавця o Використовуючи позначення арифметичних дій, записувати за правилами НАМ арифметичні вирази o Визначати поточні значення змінних величин після виконання записаної серії обчислень o Визначати необхідну для розв'язування задачі вхідну інформацію, обчислювати поточне значення величини та остаточний результат, розрізняти типи величин o Записувати умову у вигляді відношення, перевіряти істинність умови, виконувати вказівку розгалуження, підбирати тестові значення для перевірки вказівки розгалуження, складати та виконувати алгоритми з розгалуженнями o Словесно описувати алгоритм з повторенням, складати протоколи виконання циклічних алгоритмів, застосовувати цикли при розв'язуванні задач o Аналізувати умову задачі, будувати математичну модель процесу, визначати метод розв'язування та конструювати алгоритм розв'язання задачі
Тема 2. Основні поняття мови Паскаль 8 (16) o Мова програмування Turbo Pascal 7.0 o Основні поняття мови Turbo Pascal 7.0 o Прості типи мови Turbo Pascal 7.0 o Створення лінійних програм o Модуль CRT o Графічний режим роботи o Поняття системи числення. Арифметичні дії в двійковій системі числення o Елементи алгебри логіки o Обчислення значень логічних виразів	**Учні повинні знати:** o Основні режими роботи з файлами в інтерактивному середовищі Turbo Pascal 7.0 ; основні відомості про роботу у вбудованому редакторі Turbo Pascal 7.0; поняття про компіляцію та виконання програм; порядок діагностування помилок o Загальну характеристику мови Паскаль; алфавіт мови, основні поняття мови: ідентифікатор, величина, оператор, вираз: правила лінійного запису арифметичних виразів o Поняття величини в програмуванні; необхідність опису типів змінних і констант; правила запису арифметичних виразів; правила перетворення і сумісності числових типів o Загальну структуру програми, призначення розділів опису; синтаксис та семантику операторів введення/виведення; правила форматування при виведенні o Правила організації керування текстовим виведенням на екран, процедури керування звуковими сигналами o Правила ініціалізації графічного режиму, типові процедури для створення графічних примітивів o Поняття позиційної системи числення; поняття основи системи числення; правила двійкової арифметики o Поняття й апарат алгебри висловлювань: істинне висловлення, хибне висловлення, предикати, зв'язування; основні типи логічних зв'язувань o Основні типи логічних зв'язувань, логічний сенс основних тотожностей, алгоритм розв'язування логічних задач **Учні повинні вміти:** o Користуватися командами меню та «гарячими клавішами» для роботи з файлами, роботи з вікнами, отримання довідки, виконання програм o Записувати математичні вирази в лінійній формі, обчислювати поточне значення величини та остаточний результат o Правильно описувати величини; організовувати введення та виведення даних, програмувати простий діалог

	o Використовувати колір фону та тексту для оформлення виведення інформації, визначати текстові вікна, процедури керування курсором для організації діалогу o Переходити в графічний режим роботи, створювати графічні примітиви, користуватися процедурами заливки, використовувати графічні процедури для побудови діаграм, графіків функцій o Переводити числа з десяткової системи в двійкову і назад; знаходити суму і різницю двійкових чисел o Складати таблиці істинності для найпростіших логічних формул; записувати твердження природною мовою у вигляді логічних формул o Записувати умову логічної задачі у вигляді системи логічних виразів
Тема 3. Організація розгалужень 7 (14) o Вказівка розгалуження o Лист завдань o Оператор варіанта o Тип, який перераховується. Тип-діапазон	**Учні повинні знати:** o Принципи дії оператора IF; синтаксис команди IF o Галуження на три та більше частини; повну та скорочену форми команди варіанта для запису розгалужень o Правила опису та завдання змінних обмежених типів, правила опрацювання змінних обмежених типів, переваги застосування в програмах типів користувача **Учні повинні вміти:** o Обирати дії в залежності від виконання умови; виконувати, змінювати, виправляти алгоритми з розгалуженнями; редагувати Паскаль-програму, складати протокол обчислювального експерименту, оптимізувати пошук розв'язання за допомогою методу половинного ділення o Виконувати повну та скорочену форму команди варіанта o Виконувати операції над змінними обмежених типів, використовувати для введення/виведення оператор варіанта
Тема 4. Організація циклів 14 (28) o Вказівка повторення. Організація циклів з пост- та передумовою o Обчислення сум за допомогою циклів o Цикл із параметром o Обробка рекурентних послідовностей o Послідовність Фібоначчі. Правила утворення елементів цієї послідовності o Складання алгоритму з одним циклом o Вкладені цикли	**Учні повинні знати:** o Структуру циклу з пост- і передумовою; правила запису команд повторення WHILE і REPEAT; o Загальні принципи додавання членів різних послідовностей o Синтаксис і правила застосування оператора циклу з параметром **Учні повинні вміти:** o Складати протоколи виконання циклічних алгоритмів; переходити від блок-схеми і запису на НАМ до запису мовою Паскаль; задавати початкові значення змінним, які використовуються в циклі; вибирати придатний для даної задачі варіант команди повторення o Аналізувати текст програми та вносити необхідні зміни; задавати початкові значення змінним, призначеним для збереження суми, доданку, лічильника; обчислювати нове значення змінної-доданку, вибирати придатний для даної задачі варіант

	команди повторення o Виконувати, змінювати і складати циклічні алгоритми із заздалегідь відомою кількістю повторень o Обчислювати в циклі n-й член рекурентної послідовності через (n-1)-й член
Підсумкове заняття 1 (2)	
Резерв часу 1 (2)	

4. Критерії оцінювання рівня навчальних досягнень учнів з курсу «Основи алгоритмізації та програмування. 9 клас»

Рівень навчальних досягнень	**Бали**	**Критерії оцінювання навчальних досягнень учнів**
І. Початковий	1	Учень: • Описує поняття алгоритму, наводить найпростіші приклади з життя, де трапляються алгоритми
	2	• Визначає властивості алгоритмів, має початкове уявлення про способи подання алгоритмів та програм
	3	• Називає і записує за визначеними правилами основні алгоритмічні структурі • У задачах розрізняє аргументи й результати, сталі й змінні величини • Уміє запускати програму середовища програмування
ІІ. Середній	4	• Має уявлення про етапи розв'язання задач і записування програм • Розрізняє і наводить приклади алгоритмів, що містять різні алгоритмічні структури • Розуміє призначення таблиці виконавця • Уміє набирати тексти готових програм у середовищі програмування
	5	• За розібраним із вчителем планом може самостійно записати алгоритм, що містить різні алгоритмічні структури • Уміє користуватися шаблонами для введення тексту програми • Розуміє призначення процесу компіляції програми та описує послідовність подальших дій у разі появи в цьому процесі помилки
	6	• Зберігає текст програми у файлі та відкриває файл, що містить текст програмі у середовищі програмування • Уміє впорядковувати текст програми • Організовує зрозуміле введення аргументів та виведення результатів роботи програми • Уміє запускати програму на виконання та переглядати отримані результати • Самостійно складає план розв'язання задачі на використання лінійних

		алгоритмів та записує програму за правилами вивченої мови програмування • Уміє визначати значення змінних після виконання лінійного блоку програми • Уміє записувати прості логічні умови та визначати їхні значення • Розробляє і заповнює таблицю виконавця для лінійних алгоритмів
ІІІ. Достатній	7	• З допомогою вчителя чи за попередньо розробленим планом уміє записувати алгоритми, що містять оператори розгалуження і прості оператори повторення • Уміє записувати складені логічні умови та визначати їх значення • З допомогою вчителя вміє записувати програми, що містять вкладені оператори чи складні оператори розгалуження • Уміє заповнювати таблицю виконавця для програм, що містять оператори розгалуження • Користуючись запропонованими тестами, уміє перевіряти правильність отриманих результатів роботи програми • Розуміє призначення та вміє використовувати метод покрокового виконання програми • Описує різні види операторів повторення, знає їхні особливості
	8	• Пояснює спільні риси та особливості різних операторів повторення, оцінює доцільність використання певного виду циклу • Самостійно розробляє план розв'язання задачі, що передбачає використання операторів розгалуження та повторення • З допомогою вчителя записує розв'язання задачі з використанням операторів розгалуження та різних видів циклів • Складає та заповнює таблицю виконавця для алгоритмів складної структури • Оцінює правильність отриманих результатів роботи програми, користуючись системою наведених тестів, розробляє власні тести для перевірки правильності роботи програми • Обраховує значення проміжних величин у процесі виконання складних програм та перевіряє їх у покроковому режимі виконання програми
	9	• Розуміє технологію методу послідовного уточнення, уміє його застосовувати для розв'язання базових задач, визначених програмою • Уміє зображати за допомогою блок-схем складені алгоритмічні структури, потрібні для розв'язання задачі • Користуючись допомогою вчителя, розробляє різні варіанти розв'язання задачі

		та оцінює їх ефективність
IV. Високий	10	• Самостійно описує використані в програмі величини, визначає їхні типи, організовує їх введення та виведення на екран • Складає план розв'язання задачі, користуючись методом покрокової деталізації • У процесі розв'язування задачі користується (в разі потреби) всіма вивченими алгоритмічними структурами
	11	• Самостійно розв'язує запропоновані задачі, різними методами перевіряє правильність отриманих розв'язань, розробляє систему тестових прикладів для перевірки правильності роботи програми, аналізує отримані результати • Оцінює ефективність отриманого розв'язання (алгоритму та програми) і пропонує шляхи поліпшення отриманого розв'язання • Володіє всім теоретичним матеріалом (у межах програми) та вміє самостійно застосовувати набуті знання на практиці
	12	• Має стійкі знання та навички роботи з вивченим середовищем програмування, у процесі розв'язання задач проявляє творчий підхід та самостійність, для розв'язання задач вміє користуватися знаннями, отриманими на інших уроках та самостійно

5. Додатки

Орієнтовне поурочне планування навчального процесу з курсу

«Основи алгоритмізації та програмування. 9 клас»

№ п/п	Зміст уроку
1. Основи алгоритмізації 4 (8)	
1 (1, 2)	Алгоритми, властивості алгоритмів. Виконавці алгоритмів
2 (3, 4)	Загальні правила алгоритмічної мови. Величини. Вказівка присвоювання.
3 (5, 6, 7)	Алгоритми з розгалуженнями. Команда повторення. Етапи розв’язування задач.
4 (8)	Тематична атестація «Алгоритми»
2. Основні поняття мови Паскаль 8 (16)	
5 (9, 10)	Мова програмування Turbo Pascal 7.0. Основні поняття мови Паскаль

6 (11, 12)	Прості типи мови Паскаль
7 (13, 14)	Створення лінійних програм
8 (15, 16)	Практична робота №1
9 (17, 18)	Модуль CRT. Графічний режим роботи.
10 (19, 20)	Практична робота №2
11 (21, 22, 23)	Елементи алгебри логіки. Обчислення значень логічних виразів.
12 (24)	Тематична атестація «Основні поняття мови Паскаль»
3. Організація розгалужень 7 (14)	
13 (25, 26)	Вказівка розгалуження
14 (27, 28)	Лист завдань
15 (29, 30)	Оператор варіанта
16 (31, 32)	Тип, який перераховується. Тип-діапазон.
17 (33, 34)	Практична робота №3
18 (35, 36)	Практична робота №4
19 (37, 38)	Тематична атестація «Розгалуження»
4. Організація циклів 14 (28)	
20, 21	Вказівка повторення. Організація циклів. Цикл із пост- і передумовою.

(39-41)	
22 (42, 43)	Обчислення сум за допомогою циклів.
23, 24 (44-46)	Цикл із параметром
25, 26 (47-49)	Обробка рекурентних послідовностей
27 (50, 51)	Складання алгоритму з одним циклом
28 (52-56)	Вкладені цикли.
29,30 (57-62)	Задачі з вкладеними циклами
31 (63)	Практична робота №5
32 (64)	Практична робота №6
33 (65, 66)	Тематична атестація «Цикли»
34 (67, 68)	Підсумкове заняття
35 (69, 70)	Резерв часу

Література

1. Програма для загальноосвітніх навчальних закладів. Інформатика. – Запоріжжя: Прем'єр, 2003.
2. Концепція загальної середньої освіти (12-річна школа). // Інформаційний збірник Міністерства освіти і науки України. Січень 2002. №2. – К.: Педагогічна преса. 2002.
3. Бондаренко О.О., Мірошниченко А.А. Інформатика. Основи програмування мовою Паскаль для 8 – 9 класів. – Шепетівка: «Аспект», 2006.
4. Янковой В.А., Крижевская А.Н. Язык программирования Turbo Pascal 7.0. – Одесса: ОНМЦОКИТ, 2002

5. Караванова Т.П. Інформатика. Основи алгоритмізації та програмування. 777 задач з рекомендаціями та прикладами. Навчальний посібник для 8 – 9 класів із поглибленим вивченням інформатики. – Київ: «ҐЕНЕЗА», 2006

Календарне планування

	Тема уроку	Дата проведення
	1. Основи алгоритмізації	
1 2	Алгоритми, властивості алгоритмів. Виконавці алгоритмів	
3 4	Загальні правила алгоритмічної мови. Величини. Вказівка присвоювання.	
5 6 7	Алгоритми з розгалуженнями. Команда повторення. Етапи розв'язування задач.	
8	Тематична атестація «Алгоритми»	
	2. Основні поняття мови Паскаль	
9 10	Мова програмування Turbo Pascal 7.0. Основні поняття мови Паскаль	
11 12	Прості типи мови Паскаль Самостійна робота	
13 14	Створення лінійних программ Розв'язування задач на створення лінійних програм.	
15 16	Практична робота №1	
17 18	Модуль CRT. Графічний режим роботи. Елементи комп'ютерної графіки	
19 20	Практична робота №2	
21	Тематична атестація «Основні поняття мови Паскаль»	
	3. Організація розгалужень	
22 23	Елементи алгебри логіки. Обчислення значень логічних виразів.	
24 25 26 27	Вказівка розгалуження Повне розгалуження Розв'язування задач Розв'язування задач	
28	Практична робота №3	
29	Самостійна робота	
30 31 32	Оператор вибору Розв'язування задач Розв'язування задач	
33	Практична робота №4	
34	Оператор безумовного переходу. Мітки	
35 36	Самостійна робота Розв'язування задач	
37 38	Тематична атестація «Розгалуження»	
	4. Організація циклів	
39 40 41 42	Вказівка повторення. Організація циклів. Цикл із пост- і передумовою. Цикл із параметром Розв'язування задач	
43 44 45	Обчислення суми, добутку та кількості. Розв'язування задач Розв'язування задач	
46 47	Методи перебирання варіантів Розв'язування задач	
48 49 50	Табулювання функції Розв'язування задач Самостійна робота	
51	Практична робота №5	
52	Пошук максимального чи мінімального значення Розв'язування задач	

53		
54 55 56 57 58	Вкладені цикли. Задачі з вкладеними циклами Розв'язування задач Розв'язування задач Розв'язування задач	
59 60	Обробка рекурентних послідовностей Поняття про метод ітерацій. Числа Фібоначчі	
61 62	Самостійна робота	
63 64	Практична робота №6	
65 66	Тематична атестація «Цикли»	
67 68	Підсумкове заняття	
69 70	Резерв часу	

1. Основи алгоритмізації

Урок 1. Тема. Алгоритми. Властивості алгоритмів.

„Алгоритм не розкіш, а засіб досягнення мети”

Епіграф до уроку:

Коль кругом все будет мирно,
Так сидеть он будет смирно;
Но лишь чуть со стороны
Ожидать тебе войны,
Иль набега силы бранной,
Иль другой беды незваной,
Вмиг тогда мой петушок
Приподымет гребешок,
Закричит и встрепенется
И в то место обернется.
А.С.Пушкин

Мета.

Навчальна:

- дати поняття про алгоритм, його властивості;
- навчити розпізнавати алгоритми навколо себе;
- вміти розрізняти правильно та неправильно сформульовані алгоритми;
- формувати цілісну уяву про картину всесвіту;
- формувати науковий світогляд;
- синтезувати знання, отримані при вивченні різних шкільних предметів.

Розвивальна:

- розвивати логічне мислення, пізнавальний інтерес;
- формувати вміння аналізувати, узагальнювати, порівнювати, абстрагуватися, синтезувати знання, отримані при вивченні різних предметів.

Виховна:

- виховувати прагнення до отримання нових знань;
- узагальнювати знання з різних областей життя;
- виховувати почуття товариськості, взаємовиручки;
- виховувати комунікативні якості, вміння слухати;
- виховувати критичне відношення до загальноприйнятих істин;
- виховувати культуру між особистісних взаємовідносин, акуратність в роботі.

Задачі:

- Визначити наявність алгоритмів в шкільних предметах.
- Довести необхідність складання алгоритмів на любому навчальному предметі для кращого розуміння, засвоєння і запам’ятовування матеріалу.
- Показати значення інформатики для інших наук.

Тип уроку: вивчення нового матеріалу.

Хід уроку.

І. Вступна бесіда.

Багато хто вважає, що інформатика потрібна тільки для того, щоб навчитися працювати на комп’ютерах. Але цю помилкову думку ми постараємося спростувати на нашому уроці.

Кожна людина щодня зустрічається з безліччю задач від найпростіших і добре відомих до дуже складних. Для багатьох задач існують визначені правила (інструкції, команди), що пояснюють виконавцю, як розв'язувати дану проблему. Ці правила людина може вивчити чи заздалегідь сформулювати сама в процесі розв'язування задачі. Чим точніше описані правила, тим швидше людина опанує ними і буде ефективніше їх застосовувати.У нашому житті ми постійно складаємо опис деякої послідовності дій для досягнення бажаного результату, тому поняття алгоритму не є для нас чимось новим і незвичайним. Так, ранком мама перед твоїм виходом до школи, дає вказівку: "Коли прийдеш зі школи, відразу пообідай і вимий посуд. Після цього підмети підлогу, сходи в магазин і можеш трохи погуляти. Гуляти дозволяю не більше години, а потім відразу за уроки".

Ця інструкція складається з послідовності окремих вказівок, що і визначають твою поведінку після повернення зі школи. Це і є алгоритм.

Кожний з нас використовує сотні різних алгоритмів. Спробуйте згадати деякі з них (алгоритми виконання арифметичних дій, розв'язування задач, прибирання квартири, миття посуду, готування їжі - рецепти тощо). Отже, після обговорення кількох прикладів алгоритмів, давайте спробуємо сформулювати визначення, що ж таке алгоритм.

Саме слово алгоритм походить від algorithmi – латинської форми написання імені великого математика IX ст. аль-Хорезмі, який сформулював правила виконання арифметичних дій. Спочатку під алгоритмами і розуміли тільки правила виконання чотирьох арифметичних дій над багатоцифровими числами. В подальшому це поняття стали використовувати взагалі для позначення послідовності дій, які приводять до розв’язання задачі. *Алгоритмом називають зрозуміле і точне розпорядження виконавцю про виконання послідовності дій, спрямованих на досягнення зазначеної мети чи на вирішення поставленої задачі.*

В цьому означенні використовується поняття "виконавець". Що це означає? Під виконавцем алгоритму ми розуміємо будь-яку істоту (живу чи неживу), яка спроможна виконати алгоритм. Все залежить від того, якої мети ми намагаємося досягнути. Наприклад: риття ями (виконавці - людина або екскаватор), покупка деяких товарів (один із членів родини), розв'язування математичної задачі (учень або комп'ютер) тощо.

Поняття алгоритму в інформатиці є фундаментальним, тобто таким, котре не визначається через інші ще більш прості поняття (для порівняння у фізиці - поняття простору і часу, у математиці - точка).

Будь-який виконавець (і комп'ютер зокрема) може виконувати тільки обмежений набір операцій (екскаватор копає яму, вчитель вчить, комп'ютер виконує арифметичні дії). Алгоритмічне мислення допомагає чітко побачити кроки, що ведуть до мети, замітити всі перешкоди і уміло їх обійти.

Тому алгоритми повинні мати певні властивості, разом з тим, не кожна інструкція або послідовність дій може називатися алгоритмом.

Отже, сформулюємо ***основні властивості алгоритму***.

1. Зрозумілість. Щоб виконавець міг досягти поставленої перед ним мети, використовуючи даний алгоритм, він повинен уміти виконувати кожну його вказівку, тобто розуміти кожну з команд, що входять до алгоритму.

Наприклад: Мамі потрібно купити в магазині їжу. Виконавцем цього алгоритму може бути хтось із родини: батько, син, бабуся, маленька дочка тощо. Зрозуміло, що для тата достатньо сказати, які купити продукти, а далі деталізувати алгоритм не потрібно. Дорослому сину-підлітку необхідно детальніше пояснити в яких магазинах можна придбати потрібний товар, що можна купити замість відсутнього товару і таке інше. Маленькій дочці алгоритм необхідно деталізувати ще більше: де взяти сумку, щоб принести товар, яку решту грошей необхідно повернути з магазину, як дійти до магазину і як там поводитись (якщо дитина вперше йде за покупками).

Подібних прикладів можна навести безліч і запропонувати дітям самостійно підібрати ситуацію, в якій в залежності від виконавця алгоритм буде набувати все більшої деталізації. Висновок з цього діти можуть зробити самостійно: *зрозумілість* - це властивість алгоритму, що полягає в тім, що кожен алгоритм повинен бути написаний у командах, зрозумілих даному виконавцю.

2. Визначеність (однозначність). Зрозумілий алгоритм все ж таки не повинен містити вказівки, зміст яких може сприйматися неоднозначно. Наприклад, вказівки "почисти картоплю", "посоли за смаком", "прибери в квартирі" є неоднозначними, тому що в різних випадках можуть призвести до різних результатів. Крім того, в алгоритмах неприпустимі такі ситуації, коли після виконання чергового розпорядження алгоритму виконавцю не зрозуміло, що потрібно робити на наступному кроці. Наприклад: вас послали за яким-небудь товаром у магазин, та ще попередили "без (хліба, цукру і таке інше) не повертайся", а що робити, якщо товар відсутній?

Отож, *точність* - це властивість алгоритму, що полягає в тім, що алгоритм повинен бути однозначно витлумачений і на кожному кроці виконавець повинен знати, що йому робити далі.

3. Дискретність. Як було згадано вище, алгоритм задає повну послідовність дій, які необхідно виконувати для розв'язання задачі. При цьому, для виконання цих дій їх розбивають у визначеній послідовності на прості кроки. Виконати дії наступного розпорядження можна лише виконавши дії попереднього. Ця розбивка алгоритму на окремі елементарні дії (команди), що легко виконуються даним виконавцем, і називається дискретністю.

4. Масовість. Дуже важливо, щоб складений алгоритм забезпечував розв'язання не однієї окремої задачі, а міг виконувати розв'язання широкого класу задач даного типу. Наприклад, алгоритм покупки якого-небудь товару буде завжди однаковий, незалежно від товару, що купується. Або алгоритм прання не залежить від білизни, що переться, і таке інше. Отож, під масовістю алгоритму мається на увазі можливість його застосування для вирішення великої кількості однотипних завдань.

5. Результативність. Взагалі кажучи, очевидно, що виконання будь-якого алгоритму повинне завершуватися одержанням кінцевих результатів. Тобто ситуації, що в деяких випадках можуть призвести до так званого "зациклення", повинні бути виключені при написанні алгоритму. Наприклад, розглянемо таку ситуацію: роботу дано завдання залишити кімнату (замкнутий простір), не виконуючи руйнівних дій. У цьому випадку, якщо роботу не дати вказівки відкрити двері (що, можливо, закриті), то спроби залишити приміщення можуть бути безуспішними. **Ефективність** - кожний крок алгоритму повинен бути виконаний точно за скінчений проміжок часу.

Примітка: У процесі та по закінченні викладання матеріалу дітям пропонується навести приклади інструкцій, що не відповідають визначенню алгоритму чи не володіють властивостями алгоритму.

Для роботи багатьох програм необхідно задавати початкові значення. Ці значення передаються в алгоритм за допомогою аргументів.

Аргументи - це величини, значення яких необхідно задати для виконання алгоритму. Правда, деколи зустрічаються алгоритми, що не вимагають ніяких початкових значень для свого виконання. Пізніше буде нагода познайомитися з такими алгоритмами. Однак, немає жодного алгоритму, що не дає ніякого результату. Дійсно, який же зміст у такому алгоритмові? Прикладом різноманітності результатів роботи програм є ігрові комп'ютерні програми. Одержувана ними під час роботи закодована інформація певним чином перетворюється у графічні та звукові образи.

Результати - це величини, значення яких одержуються внаслідок виконання алгоритму.

При складанні багатьох алгоритмів виникає необхідність окрім аргументів та результатів використовувати ще додаткові величини. Введення в алгоритм таких величин залежить від самого автора алгоритму.

Проміжні величини — це величини, які додатково вводяться в ході розробки алгоритму.

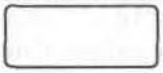
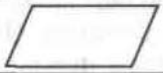
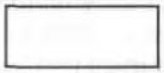
Тепер залишається з'ясувати, яким чином можна подати алгоритм виконавцю. Існує кілька методів запису алгоритмів, вибір яких залежить від виконавця та того, хто його задає.

Перший спосіб - це ***словесний опис*** алгоритму. Сьогодні на уроці розібрано вже кілька алгоритмів, і всі вони подавалися виконавцю за допомогою словесного опису.

Другий спосіб - це подача алгоритму у вигляді ***таблиць, формул, схем, малюнків*** тощо. Наприклад, всіх вас вчили в дитячому садочку правилам поведінки на дорозі. І найкраще діти, вочевидь, сприймають алгоритм, що поданий у вигляді схематичних малюнків. Дивлячись на них, дитина, а потім і доросла людина, відпрацьовує ту лінію поведінки, що їй пропонується. Аналогічно можна навести приклади алгоритмів, що записані у вигляді умовних позначок на купленому товарі, щодо його користування (заварювання чаю, прання білизни тощо). В математиці наявність формул дозволяє розв'язати задачу, навіть "не використовуючи слів".

Третій спосіб - запис алгоритмів за допомогою ***блок-схеми***. Цей метод був запропонований в інформатиці для наочності представлення алгоритму за допомогою набору спеціальних блоків. Основні з цих блоків наступні:

Четвертий спосіб - ***навчальні алгоритмічні мови*** (псевдокоди). Ці мови мають жорстко визначений синтаксис і вже максимально наближені до машинної мови (мови програмування). Але створені вони з навчальною метою, тому мають зрозумілий для людей вигляд. Таких псевдокодів зараз існує велика кількість, починаючи

	Початок і кінець алгоритму
	Введення/виведення даних
	Вибір напрямку виконання алгоритму залежно від деяких змінних умов
	Виконання операцій, у результаті яких відбувається зміна значення даних

з графічних середовищ "Алгоритміка", "Роботландія", "Лого-світи", "Черепашка" тощо і закінчуючи текстовими "національними" реалізаціями алгоритмічних мов, подібних до Паскаля. Ці псевдокоди мають програмну реалізацію і дуже широко застосовуються на етапі навчання основам програмування.

П'ятий спосіб максимально наближений до комп'ютера - це ***мови програмування***. Справа в тому, що найчастіше в практиці виконавцем створеного людиною алгоритму являється машина і тому він повинен бути написаний мовою, зрозумілою для комп'ютера, тобто мовою програмування.

Приклади алгоритмів

1. Знайти найбільший спільний дільник двох натуральних чисел m і n (алгоритм Евкліда). Складемо алгоритм розв'язання цієї задачі, який базується на тій властивості, що якщо m>n, то найбільший спільний дільник чисел m,n такий самий, як і чисел m-n, n.

Алгоритм буде таким:

1) якщо числа рівні, то взяти любе з них за відповідь, в іншому випадку продовжити виконання алгоритму;
2) визначити більше із чисел;
3) замінити більше число різницею більшого і меншого чисел;
4) почати алгоритм спочатку.

2. Алгоритм «відгадування» задуманого числа. Нехай хто-небудь задумає довільне натуральне число. Йому пропонується провести з цим числом слідуючі дії і потім повідомити результат:

1) помножити задумане число на 5;
2) додати 8;
3) суму помножити на 2.

Необхідно за результатом «відгадати» задумане число.

Розв'язання даної задачі зводиться до розв'язання рівняння $(x\cdot5+8)\cdot2=a$, де x – невідоме задумане число, а – отриманий результат.

«Відгадування» x можна доручити виконавцю, зовсім незнайомому із змістом задачі. Для цього достатньо повідомити йому слідуючий алгоритм:

1) відняти від результата 16;
2) в отриманій різниці відкинути крайню праву цифру, отримане число і буде шуканим.

Виконуючи алгоритм, виконавець може не вникати в зміст того, що він робить, і разом з тим отримати потрібний результат. У цьому випадку говорять, що виконавець діє формально.

IV. Підсумок уроку.

Домашнє завдання:

- прочитати сторінки запропонованого підручника;
- вивчити означення, що прочитані на лекції (що таке алгоритм, властивості алгоритму, способи подачі алгоритму);

- придумати будь-який алгоритм на побутову тему (кулінарний, прибирання кімнати, виконання уроків тощо);
- продумати варіанти, коли в запропонованих алгоритмах можуть не виконуватися властивості алгоритмів.
- записати алгоритм знаходження середини відрізка за допомогою циркуля лінійки.
- скільки разів буде виконуватися крок 3 алгоритма Евкліда для m=100, n=18?

2. Виконавці алгоритмів

Мета уроку: Дати поняття про виконавця, характеристик виконавця, моделювання та створення інформаційних моделей об'єктів; навчити*:* визначати цілі, яких може досягти виконавець, складати алгоритми розв'язування задач для визначеного виконавця.
Тип уроку Лекційний.

Хід уроку.

І. Актуалізація опорних знань учнів.

Фронтальне опитування

1. Яке походження терміна «алгоритм»?
2. Що ми розуміємо під поняттям «алгоритм»?
3. Що таке допустимі команди виконавця?
4. Які є способи опису алгоритмів?
5. Які властивості повинен мати алгоритм?
6. Що означає скінченність (дискретність) алгоритму?
7. Що таке формальність алгоритму?
8. Що означає масовість алгоритму?

ІІ. Викладання нового навчального матеріалу.

1. Виконавець алгоритму

Виконавцем алгоритму може бути людина, машина, комп'ютер, система людина-машина, верстат-автомат, робот тощо, яких «навчено» виконувати вказівки алгоритму.

Характеристики виконавця

Середовище – «місце проживання» виконавця.

Припустимі дії – обмежений набір дій, що вміє виконувати даний виконавець. Описати виконавця – значить вказати його припустимі дії. Досяжні цілі – результати, що виконавець може одержати за допомогою своїх припустимих дій.

Система команд виконавця – суворо заданий список вказівок, як виконати визначені дії. Виконавця можна представити у виді пристрою з кнопками, де кожна кнопка відповідає одній команді. Натискання кнопки означає виклик команди.

Відмова – виникає при виклику команди в неприпустимому для даної команди стані середовища.

При побудові алгоритму часто виникає необхідність пояснити виконавцю деякі складні дії, якщо їх виконання не входить в систему команд виконавця. Наприклад, перший раз даючи дитині завдання пришити ґудзик до плаття, їй треба пояснити, як необхідно підбирати нитки для шиття, як вдягати нитку в голку, як тримати голку та ґудзик при роботі, яка різниця між пришиванням ґудзика до тоненької сорочки та товстої куртки (в другому випадку ґудзик робиться на "ніжці"). В подальшому такі пояснення будуть вже зайві, бо алгоритм "пришивання ґудзика" стає вже командою в системі команд виконавця "дитина".

Взагалі кажучи кожна дія людини (якщо вона її може виконати) може вважатися командою її "системи команд", хоча колись, на етапі навчання, учитель або хтось інший ретельно пояснював, яку треба виконати послідовність дій, щоб досягти поставленої мети.

Узагальнюючи сказане, можна сказати, що кінець кінцем кожну задачу можна вважати окремою командою виконавцю, якщо його навчено виконувати поставлене завдання. Якщо ж виконавець не знає, як розв'язувати запропоновану задачу, виникає потреба розкласти її на такі підзадачі, що являються "посильними" для виконання, тобто входять до системи команд виконавця. Продовжуючи цей процес, остаточно отримують алгоритм, що складається з простих команд, зрозумілих виконавцю, або остаточно переконуються, що дана задача непосильна для вибраного виконавця, тому що в його системі команд не існує необхідних для цього команд. Наприклад, як би ми не деталізували алгоритм побудови багатоповерхової будівлі для дитини, задача кінець кінцем являється для неї непосильною.

Примітка: на даному етапі уроку можна дати дітям завдання придумати задачу, яка б була непосильною для вибраного виконавця (виконавцем може бути людина, комп'ютер, якийсь пристрій тощо). Наприклад, спробуйте створити алгоритм виконання ремонту кімнати, розрахований на виконавця "екскаватор".

Запропонований підхід до конструювання алгоритмів називається *методом покрокової деталізації зверху вниз*. Вочевидь, що при такому підході кожна операція остаточно буде подана у вигляді лише одного з трьох типів базових структур алгоритмів - лінійної (в літературі часто ця структура називається слідування), розгалуження або повторення (циклу). Степінь деталізації алгоритму при цьому сильно залежить від того, на якого виконавця його орієнтовано.

Досить складну конструкторську задачу неможливо розв'язати без поступового заглиблення в деталі. Подумайте, наприклад, як розробляється конструкція сучасного теплохода, автомобіля або літака. (Можна дати дітям можливість самостійно це продумати).

Розглянутий принцип конструювання алгоритмів не залежить від конкретних особливостей поставленої задачі та вибору виконавця. Проте набір команд системи команд вибраного виконавця суттєво впливає на ступінь деталізації алгоритму та, кінець кінцем, на його структуру.

Візьмемо, наприклад, простий алгоритм "переходу через вулицю". Взагалі для кожної дитини можна вважати це командою, що входить до її "системи команд". Але кожен пам'ятає, як в дитинстві батьки та вихователі неодноразово повторювали: якщо в місті, де необхідно перейти вулицю, є підземний перехід, скористуйся ним, якщо немає, відшукай місце, де є світлофор, і перейди вулицю, користуючись правилами; якщо немає ні підземного переходу, ні світлофора... (далі діти самостійно продовжать це алгоритм). Однак, навіть в цьому алгоритмі є необхідність дещо деталізувати. Наприклад, що значить "перейди вулицю, користуючись правилами", при наявності світлофора? А якщо алгоритм складається для зовсім маленької дитини, то що таке світлофор і як його шукати? А якщо виконавець взагалі прибулець з інших світів? Що таке "зелений", "червоний", "жовтий"? Що таке підземний перехід? Перелік питань можна доповнити нескінченою послідовністю непорозумінь.

Алгоритми, що складаються для розв'язування окремих підзадач основної задачі, називаються допоміжними.

Вони створюються при поділі складної задачі на прості або при необхідності багаторазового використання одного ж того набору дій в одному або різних алгоритмах. Допоміжний алгоритм повинен мати тільки один вхід та один вихід, причому того, хто користується ним, зовсім не цікавить, як реалізований цей алгоритм. Головне, щоб всі команди, що входять до складу допоміжного алгоритму входили до системи команд обраного виконавця. Зверніть увагу ще на те, що в реальному житті допоміжні алгоритми можуть виконувати, навіть, зовсім інші виконавці. Наприклад, якщо батьки вдома вирішили зробити ремонт, це не значить, що вони самостійно повинні зробити собі шпалери та клей. Алгоритми виробництва матеріалів існують і їх хтось виконує, а ми тільки користуємось результатами їх роботи.
Таким чином, можна вважати допоміжний алгоритм своєрідним "чорним ящиком", на вхід якого подаються деякі вхідні дані, а на виході ми отримуємо очікуваний результат. Головне чітко домовитись про правила оформлення вхідних даних та вигляд результату. Невиконання домовленостей може привести до збою у виконанні допоміжного алгоритму або до отримання неочікуваного результату.

Описаний метод послідовної деталізації лежить в основі технології структурного програмування і широко застосовується при використанні таких мов програмування, як Паскаль, С, С++ та інших.

При описуванні програми для комп'ютера мовами високого рівня допоміжні алгоритми реалізовуються у вигляді підпрограм. Правила опису, звернення до них та повернення в точку виклику визначаються конкретною мовою програмування. Для зручності часто використовувані підпрограми можна об'єднувати в бібліотечні модулі і при необхідності підключати їх в свої програми.

III. Закріплення теоретичного матеріалу.

Креслення різноманітних ліній із записом кроків в програмі «Кенгуру» («Сходинки») та «Лого-Миры»

IV. Підсумок уроку.

Домашнє завдання:

- вивчити означення, що прочитані на лекції (що таке допоміжний алгоритм, в чому сутність метода покрокової деталізації та проектування зверху вниз);
- придумати будь-який алгоритм, в якому в залежності від виконавця необхідна різний степінь деталізації;
- продумати приклади алгоритмів, для яких будь-який степінь деталізації не дозволить виконати їх заданим виконавцем.
- записати алгоритм, за допомогою якого можна перейти вулицю у місці переходу. На яких виконавців розрахований цей алгоритм?

Урок № 3 Загальні правила алгоритмічної мови

Мета уроку: ознайомити учнів із словником алгоритмічної мови; службовими словами, правилами послідовного запису команд в алгоритмі; визначенням лінійного алгоритму; навчити складати та виконувати прості алгоритми, однозначно формулювати вказівки виконання певних дій та записувати їх за правилами навчальної алгоритмічної мови.

Тип уроку: вивчення нового навчального матеріалу

Хід уроку.

І. Актуалізація опорних знань учнів.

Інформаційний диктант

1. Що ми називаємо алгоритмом? Як виник термін «алгоритм»?
2. Наведіть власний приклад алгоритму.
3. Запишіть основні властивості алгоритмів.
4. Що розуміється під формальним виконанням алгоритму?
5. Хто або що може бути виконавцем алгоритму?
6. Які характеристики виконавця вам відомі?

II. Викладання теоретичного матеріалу.

1. Поняття про навчальну алгоритмічну мову.

Алгоритмічна мова – це система позначень і правил для однакового і точного запису алгоритмів і їх виконання. Алгоритмічна мова визначає способи запису тексту алгоритму (синтаксис алгоритму) і правила інтерпретації записаного тексту виконавцем (семантика мови). З одного боку, алгоритмічна мова близька до звичайної, з іншого – включає в себе і математичну символіку: числа, позначення величин і функцій, знаки операцій. Правила алгоритмічної мови лежать в основі мов програмування для ПК. Тому вивчення алгоритмічної мови допоможе вам швидше опанувати мову програмування.

2. Алфавіт мови

Алфавіт мови — це символи, які дозволено до використання в мові. В навчальній алгоритмічній мові використовуються:

• 33 літери українського алфавіту;
• 26 латинських літер (від *A* до *2);*
• 10 арабських цифр;
• 28 спеціальних знаків (?, !, #, $,%тощо).

Звичайно ж, у конкретних мовах програмування допускається дещо інший набір символів (наприклад, алфавіт мови Паскаль не містить літер українського алфавіту). Кожний символ алфавіту мови програмування має свій числовий код. Яким кодам відповідають символи, наведено в таблицях кодування символів.

3. **Синтаксис мови**

Синтаксис мови — це правила написання команд, службових слів,розділових знаків.

Розділовими знаками можуть бути такі символи:

1) ; —розділовий знак між командами;
2) . — розділовий знак між цілою й дробовою частинами числа;
(«,») —використовуються у записі виразів підпрограм;
[«, «] — для запису індексованих змінних;
5), — розділовий знак між елементами списку або індексами масиву;
6) : — є елементом синтаксису деяких команд, а також використовується для задання розмірів масиву. У мові програмування Паскаль для задання розмірів масиву використовується дві крапки, розміщені поруч...

4. **Елементи мови**

Елементами мови є:

Символи — основні неподільні знаки, за допомогою яких записуються тексти.

Службові слова — скорочення деяких слів, за допомогою яких записуються алгоритми. Наприклад: АЛГ, АРГ, РЕЗ, ПОЧ, КІН та ін.

Команди — дії алгоритму. Вони бувають прості й складені.

5**. Об'єкти мови**

Об'єктами мови є:

1. Константи, змінні.
2. Допоміжні алгоритми (підпрограми — функції або процедури).

Константи — це постійні величини, які визначаються на початку програми та не змінюють свого значення в процесі розв'язання завдання.

Числова константа — це деяке число. Числа можуть бути цілими або дробовими й подаватися у звичайній або експонентній формі.

У звичайній формі запису дробового числа ціла частина від дробової відділяється крапкою, а не комою. Така форма запису числа називається *записом з фіксованою крапкою,* наприклад:

2, 2.4, **-7.12,** 0.3.

Для запису надто великих або надто малих чисел зручніше застосовувати експонентну форму запису. її називають *записом із плаваючою крапкою.* Вона складається з мантиси (можливо зі знаком), літери *E* десяткового порядку (цілого числа, можливо зі знаком), наприклад:

2.1E-5, IE6, 1E-9

Текстова константа *— це будь-яка послідовність символів, узята в лапки «"» або апострофи « '» (у мові Паскаль можливий тільки останній варіант)*

Усередині текстової константи можуть бути й лапки, але тоді їх також треба взяти в лапки. Приклади текстових констант:

"мама",
"1234",
"Корабель ""Ластівка""",
"Справа *306",
"Значення виразу =".

Тому надалі ми подаватимемо текстові константи в лапках або апострофах.

Змінні** — це величини, які можуть змінювати своє значення в процесі розв'язування завдання. **Змінна** та **комірка** в програмуванні — поняття тотожні, оскільки кожній змінній у пам'яті комп'ютера виділяється місце, яке називається **коміркою.

Змінні називаються ***ідентифікаторами.** Ім'я змінної має обов'язково починатися на літеру, а далі може йти послідовність літер або цифр та знаків підкреслення.*

Приклади ідентифікаторів змінних:

a, ai, **в2а, сума** (у Паскалі україномовні ідентифікатори заборонені).

Імена змінним дає автор алгоритму. Ім'я змінної бажано добирати таким чином, щоб воно за змістом підкреслювало її призначення.

Не можна давати змінним імена, що збігаються з іменами службових слів, зарезервованих у цій мові програмування.

Кожна змінна повинна мати певне значення. Якщо в процесі виконання алгоритму змінна величина не одержала конкретного значення, то її значення буде непередбаченим.

*Алгоритм, який використовується у складі іншого алгоритму, називають **допоміжним алгоритмом** стосовно алгоритму, в якому він використовується.*

Вираз, *«алгоритм, який використовується у складі іншого алгоритму»,* необхідно розуміти так:

• допоміжний алгоритм може бути у складі основного алгоритму;

• допоміжний алгоритм може подаватися у вигляді окремого файла на диску, а в основному алгоритмі встановлюється з ним зв'язок.

У будь-якій мові програмування є *фіксований набір стандартних функцій.* Однак за необхідності його можна розширювати, створюючи власні функції користувача. За допомогою об'єктів мови створюються вирази.

***Вираз** — це сукупність об'єктів мови, пов'язана деякими операціями для обчислення одного значення певного типу.*

Вирази бувають:

- арифметичні;
- алгебраїчні;
- логічні;
- літерні (рядкові, текстові).

***Арифметичні вирази** — це числа, з'єднані знаками арифметичних операцій (можливо, із застосуванням дужок).*

***Алгебраїчні вирази** — це числа, літери, функції, з'єднані знаками арифметичних операцій (можливо, із застосуванням дужок).*

***Логічні вирази** — це умови. Умови бувають прості й складені. **Прості умови** складаються з одного відношення між величинами.*

Складені умови** складаються з кількох простих умов, з'єднаних між собою логічними **операціями:

І (and) — кон'юнкція (логічне множення);

Або (or)— диз'юнкція (логічне додавання);

Не (not)— заперечення.

Приклади складених умов:

X>2 і X<4, a<3 АБО a<5, НЕa = 0.

Мовою Паскаль відповідно умови пишуться так:

(x > 2) **and** (x < 4), (a < 3) or (a<5), not(a) = 0 Значенням логічного виразу є **«Істина»** або **«Хибність».**

Літерні вирази** — це текстові константи, змінні, функції, що опрацьовують текст, з'єднані знаком плюс «+».Знак плюс означає операцію склеювання (конкатенацію). Значенням літерного виразу є **текстовий рядок.

Структура алгоритму:

```
алг заголовок алгоритму (список параметрів із вказівкою їх
      арг список аргументів
      рез список результатів
поч список проміжних результатів із вказівкою їх типів
      серія вказівок
кін
```

Лінійний алгоритм - алгоритм, який містить лише вказівки про безумовне виконання деякої операції, без повторень або розгалужень (просте слідування).

Приклад 1. Скласти алгоритм знаходження середнього арифметичного трьох чисел.	**Приклад 2.**
алг середнє арифметичне	алг створення програм
	поч
	складі алгоритм;

apі a,b,c Рез хг поч х к:=a+b+c; хг:=я/3 кін	написати програму; відлагоди і и програму на ПК; отримані рішення задачі; проаналпуішти **результатів** кін

Приклад 1. Записати алгоритмічною мовою алгоритм знаходження середини відрізка, якщо в систему виконавця входять звичні дії із циркулем і лінійкою.

алг розподіл відрізка *AB* навпіл

поч

поставити ніжку циркуля в точку *A*, встановити розчин циркуля рівним довжині відрізка *AB*,
провести коло, поставити ніжку циркуля в точку *B*, провести коло, через точки перетину кіл
провести пряму, відзначити точку перетинання цієї прямої з відрізком *AB*

кін

III. Закріплення нового матеріалу.

Запишіть алгоритми для розвязування задач:

1) деяке задане число х збільшити у 16 разів, використовуючи лише операції множення на 2 і додавання;
2) деяке задане число х зменшити на 8, використовуючи лише операцію віднімання 1 і 3;
3) складіть алгоритм обчислення площі трапеції;
4) запишіть алгоритм обчислення шляху, який долає автомобіль зі швидкістю v за час t.
5) на інший берег річки треба перевезти човном вовка, козу та капусту, виконуючи умову: не можна залишати разом у човні чи на березі вовка і козу, або козу і капусту (за один раз можна перевозити не більше одного об'єкта).

IV. Підсумок уроку.

Домашнє завдання:

- вивчити означення, що прочитані на лекції ;
- записати алгоритм обчислення площі прямокутного трикутника;
- задано два кути трикутника, складіть алгоритм, який визначає величину третього кута трикутника;
- задано діаметр кола, складіть алгоритм, який визначає довжину кола та площу круга.

Урок 4. ***Величини. Вказівка присвоювання***

Мета уроку: ввести поняття перетворення інформації при розв'язуванні задач, основні характеристики величин, правила виконання операції присвоювання; навчити визначати необхідну для розв'язування задачі вхідну інформацію, обчислювати поточне значення величини та остаточний результат, розрізняти типи величин.

Тип уроку: формування навичок.

Хід уроку.

І. Актуалізація опрних знань учнів.

Самостійна робота.

І варіант.

Скласти алгоритм, який визначає периметр прямокутника, якщо задано його площу і ширину.

ІІ варіант.

Скласти алгоритм, який визначає периметр прямокутника, якщо задано його площу і довжину.

ІІ. Викладання нового теоретичного матеріалу.

1. Поняття величини.

Величина - це об'єкт, який має стале або змінне значення.

Величини діляться на змінні й сталі. *Сталою* називається величина, значення якої не змінюється в процесі виконання алгоритму, а залишається тим самим, зазначеним у тексті алгоритму (наприклад, 15; 2,4; 3,14 і т.д.). *Змінною* називається величина, значення якої змінюється в процесі виконання алгоритму.

При написанні алгоритму для змінних величин вводяться позначення аналогічно позначенням змінних у курсі алгебри. Таке позначення змінної величини в алгоритмічній мові називається *ім'ям* величини.

При виконанні алгоритму в кожний момент часу величина звичайно має деяке значення. Воно називається *поточним* значенням. У процесі виконання алгоритму величина може не одержати конкретного значення. Така величина називається невизначеною.

У шкільному курсі математики й фізики найчастіше зустрічаються числові величини, значеннями яких є натуральні, цілі, дійсні числа. Однак в алгоритмах настільки ж часто зустрічаються й нечислові величини: слова, таблиці, списки, тексти, графіки, геометричні фігури й т.д. У курсі математики й фізики величини найчастіше позначаються однією буквою латинського або грецького алфавіту. Оскільки при застосуванні ПК і при записі алгоритмів і програм розмаїтість величин дуже велика, в алгоритмічній мові прийнято використовувати як імена величин довільні букви, буквосполучення й будь-які слова, що пояснюють зміст і призначення величини в алгоритмі.

Величини, значеннями яких є слова або текст, називаються *літерними.*

Для того щоб виділити текст, що є значенням літерної величини, значення літерних величин беруться в лапки наприклад „немає розв'язку".

Як ми бачимо, величини можуть мати різний тип. Вони можуть бути натуральними, цілими, дійсними, літерними, відповідно до того, що може бути їхнім значенням. Скорочено типи змінних позначаються словами нат (натуральний), цілий (цілий), дійсн (дійсний), літ (літерний)

Основні характеристики величин:

Ім'я (ідентифікатор)	Тип	Вигляд	Значення
Правила: - тільки латинські літери, цифри, - спершу літера, потім цифра, - не застосовувати службових слів, - не припустимі розділові	Ціл а. дійсн К. літ Н.	Змінна Конс-танта	А:=А+5 Вираз - опис правила обчислення деякої величини. Лінійна форма запису!

2. Оператор присвоювання

При програмуванні більше складних дій виникає задача обчислення чого-небудь без виведення на екран проміжних результатів. Наприклад, у виразі $(a-2b)^2 +7a – 2b$ спочатку зручно обчислити значення $a – 2b$, зберегти його, а потім, використовуючи отримане число, знайти кінцевий результат. У цьому випадку не обійтися без спеціального оператора присвоювання, записуваного за допомогою двох символів:

змінна := вираз;

Працює оператор присвоювання так — спочатку обчислюється значення арифметичного *виразу* шляхом підстановки всіх вхідних у нього змінних; результат записується в *змінну.* Ліворуч може перебувати тільки ім'я змінної, але в жодному разі не вираз. Наприклад:

a := 2 + 7;	у результаті одержимо значення a =9
c := a - 4;	c дорівнює 5
c := c + 3;	значення c збільшується на 3 і стає рівним 8
c + 1 : = 2 - a;	невірно, тому що ліворуч від знака присвоювання стоїть не змінна, а вираз!

Програмісти - початківці іноді плутають оператор присвоювання й математичний символ рівності, оскільки їхні позначення схожі один на одного. Це різні речі! Математик нас не зрозуміє, якщо ми напишемо $c = c + 3$, оскільки цей запис рівносильний неправильній тотожності 0 = 3. Двійка в щоденнику за такий вираз гарантована! Однак програміст порахує вираз виду c := c + 3 нормальним, тому що, з його погляду, це не відношення рівності, а послідовність дій, що складається з обчислення виразу в правій частині оператора присвоювання і запису отриманого результату у відповідну комірку пам'яті замість старого значення до змінної с. У даному прикладі якщо до виконання оператора с := с + 3 змінна з мала значення 5, те після його виконання вона буде мати значення 8.

змінна:=	K:=7.5; A:='слово'
змінна:= змінна	C:=B; B:=A; A:=C; (обмін значеннями)
змінна:= вираз	A:=5*B+C;

3. ЗАГОЛОВОК АЛГОРИТМУ

Запис усякого алгоритму починається із заголовка. От приклад заголовка алгоритму знаходження остачі від ділення двох натуральних чисел:

алг остача від ділення (нат *ділене,* нат *дільник,* нат *остача)*

арг *ділене, дільник*

рез *остача*

У цьому прикладі остача від ділення — назва алгоритму, *ділене, дільник,* і *остача* — імена величин. Перед кожним ім'ям величини зазначений її тип. Вихідними даними для алгоритму знаходження остачі від ділення двох натуральних чисел є величини *ділене, дільник* типу нат, а в результаті його виконання величина *остача* здобуває значення, рівне остачі ділення значень величин *ділене* й *дільник.* Величини, які є вихідними даними для алгоритму, називаються *аргументами.* Їхній список міститься після службового слова арг (аргумент). Результати алгоритму (у даному прикладі — величина *остача)* перераховуються після службового слова резульатат).

Загальний вигляд заголовка алгоритму такий:

алг назва алгоритму (список величин із вказівкою типів)

арг імена аргументів

рез імена результатів

Імена аргументів *і* результатів алгоритму перераховуються через кому.

В процесі розв'язування задачі спочатку необхідно визначитися з набіром даних, необхідних для розв'язання задачі, які називаються **вхідна інформація.** Саме розв'язання – це **обробка інформації –** тобто процес перетворення вхідної інформації у **результат.**

Приклад 6.1. Заголовок алгоритму Евклида:

алг знаходження найбільшого спільного дільника (НСД) (нат *M, N,* нат *НСД)*
арг *M, N*
рез *НСД*

III. Формування практичних умінь та навичок.

1. Яка вхідна інформація потрібна для таких задач:
а) знайти середнє арифметичне значення трьох чисел;
б) змінити знак числа на протилежний;
в) знайти цілу частину числа;
г) знайти площу та периметр прямокутника;
д) перекласти слово на англійську мову.
2. Вкажіть тип величини, якщо її значення дорівнює:
а) 25; б) 3.5; в)'число'.
3. Визначить тип даних для величин:
а) маса людини в кг; б) кількість учнів у класі; в) назва книги
4. Чому дорівнює значення Y після виконання послідовності присвоювань:
а) x:=5;y:=1; y:=x;
б) y:=5; x:=8; y:=x;
в) y:=5; x:=8; y:=y*x;
г) x:=5; y:=5*x;
д) y:=1;a:=5-y; y:=y+2*a.

IV Тестові завдання

1. Вкажіть тип величини, якщо її значення дорівнює:
а) 25; б) 36.6; в) 'слово'.
2. З наведених значень виберіть припустимі для величин: а) цілого типу; б) дійсного типу; в) літерного типу:
-5; 3.7; 38; 'три'; 20.2; '23'; 14.
3. Визначите тип для величин:
а) вага людини; б) кількість учнів у класі; в) назва дня тижня.
4. Для величини КІЛЬКІСТЬ СТОРІНОК У КНИЗІ виберіть припустиме значення:
-46; 'дев'яносто'; 175; 65,1.
5. Наведіть кілька припустимих значень для величин: а) назва породи дерева; б) висота будинку; в) вага людини.
6. Визначте, дана величина є постійною чи змінною: а) кількість днів у тижні; б) назва природного супутника Землі; в) кількість днів у січні місяці; г) кількість днів у місяці.
7. Змінні x, y, z мають відповідно тип ціл, дійсн, літ. Виберіть запис із припустимими значеннями для x, y, z (з дотриманням порядку слідування):
а) 15.3; 6; 'п'ять';
б) 38; 26.04; 'сім';
в) 'так'; 18; 10.3.
8. Наведіть приблизний припустимий інтервал значень для величин: а) кількість днів у місяці; б) швидкість автомобіля.
9. Випишіть номера команд, у записі яких допущені помилки:

x:=a+b	y:='ні'	2*a:=b+1
2B:=9	a1:=7-d	c:=c+1

10. Запишіть розпорядження у видгляі команди присвоювання:
а) збільшити значення A на одиницю;
б) V привласнити значення суми величин K і N.
11. Знайдіть значення змінної A після виконання команди присвоювання: A:=(B+3)*2; якщо початкове значення змінної B дорівнює: а) 2; б) 1.5.
12. Знайдіть значення змінної c після виконання серії команд:
a:=8; b:=a*4; c:=b-28.

V. Підсумок уроку.

Домашнє завдання:

- вивчити означення, що прочитані на лекції;
- записати алгоритм одержання числа 512 із числа 2;
- записати у вигляді схеми алгоритму послідовність виконання дій при обчисленні таких виразів:

1) $(x + y)^2$-x/y+10;
2) x-y/(a+2);
3) (a+в)/(a-в)/a*в.

Урок 5 Тема. Алгоритми з розгалуженнями

Мета: *ознайомити учнів із синтаксисом та семантикою вказівки розгалуження, повною та скороченою формами вказівки розгалуження; навчити записувати умову у вигляді відношення; перевіряти істинність умови; виконувати вказівку розгалуження; підбирати тестові значення для перевірки вказівки розгалуження; складати та виконувати алгоритми з розгалуженнями.*

Тип уроку: комбінований.

Хід уроку.

І. Актуалізація опорних знань учнів (Бесіда з елементами мотивації навчання)

- Давайте пригадаємо, які існують форми запису алгоритмів.

-Які позначення використовуються для запису алгоритму у вигляді блок-схеми?

-Як записати у вигляді блок-схеми лінійний алгоритм?

-На вашу думку, чи достатньо знати тільки лінійну алгоритмічну структуру, щоб розв'язувати різноманітні задачі?

-Складіть алгоритм переходу вулиці, де встановлено світлофор.

-Зверніть увагу, що в цьому прикладі вам треба обрати дію в залежності від кольора світлофора, тобто в залежності від виконання певної умови, виконується одна, або інша послідовність дій.

Уявім собі…

Англія, Лондон, Бейкер-Стрит, будинок знаменитого детектива Шерлока Холмса. Увійдемо з дозволу хазяїна, присядемо біля каміна й послухаємо його діалог з доктором Ватсоном в оповіданні Артура Конан-Дойля «Строката стрічка»:

«....Так що ж ви про все це думаєте, Ватсон? - запитав Шерлок Холмс, відкидаючись на спинку крісла.

- По-моєму, це найвищою мірою темна й брудна справа.

- Досить брудна й досить темна. Але *якщо* наша гостя права, стверджуючи, що підлога й стіни в кімнаті міцні, так що через двері, вікна й коминкову трубу неможливо туди проникнути, виходить, її сестра в хвилину своєї таємничої смерті була зовсім одна...

- У такому випадку, що означають ці нічні свисти й дивні слова вмираючої?

- Уявити собі не можу.

- *Якщо* зіставити факти: нічні свисти, цигани, з якими в цього старого доктора такі близькі стосунки, натяки вмираючої на якусь стрічку й, нарешті, той факт, що місс Элен Стоунер чула металевий брязкіт, що міг видавати залізний засув від ставні... *Якщо* згадати до того ж, що доктор зацікавлений у запобіганні заміжжя своєї пасербиці, — я думаю, що ми напали на вірні сліди, які допоможуть нам розгадати цю таємничу подію.

- Але тоді при чому тут цигани?

- Гадки не маю...»

Зверніть увагу на виділене курсивом слово *якщо,* що кілька разів використовує Шерлок Холмс у своїх міркуваннях про здійснений злочин. Уживання цього слова має глибокий зміст — у ситуації невизначеності воно дозволяє направити хід міркувань по одному з декількох можливих шляхів. Слово *якщо* або пари *якщо... то* є незмінними супутниками логічних міркувань. Мабуть, мистецтво вживання цих слів і становить секрет геніальності Шерлока Холмса?

ІІ. Викладання нового теоретичного матеріалу.

1. Відношення між величинами в якості умов

Як і в математиці, в алгоритмічній мові використовуються слідуючі знаки відношення між величинами:

Для числових величин

<	*Менше*	≥	*Не менше*	=	*Дорівнює*
>	*більше*	≤	*Не більше*	≠	*Не дорівнює*

Для літерних величин

= дорівнює *≠ не дорівнює*

В алгоритмах роботи з числовими та літерними величинами такі відношення між величинами і використовуються у якості простих умов.

2. Поняття розгалуження.

Розгалуження - це така форма організації дій, при якій в залежності від виконання або невиконання деякої умови здійснюється або одна, або друга послідовність дій.

алг об'їзд небезпечної ділянки
поч зменшити швидкість
якщо ремонт закінчено
то їхати 5 км по відремонтованому шосе
інакше їхати 10 км в об'їзд
все
зупинитися біля АЗС
кін

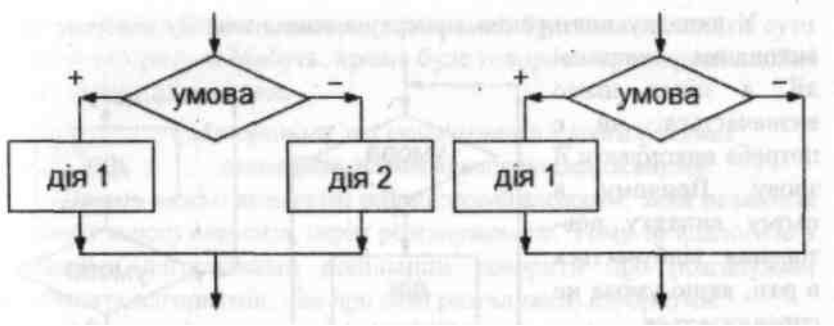
Умова - це є будь-яке твердження або запитання такого виду, що допускає лише дві можливі відповіді «так» або «ні».

Якщо відповідь на умову є позитивною, то виконується серія команд, записана після слова **то**, якщо ж відповідь негативна, - серія команд після слова інакше, а для скороченої форми - у цьому випадку нічого не виконується.

Після виконання серії команд виконавець переходить до наступної після розгалуження команди.

Якщо серія команд після слів **то** … інакше складається з більш, ніж однієї команди, то обов'язковими будуть слова **пс**, **кс** (початок та кінець серії), як-що у серії одна команда - ці слова необов'язкові. Будь-яка команда серії може знову бути командою розгалуження. У цьому випадку кажуть, що команди розгалуження вкладені одна в одну.

Повна форма розгалуження:	Скорочена форма розгалуження
Якщо < умова > то серія 1 інакше серія 2 Все	Якщо < умова > то серія все



Повна форма розгалуження:	Скорочена форма розгалуження:
Якщо на дворі холодно то одягти пальто інакше одіти куртку	Якщо на дворі дощ то взяти парасольку Все

III. Формування навичок. Розв'язування вправ

Приклад 1. *Складіть алгоритм знаходження більшого з трьох чисел.* Алгоритм має вигляд:

АЛГ Приклад 1(дійсн a,b,c,r)
АРГ a,b,c
РЕЗ r
ПОЧ
якщо a>b
то r:=a
інакше r:=b
все
якщо r<c
то r:=c
все
ДРУКУВАТИ("Більше з трьох чисел дорівнює ",r)
КІН

Пояснения до алгоритму

Спершу відбувається порівняння двох будь-яких чисел і більше число вміщується в комірку результату (при цьому використовується повна форма команди розгалуження). Потім наступне число порівнюється із вмістом комірки результату і найбільше з них вміщується в комірку результату (при цьому використовується скорочена форма команди розгалуження).

2. Побудувати алгоритм розв'язання слідуючої задачі: «Ракета запускається з точки на екваторі Землі зі швидкістю у (в км/с) за напрямом руху Землі по орбіті навкруги Сонця. Який буде результат цього запуску ракети в залежності від швидкості у?»

Алг запуск ракети **(дійсн** *в,* **літ** *А)*
Арг *у*
рез *А*
поч

якщо *y* < 7,8
то A:= „ракета впаде на Землю"
інакше якщо *y* < 11,2
то A:= «ракета стане супутником Землі»
інакше якщо *y* <16,4
то A:= «ракета стане супутником Сонця»
інакше *A* : = «ракета покине Сонячну систему»
все
все
все
кін

3. Алгоритм розв'язання нерівності *ax>b* (НЕР), де *a* й *b* — довільні дійсні числа

алг НЕР (дійсн *a, b,* з, літ *y)*
арг *a, b*
рез *c, y*
поч
якщо *a≠0*
то *c.: = b/a*
якщо *a>0*
то *y:* = „*x*> *c*"
інакше *y:* = „*x* < *c*"
все
інакше
якщо *b* < 0
то y: = „x - будь-яке число"
інакше *y:*= „розв'язків немає"
все
все
кін

Завдання 1.

Визначте, які з тверджень істинні, які хибні, які зовсім не можуть використовуватися в якості умов, а для яких істинність залежить від значень змінних:

а) 5>6;
б) x2 +y2<3;
в) наш клас дружний;
г) число x парне;
д) три точки;
є) три точки належать одній прямій; є) колір олівця; з) тротуар мокрий; ж) A дорівнює B;

Завдання 2.

Поясніть, чому неправильні такі алгоритми:

А) Якщо завтра викличуть	б) Якщо нога піднята
то вчити уроки	то опустити ногу
інакше піти на прогулянку	інакше Упадеш
Все	Все

Завдання 3.

Задано число x. Скласти алгоритм, за допомогою якого збільшіть число x на 1, якщо воно додатнє.

Завдання 4.

Задано число x. Скласти алгоритм, за допомогою якого збільшіть число x на 1, якщо воно додатнє, в іншому випадку зменшіть число x на 1.

Завдання 5.

Задано два числа x, y. Складіть алгоритм, за допомогою якого змінній z присвойте значення 1, якщо x менше y; змінній z присвойте значення нуль, якщо x = y; змінній z присвойте значення мінус одиниця, якщо x більше y.

IV. Підсумок уроку.

Домашнє завдання:

вивчити означення, що прочитані на лекції;
Записати команди розгалуження для таких дій:
а) якщо число більше ніж 2, то піднести його до кубу, інакше – до четвертого степеня;
б) якщо ціна книжки не перевищує N гривень, купити цю книжку і сувенир, інакше купити тільки книжку;

в) задано два числа А і В, від більшого відняти 1, а менше помножити на 24
Дано довжини сторін трикутника. Складіть алгоритм, який визначає, чи є цей трикутник рівнобедреним, рівностороннім, різностороннім;
Дано дісне число х. Складіть алгоритм, за допомогою якого визначіть, чи є воно коренем рівняння:
а) 5х+1=0;
б) 15х-10,8=0

Урок 6. Тема. Команда повторення

Мета: ввести поняття циклічного алгоритму, правила організації повторення дій, структуру циклу з передумовою; правила запису команди повторення «ДОКИ»; навчити учнів словесно описувати алгоритм з повторенням, складати протоколи виконання циклічних алгоритмів; застосовувати цикли при розв'язуванні задач.

Тип уроку: вивчення нового навчального матеріалу

Хід уроку.

І. Актуалізація опорних знань учнів

Диктант-лото

ІІ. Мотивація навчання

Перенесемося з Англії кінця XIX століття на дві тисячі років тому - у Давню Грецію. Давня Греція - країна великих учених, поетів і легендарних героїв. Познайомимося з історією одного з них.

«...Сізіф, син бога володаря всіх вітрів Эола, був засновником міста Коринфа, що у найдавніші часи називався Эфірою.

Ніхто у всій Греції не міг рівнятися по підступництву, хитрості й спритності розуму із Сізіфом. Сізіф завдяки своїй хитрості зібрав незлічимі багатства в себе в Коринфі; далеко поширилася слава про його скарби.

Коли прийшов до нього бог смерті похмурий Танат, щоб звести його в сумне царство Аїда, то Сізіф, ще раніше відчувши наближення бога смерті, підступно обдурив бога Таната й закував його в окови. Перестали тоді на землі вмирати люди. Ніде не відбувалося велике пишне поховання; перестали приносити й жертви богам підземного царства. Порушився на землі порядок, заведений Зевсом. Тоді громовержець Зевс послав до Сізіфа могутнього бога війни Ареса. Він звільнив Таната з оковів, а Танат вивергнув душу Сізіфа й відвів її в царство тіней померлих.

Але й отут зумів допомогти собі хитрий Сізіф. Він сказав дружині своїй, щоб вона не ховала його тіла й не приносила жертви підземним богам. Послухалася чоловіка дружина Сізіфа. Аїд і Персефона довго чекали похоронних жертв. Все немає їх! Нарешті наблизився до трону Аїда Сізіф і сказав владиці царства померлих, Аїду:

- О, володар душ померлих, великий Аїд, рівний могутністю Зевсу, відпусти мене на світлу землю. Я велю дружині моєї принести тобі багаті жертви й повернуся обернено в царство тіней.

Так обдурив Сізіф владику Аїда, і той відпустив його на землю. Сізіф не повернувся, звичайно, у царство Аїда. Він залишився в пишному палаці своєму й весело банкетував, радіючи, що один із всіх смертних зумів повернутися з похмурого царства тіней.

Розгнівався Аїд, знову послав він Таната за душею Сізіфа. З'явився Танат у палац хитрішого із смертних і застав його за розкішним банкетом. Вивергнув душу Сізіфа ненависний богам і людям бог смерті; назавжди відлетіла тепер душу Сізіфа в царство тіней..

Тяжке покарання несе Сізіф у загробному житті за всі підступництва, за всі омани, які зробив він на землі. Він засуджений вкочувати на високу, круту гору величезний камінь. Напружуючи всі сили, трудиться Сізіф. Піт градом струменіє з нього від тяжкої роботи. Всі ближче вершина; ще зусилля, і скінчена буде праця Сізіфа; але вирівається з рук його камінь і з шумом котиться долілиць, піднімаючи хмари пилу. Знову приймається Сізіф за роботу.

Так вічно котить камінь Сізіф і ніколи не може досягти мети - вершини гори...»

Нічого не скажеш - сумна історія! Адже Сізіф – взагалі - непоганий хлопець, у всякому разі в розумі йому не відмовиш! Чи не можна допомогти бідоласі?

От що цікаво - дайте прочитати давній міф програмісту, і він скаже: «Нічого страшного. Мова йде про виконання нескінченного циклу. Я й сам іноді попадаю в таке ж положення й знаходжу з нього вихід!»

Цикл є однією з найважливіших алгоритмічних структур і являє собою послідовність операторів, що повторюється неодноразово. Цикли дозволяють записати повторювані дії в компактній формі.

ІІІ. Викладання нового матеріалу.

Якщо дії P1, ..., Pn треба повторяти, поки виконується деяка умова А, то використовується вказівка:

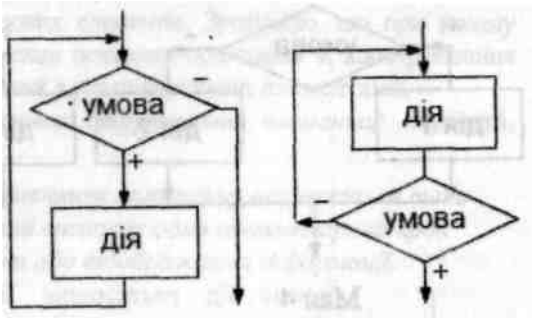
Поки A
Пц P1
…….
Pn
Кц

ЦИКЛ - це форма організації дій, при якій одна і та сама послідовність дій виконується кілька разів доти, поки виконується деяка умова.

Виконання команди приводить до того, що вказана в ній серія команд виконується декілька разів підряд. Вона виконується стільки разів, скільки потрібно для того, щоб умова перестала дотримуватися. Якщо умова не

дотримується з самого початку, то серія не виконується жодного разу. Умова циклу перевіряється перед виконанням серії, але не в процесі її виконання

Оператори повторення призначені для багаторазового виконання оператора або групи операторів, їх ще на-



зивають операторами організації циклу. Багаторазово повторювані частини обчислень називають тілом циклу, а змінні, що впливають на виконання циклу, — змінними циклу.

Алгоритм циклічної структури у загальному вигляді має містити:

1. підготовку циклу: задання початкових значень змінним циклу перед його виконанням;
2. тіло циклу: дії, повторювані в циклі для різних значень змінних циклу;
3. модифікацію (зміну) значень змінних циклу перед кожним новим його повторенням;
4. керування циклом: перевірку умови продовження (або закінчення) циклу й перехід на початок тіла циклу, якщо виконується умова продовження циклу (або вихід із циклу по його закінченні).

Для цього передбачено кілька видів команд організації циклів. Команда «**доки»** (цикл із попередньою перевіркою (передумовою)

Формат команди алгоритмічною мовою:

доки умова
пц
серія
кц

Робота команди **«Доки»**

Команда працює в такий спосіб: спочатку перевіряється умова, і якщо вона істинна, виконується зазначена послідовність команд, після чого знову перевіряється умова; якщо умова хибна, відбувається вихід з конструкції. Цикл може взагалі не виконуватися, якщо умова апріорі хибна.

Наведемо приклад:

i:=1; s:=0
доки i<10
пц s:=s+i
i:=i+1
кц

У результаті виконання цього прикладу в комірці *s* буде занесено результат обчислення: сума чисел від 1 до 9 включно. При *i* = 10 відбудеться вихід з циклу. Якщо до моменту початку роботи циклу значення *i* буде більше або дорівнюватиме 10, то цикл взагалі виконуватися не буде.

Команда «ПОВТОРЮВАТИ» (цикл із постумовою (післяперевіркою).

Формат команди алгоритмічною мовою:

Повторювати
пц
серія
кц
до умова

Робота команди **«Повторювати»**

Робота команди полягає в тому, що на початку виконується серія, і лише потім перевіряється умова. Якщо вона хибна, то знову виконується серія тощо. Робота циклу завершиться, якщо умова стане істинною. У будь-якому разі цикл спрацьовує хоча б один раз. Наприклад:

Формат команди алгоритмічною мовою:

i:=1; s:=l
повторювати
пц

s:=s+i
i:=i+1 **кц до** i=10

Цикл буде працювати доти, доки і не дорівнюватиме 10. Значення 8 при цьому дорівнюватиме 46

Команда повторення з параметром **«для».**
Формат команди:

для і від j до k крок m
пц
серія
кц

Робота команди **«для»**

де і — керуюча змінна циклу (параметр);
I — нижня межа зміни значення і;
k — верхня межа зміни значення і;
m — значення кроку. Якщо воно дорівнює 1, то його можна опускати;
***j,k,m** — У загальному випадку можуть бути виразами.*

Команда працює в такий спосіб:

1. Керуюча змінна *і (змінна циклу)* на початку набуває значення *j* і здійснюється перевірка на кінцеве значення k; якщо умова «істинна», то виконуються команди циклу.

2. Потім, щоразу до керуючої змінної і додається значення кроку m і здійснюється перевірка на кінцеве значення k; якщо умова «істинна», то виконуються команди циклу.

3. Виконання серії циклу відбувається тільки після виконання однієї з умов:

• i < k, якщо крок додатний (у цьому випадку *j* < k);
• *i* > k, якщо крок від'ємний (у цьому випадку j > k).

4. За невиконання умови відбувається вихід з циклу.

Наведемо фрагменти алгоритмів з використанням команди «**ДЛЯ».**

для і **від** 1 **до** 10 крок 1
пц
Друхувати і
кц

або

для і **від** 10 до 1 крок -1
пц
Друкувати і
кц

У результаті виконання фрагментів алгоритмів на дисплеї одержимо:
• у першому випадку 1 23456789 10;
•у другому 10 98765432 1.

Загалом, алгоритм роботи команди **для** однаковий для всіх алгоритмічних мов програмування.

Використовуючи цикл **для,** необхідно дотримуватися певних правил:

1. *Змінна, що керує циклом, не повинна змінюватися в циклі командою присвоювання (це може призвести до помилки або неочікуваного результату).*
2. *Не рекомендується виходити з циклу, не дочекавшись його завершення.*
3. *Не можна входити в цикл, оминаючи команду* **для**.
4. *У внутрішніх циклах межі зміни змінних краще обчислювати перед циклом, це заощаджує час роботи циклів.*
5. *Неприпустимо використовувати у вкладених циклах однакову змінну (це стосується й інших команд циклу).*

Команду повторення з параметром зручно використовувати, коли наперед відома кількість повторень циклу. Вона передбачає початкове присвоювання значення керуючої змінної, йоги модифікацію й перевірку умови на вхід-вихід із циклу.

Порівняно з командами **ДОКИ** або **ПОВТОРЮВАТИ,** її можливості обмежені, оскільки останні можуть мати кілька керуючих змінних циклу і кількість повторень заздалегідь може бути невідомою.

Цикл можна організувати й за допомогою команд, **ЯКЩО** й **ПЕРЕЙТИ** M, де M — мітка рядка, I яку здійснюється перехід, але з метою збереження структурності алгоритмічної мови такий варіант використовувати не рекомендується. Це вважається поганим стилем програмування.

Наведемо кілька прикладів циклічних алгоритмів.

1. Запишіть у виді команди повторення прислів'я:
а) Куй залізо, поки гаряче.
б)Скільки вовка не годуй, він у ліс дивиться.

2. Скласти алгоритм Евкліда знаходження найбільшого спільного дільника

Алгоритм знаходження найбільшого спільного дільника (НСД)

алг НСД (нат *M, N,* нат *НСД)*
арг *M, N*

рез ***НСД***
поч нат *x, y*
x:= M; в:= N поки x≠*y*
пц
якщо x > y
то x: = x — *y*
інакше *y*: = *y* — x все
КЦ
НСД: = x
кін

Задачі для самостійного розв'язання.

2. Складіть алгоритм, за допомогою якого визначте добуток:
А) перших 300 натуральних чисел;
Б) усіх двозначних парних чисел;
В) усіх двозначних непарних чисел;
Г) усіх тризначних чисел, кратних 7
3. Складіть алгоритм, за допомогою якого визначте суму:
а) усіх двозначних парних чисел;
б) усіх двозначних непарних чисел;
в) усіх тризначних непарних чисел.

Домашнє завдання:
Виконати обчислення з використанням команди повторення:
а) знайти суму перших 20 значень натурального ряду чисел;
б) знайти суму значень натурального ряду чисел від 15-го по 80-е;
в) знайти суму значень тих натуральних чисел, які кратні 7 і не перевищують 500.

IV. Підсумок уроку.

Домашнє завдання:
вивчити означення, що прочитані на лекції;
1. Складіть алгоритм, за допомогою якого можна визначити суму:
А) перших 300 натуральних чисел;
Б) квадратів перших 500 натуральних чисел.
2. Складіть алгоритм, за допомогою якого можна визначити добуток:
А) всіх парних двозначних чисел;
Б) всіх чотиризначних чисел, кратних 5.

Урок №7. Етапи розв'язування задач

Мета уроку: Дати поняття про моделювання та створення інформаційних моделей об'єктів; навчити аналізувати умову задачі, будувати математичну модель процесу (явища) за схемою ДАНО – ТРЕБА – ЗВ'ЯЗОК, визначати метод розв'язування та конструювати алгоритм розв'язування.

Тип уроку: Лекційний.

Хід уроку.

І. Актуалізація опорних знань учнів.

Усне опитування за темою попереднього уроку з метою визначення засвоєного матеріалу.

4. Розкажіть про організацію циклів:
- з передумовою;
- з післяумовою;
- за допомогою команди «повторення з параметром»
5. Що має містити алгоритм циклічної структури?
6. Порівняйте можливості команди ДЛЯ з командою ДОКИ або ПОВТОРЮВАТИ.
7. Які правила слід виконувати під час використання команди ДЛЯ?

ІІ. Викладання нового теоретичного матеріалу*:*

При проведенні різних експериментів, наукових і навчальних розрахунків, досліджень часто буває необхідно будувати якісь зменшені копії дійсних об'єктів, математичні графіки або використовувати інші засоби для наочного представлення поведінки різних об'єктів і процесів, тобто займатися моделюванням.

З давніх давен людина використовує моделювання для дослідження об'єктів, процесів та явищ в різних галузях своєї діяльності. Результати цих досліджень допомагають визначити та покращити характеристики реальних об'єктів та процесів, краще зрозуміти сутність явищ та пристосуватись до них або керувати ними, конструювати нові та модернізувати старі об'єкти. Моделювання допомагає людині приймати обґрунтовані рішення та передбачати наслідки своєї діяльності. Поняття комп'ютерного моделювання відображає використання в цьому процесі

комп'ютера, як потужного сучасного засобу обробки інформації. Завдяки комп'ютеру суттєво розширюються галузі застосування моделювання, а також забезпечується всебічний аналіз отриманих результатів.

Що ж таке модель? Під цим словом ховаються і матеріальні моделі реально існуючих об'єктів на виставці, і телевізійна красуня, що рекламує товари та сучасний одяг, і макет Ейфелевої башти, і теорія розвитку суспільства, і всім відома формула земного тяжіння $P = mgh$, і багато чого іншого. Як же в одному слові можна об'єднати такі різні поняття?

Справа в тому, що поняття моделі об'єднує дещо спільне, а саме те, що ***модель*** - це штучно створений людиною абстрактний або матеріальний об'єкт. Аналіз та спостереження моделі дозволяє пізнати сутність реально існуючого складного об'єкта, процесу чи явища, що називаються прототипами об'єкта. Таким чином, ***модель*** - це спрощене уявлення про реальний об'єкт, процес чи явище, а ***моделювання*** - побудова моделей для дослідження та вивчення об'єктів, процесів та явищ.

Може виникнути питання, чому не можна дослідити сам оригінал, навіщо створювати моделі? Для цього може існувати багато причин:

- оригінал може не існувати в часі (гіпотеза про загиблий материк Атлантида, про побудову Єгипетських пірамід, про можливу "ядерну зиму", що може початися після атомного бомбардування);
- реально цей об'єкт не можна побачити (модель земної кулі, сонячної системи або атома);
- людина хоче побачити об'єкт, але не має можливості потрапити на місце його знаходження (модель Ейфелевої башти, єгипетської піраміди, Софіївського собору тощо);
- процес, який досліджує вчений, небезпечний для життя (ядерна реакція).

Таких прикладів можна придумати багато. І ви, якщо подумаєте, можете згадати багато моделей, що бачили у своєму житті.

Згадайте, які моделі ви бачили у своєму житті (сама тривіальна відповідь - іграшки). Навіть, коли ви зранку складаєте план своїх дій на день, це теж можна вважати моделюванням.

Для одного і того ж об'єкта можна створити велику кількість моделей. Все залежить, по-перше, від мети, що ви поставили перед собою, а по-друге, від методів та засобів, за допомогою яких ви збираєте інформацію про прототип. Наприклад, якщо ви хочете познайомитись з новим містом, то карта цього міста, фотографії, розповіді мешканців або кіноальманах дадуть вам зовсім різні уявлення про об'єкт, причому ці уявлення можуть зовсім не співпасти з тими враженнями, що ви отримаєте потім після відвідування цього міста безпосередньо. Модель цього ж самого міста для його мешканців взагалі буде іншою, тому що для них головне - це забезпечення нормальної життєдіяльності. Як ви вже переконались, кількість моделей та їх різноманітність дуже велика. Щоб не загубитися в цьому розмаїтті, необхідно мати якусь класифікацію моделей. Розглянемо найбільш суттєві ознаки, за якими класифікуються моделі:

- галузі використання;
- урахування в моделі фактора часу;
- спосіб представлення моделей.

Розглядаючи моделі з позиції галузі використання, можна сказати, що вони бувають:

- *навчальні* - наочні посібники, тренажери, навчальні програми;
- *дослідні* - створюються для дослідження характеристик реального об'єкта (модель теплоходу перевіряється на усталеність, а модель літака - на аеродинамічні характеристики);
- *науково-технічні* - для дослідження процесів та явищ (ядерний реактор або синхрофазотрон);
- *ігрові* моделі - для вивчення можливої поведінки об'єкта в запрограмованих або непередбачених ситуаціях (військові, економічні, спортивні ігри тощо);
- *імітаційні* моделі - виконується імітація дійсної ситуації, що багато повторюється для вивчення реальних обставин (випробування лікарських препаратів на мишах або інших тваринах, політ собаки в космос).

З урахуванням фактора часу моделі можуть бути динамічні та статичні. В першому випадку над об'єктом виконуються дослідження на протязі деякого терміну, а в другому - робиться одноразовий зріз стану (наприклад, постійний нагляд сімейного лікаря та одноразове обстеження в поліклініці).

За способом представлення моделі можуть бути матеріальні та інформаційні. Матеріальні моделі - це предметне відображення об'єкта зі збереженням геометричних та фізичних властивостей. Наприклад, іграшки, чучела тварин, географічні карти, глобус і таке інше - це матеріальні моделі реально існуючих об'єктів. Матеріальною моделлю можна також назвати хімічний або фізичний дослід. Ці моделі реалізують матеріальний підхід до вивчення об'єкта чи явища.

Інформаційна модель - це сукупність інформації, що характеризує властивості та стан об'єкта, процесу чи явища, а також взаємодію із зовнішнім світом.

Інформаційні моделі можуть бути:

- *вербальними* - моделі отримані в результаті розумової діяльності людини і представлені в розумовій або словесній формі;
- *знаковими* - моделі, що виражені спеціальними знаками (малюнками, текстами, схемами, графіками, формулами тощо).

За формою представлення можна виділити наступні види інформаційних моделей:

- *геометричні* - графічні форми та об'ємні конструкції;
- *словесні* - усні та письмові описи з використанням ілюстрацій;

- *математичні* - математичні формули, що відображають зв'язок різних параметрів об'єкта;
- *структурні* - схеми, графіки, таблиці;
- *логічні* - моделі, в яких представлені різні варіанти вибору дій на основі різних умовиводів та аналізу умов;
- *спеціальні* - ноти, хімічні формули і таке інше;
- *комп'ютерні* та *некомп'ютерні*.

В сучасному світі розв'язування складних наукових та виробничих задач неможливе без використання моделей та моделювання. Серед різних видів моделей особливе місце займають математичні моделі, тому що вони дозволяють враховувати кількісні та просторові параметри явищ та використовувати точні математичні методи. Вивчення реальних явищ за допомогою математичних моделей, як правило, вимагає застосування обчислювальних методів. При цьому широко використовуються методи прикладної математики, математичної статистики та інформатики.

Підсумуємо вищевикладене.

Моделювання - це особлива форма експерименту, яка полягає в тому, що досліджується не сам об'єкт, а деяка його заміна.

Форми моделювання дуже різноманітні і залежать як від самого об'єкта, так і від мети його вивчення.

Інформаційна модель - це матеріальний або ідеальний об'єкт, що використовується замість оригіналу явища (процесу) при його дослідженні, при цьому зберігається інформація про деякі важливі для даного дослідження типові риси і властивості оригіналу, тобто його істотні сторони.

Модель необхідна для того, щоб:

- зрозуміти, як влаштований об'єкт: які його структура, основні властивості, закони розвитку і взаємодії з навколишнім світом;
- навчитися керувати об'єктом або процесом і визначати найкращі способи керування при заданій меті і критеріях (оптимізація);
- прогнозувати прямі чи непрямі наслідки впливу на об'єкт.

Інформаційні моделі можуть формулюватися на будь-яких мовах: російській, англійській й ін. В них можуть бути використані мова графічних побудов, мова хімії, біології і т.д.

Окремим типом інформаційних моделей є математична модель.

Математична модель - це заміна оригіналу явища (процесу) відповідним аналогом за допомогою математичних залежностей.

Рішення практичної задачі починається з опису вихідних даних і мети задачі. Точне формулювання умов і мети рішення - це математична постановка задачі, а математичний опис найбільш істотних властивостей реального об'єкта - це математична модель.

Комп'ютерна модель - математична модель, яка реалізована за допомогою певних програмно-апаратних засобів.

Існують задачі, які важко або неможливо розв'язати без застосування комп'ютерів. Це різного роду задачі моделювання, наприклад, моделювання фізичних процесів, біологічні, екологічні, економічні задачі моделювання, задачі оптимізації й інші.

Етапи розв'язання прикладних задач з використанням комп'ютерів

Етап	**Опис етапу**
Математична постановка задачі	1. Визначити умови задачі: - Що дано? - Які дані допустимі? - Які результати, в якому вигляді повинні бути отримані?
Побудова математичної моделі, вибір методу	1. Розгорнутий змістовний опис задачі замінити її математичною моделлю за допомогою математичних залежностей. 2. Обгрунтовано вибрати метод
Складання алгоритму на основі обраного методу	Алгоритм у більшій мірі визначається обраним методом, хоча той самий метод може бути реалізований за допомогою різних алгоритмів. При складанні алгоритму слід враховувати всі його властивості.
Складання програми	Програмування (складання програми) - кодування алгоритму на одній з мов програмування
Тестування і налагодження програми	Перевірка правильності роботи програми за допомогою тестів і виправлення виявлених помилок. Тест - це спеціально підібрані вихідні дані і результати, отримані

Аналіз результатів	Після остаточного виконання програми зробити аналіз результатів. Можлива зміна самого підходу до рішення задачі і повернення до першого етапу для повторного ви-

Математична модель - це перелік вхідних даних, результатів, які потрібно отримати, та зв'язок між величинами, виражений у вигляді математичних співвідношень.

Для побудови математичної моделі використовується така форма:

Дано: <Перелік початкових даних>

Потрібно: <Перелік погрібних результатів>

Зв'язок: <Система рівнянь або тверджень, що зв'язують вхідні та шукані дані>

При <Умови допустимості початкових даних>

Приклади математичної моделі

Задача: Розв'язати лінійне рівняння типу ax+b=c

Дано: a, b, c Потрібно: x

Зв'язок: якщо a=0 і c – b = 0, то x - будь-яке число;

якщо a=0 і c – b $\neq$0, то коренів немає;

якщо a $\neq$ 0 і c – b = 0, то x =0;

якщо a $\neq$ 0 і c – b $\neq$ 0, то x=(c – b)/a.

III. Формування практичних навичок.

Побудувати математичну модель та алгоритм розв'язування задач:

1. У цариці Несміяни кругле обличчя, радіус якого дорівнює R. Визначте довжину сторони такого квадратного дзеркала, щоб, коли Несміяна милується собою, її відображення поміщалося в дзеркалі.

2. З тераріуму втекли x гадюк, у кобр та z гюрз. Довжина кожної гадюки 1м, кожної кобри 1м 30см, а гюрзи 1м 15см. Скільки повних метрів отруйних змій втекло з тераріуму? Яку довжину вони складають у сантиметрах?

IV. Підсумок уроку.

Домашнє завдання:

- вивчити означення, що прочитані на лекції (що таке модель та моделювання, класифікація моделей, комп'ютерне моделювання);
- придумати приклади моделей, що зустрічаються нам у повсякденному житті;
- придумати набір елементів конструктора, з якого потім можна зібрати яку-небудь іграшку;
- Побудувати математичну модель та алгоритм розв'язування задач:

а) знайти корені квадратного рівняння типу $ax^2+Ь=0$;

б) знайти корені квадратного рівняння типу $ax^2+bx+c=0$;

в) прямокутник, довжини сторін якого відповідають умові a/b=b/(a-b), називається золотим. Визначити, чи є даний прямокутник золотим.

Завдання до тематичної атестації з теми «Основи алгоритмізації»

1. Що таке алгоритм?
2. Що таке змінні в алгоритмах?
3. Що таке константи в алгоритмах?
4. Які основні властивості алгоритмів?
5. Які існують основні форми подання алгоритмів?
6. Хто може бути виконавцем алгоритму?
7. Чим викликано існування багатьох способів опису алгоритмів?
8. Назвіть та опишіть базові структури алгоритмів.
9. Як записується і використовується структура "слідування"?
10. Як записується і використовується структура "розгалуження"?
11. Як записується і використовується структура "повторення" (цикл-ДОКИ)?
12. Як записується і використовується структура "повторення" (цикл-ДО)?
13. Для чого необхідна алгоритмічна мова?
14. З чого складається алгоритмічна мова?
15. З чого складається алфавіт алгоритмічної мови?
16. Які правила опису алгоритмів НАМ?
17. Яка структура алгоритму, записаного на НАМ?
18. Які основні вказівки використовуються в НАМ?
19. Як записується і виконується вказівка розгалуження?
20. Як записується і виконується вказівка повторення?
21. Що таке величина?
22. Які існують типи величин?
23. Які існують види величин?
24. Що називається аргументом алгоритму?
25. Що називається результатом алгоритму?
26. Як записується і виконується вказівка присвоювання?

Завдання для самостійного виконання

1. Записати в словесній формі алгоритми:

а) Знаходження остачі під ділення числа a на число b,

б) Розклад числа N на прості множники,

в) Перевірки ознаки подільності даного числа a на 3,

г) Знайти найбільший спільний дільник двох натуральних чисел a і b.

2. Перерахувати всі службові слова, які використовуються:

а) у вказівці розгалуження

б) у вказівці повторення

3. Скласти алгоритм обчислення Y за формулою Y=(5x – 2)(x + 4).

5. Скласти алгоритм обчислення Y за формулою $Y = \begin{cases} x+2, & якщо\ x <= 5, \\ 100, & якщо\ x > 5. \end{cases}$

6. Скласти алгоритм знаходження найбільшого з трьох величин a,b,c.

7. Скласти алгоритм пофарбування підлоги, яка зроблена з 30 дощок.

2. Основні поняття мови Паскаль

Урок 9. Мова програмування Turbo Pascal 7.0

Мета уроку: Дати дитині поняття про програму, класифікацію мов програмування, поняття системи програмування, поняття про інтерпретацію та компіляцію. Дати поняття про основні резики роботи з файлами в інтерактивному середовищі ТП 7.0; основні відомості про роботу у вбудованому редакторі ТП 7.0; поняття про компіляцію та виконання програм; порядок діагностування помилок. Навчити користуватися командами меню та «гарячими клавішами» для роботи з файлами, роботи з вікнами, отримання довідки, виконання програм.

Тип уроку: Лекційний.

Хід уроку.

І. Організаційний момент.

ІІ. Викладання нового теоретичного матеріалу

1. Мови програмування. Паскаль.

Першими мовами програмування були FORTRAN, COBOL, ALGOL і деякі інші. У кожної з них були свої позитиви й свої недоліки. Одною з найбільш удалих довгий час уважалася ALGOL, настільки вдалою, що цю мову стали використовувати в спеціальній літературі для запису алгоритмів. Але й вона не була позбавлена недоліків, зокрема, остання версія ALGOL'а була зайво громіздкою, тому швейцарський професор наприкінці 60-х років XX ст. Никлаус Вірт зайнявся розробкою своєї власної мови, що успадкувала би від ALGOL'а краще, але була би більше лаконічною і мала би більше чітку логічну структуру. Призначалася нова мова для навчання студентів, і сам Вірт спочатку ставився до неї як до іграшки. Мова була названа на честь французького філософа й винахідника механічного калькулятора Блеза Паскаля - Паскалем. Нова мова виявилася настільки вдалою, що швидко завоювала популярність.

Поява нової мови збіглася з початком ери персональних комп'ютерів, які стали доступними майже кожній звичайній людині (це відбулося не відразу, і були часи, коли вартість «персоналки» була порівнянна з вартістю легкового автомобіля!). Масла у вогонь підлила фірма Borland, яка у першій половині 80-х випустила пакет Турбо Паскаль, що містив не тільки транслятор, але й редактор, а також інші програми, які значно полегшували процес програмування. Це зробило Турбо Паскаль популярнішою системою програмування.

Турбо Паскаль. Чому не просто Паскаль, а *Тур*бо Паскаль? Слово «Турбо» в англійському лексиконі означає прискорення. Транслятор, що входить до складу Турбо Паскаля, дуже швидко переводить програму з мови програмування в машинні коди, помітно швидше, ніж транслятори в інших системах програмування. От тому - Турбо.

Скажемо ще, що Турбо Паскаль - це не окрема мова програмування, а «розширення» звичайного, стандартного Паскаля, що включає інтегроване середовище програмування. Слова «інтегроване середовище» означають, що з однієї програми є доступ до редактора текстів, транслятора, довідкової системи, налагоджувача та ін. До складу Турбо Паскаля входять додаткові набори процедур, які дозволяють не займатися щораз программуванням деяких складних дій, таких, наприклад, як виведення графіки. Крім Турбо Паскаля є й інші системи програмування на Паскалі, але ми будемо використовувати саме Турбо Паскаль.

2. Поняття про мови прогамування.

Процес роботи комп'ютера полягає у виконанні програм, тобто деякого набору команд, що надходять у визначеному порядку. Машинний код команди складається з нулів та одиниць та указує, яку саме дію треба виконати центральному процесору. Отже, щоб задати комп'ютеру послідовність дій, яку він має виконати, треба задати послідовність двійкових кодів відповідних команд. Писати такі програми дуже складна справа. Раніше для цього програміст повинен був пам'ятати не тільки всі комбінації нулів та одиниць двійкового коду кожної команди, але й двійкові коди адрес даних, що використовувалися під час виконання програми. Щоб полегшити роботу програмістів, було розроблено багато мов програмування, які в більш наочному для людини вигляді подавали послідовність дій комп'ютера.

Алгоритмічні мови, призначені для побудови описів алгоритмів, що виконуються електронними обчислювальними машинами, називаються ***мовами програмування***.

Описи алгоритмів мовою програмування називають ***програмами***.

Набагато легше написати програму мовою, що наближена до людської, а перекладання цієї програми на машинні коди доручити комп'ютеру. Так з'явилися мови, що призначені спеціально для написання програм - мови програмування.

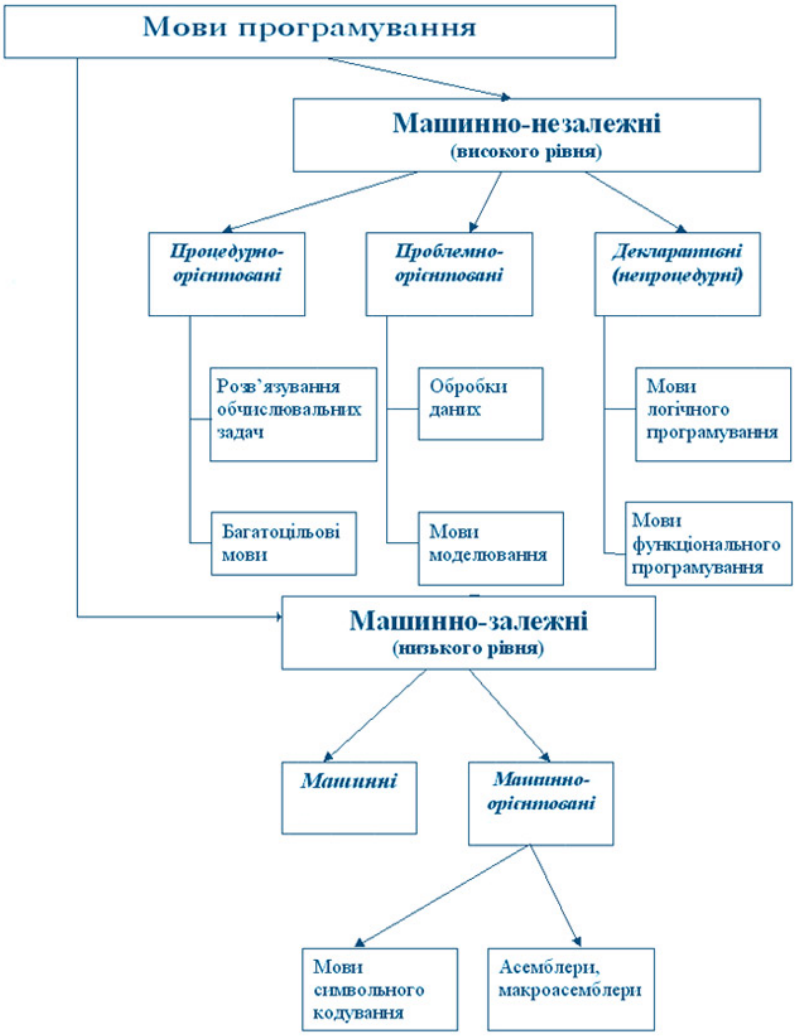
Існує багато різних мов програмування. Взагалі, для розв'язування більшості задач можна використовувати будь-яку з них. Тільки досвідчені програмісти знають, яку мову програмування краще використовувати для розв'язування складних спеціалізованих задач, щоб урахувати особливості тієї чи іншої з них.

Всі існуючи мови програмування можна поділити на дві групи:

- мови низького рівня;
- мови високого рівня.

До мов низького рівня належать мови ***асемблера*** (від англ. to assemble - складати, компонувати). У мові асемблера використовуються символьні позначення команд, які легко зрозуміти і запам'ятати. Замість послідовностей двійкових кодів команд записуються їх символьні позначення, а замість двійкових адрес даних, які

використовуються під час виконання програми, - символьні імена цих даних. Іноді мову асемблера називають ***мнемокодом або автокодом***.



Більшість програмістів при складанні програм користуються деякою мовою високого рівня. Для описування алгоритмів такою мовою використовується певний набір символів - алфавіт мови. З цих символів складаються так звані ***службові слова*** мови, кожне з яких має певне призначення. Службові слова зв'язуються одне з одним в речення за певними синтаксичними правилами мови і визначають деяку послідовність дій, які мусить виконати комп'ютер.

Використання мов високого рівня надає можливість описувати програми для комп'ютера, використовуючи загальноприйняті позначення операцій і функцій.

Та програми, що написані на мовах програмування високого рівня (алгоритмічних мовах програмування), комп'ютер "не розуміє". Для того, щоб він міг виконати програму, її потрібно перекласти на машинну мову. Для такого перекладу використовують спеціальні програми, що мають назву - транслятори.

Транслятор - це програма, що призначена для перекладу тексту програми з однієї мови програмування на іншу. Процес перекладання називається ***трансляцією***.

Будь-який транслятор виконує дві основні задачі.

Перша - аналіз програми, що транслюється, в результаті чого визначається її коректність. При виявленні помилок транслятор вказує на ті місця тексту програми, де порушені правила її написання.

Друга - генерація вихідної програми мовою команд комп'ютера.

Розрізняють два типи трансляторів:

- компілятори
- інтерпретатори.

Компілятор - це програма, призначена для перекладу в машинні коди програми, що написана мовою високого рівня. Процес такого перекладу називається компіляцією.

Кінцевим результатом роботи компілятора є програма в машинних кодах, яка потім виконується ПК. Скомпільований варіант програми можна зберігати на диску. Для повторного виконання програми компілятор вже не потрібен. Досить завантажити з диска в пам'ять комп'ютера скомпільований перед цим варіант і виконати його.

Існує інший спосіб поєднання процесів трансляції та виконання програм. Він називається **інтерпретацією**.

Інтерпретатор - це програма, що призначена для трансляції та виконання вихідної програми по командах (на відміну від транслятора, який цей процес виконує в цілому).

У процесі трансляції відбувається перевірка програми на відповідність до правил її написання. Якщо в програмі знайдені помилки, транслятор виводить повідомлення про них на екран монітора. Інтерпретатор повідомляє про знайдені помилки після трансляції кожної команди програми, а компілятор - після завершення компіляції всієї програми. Знайти та виправити в цьому випадку помилки значно складніше, ніж при інтерпретації. Через це програми-інтерпретатори розраховані, в основному, на мови, що призначені для навчання програмуванню, і використовуються програмістами-початківцями.

Як правило, програми компілятори та інтерпретатори називаються так само, як і мови, для перекладу з яких вони призначені. Слова Паскаль, Бейсік, Сі можна сприймати і як назви мов, і як назви відповідних програм - трансляторів.

3. Опрацювання середовища ТП 7.0

1. Увійдіть в каталог системи програмування ТП 7.0.
2. Виконайте команду turbo.exe.

F10 - вихід в головне меню.

Esc - вийти з головного меню та повернутися у вікно редактора текстів.

Дія	**Доступ через меню**	Гаряча клавіша
Створити новий файл	File/New	
Відкрити файл	File/Open	F3
Зберегти файл	File/Save	F2
Виконати програму	Run/Run	Ctrl+F9
Переглянути екран	Debug/User Screen	Alt+F3
Створити exe-файл	Compile/Compile	Alt+F9
Викликати довідкову систему	Help	F1
Отримати довідку про вказаний курсором		Ctrl+F1
Перейти до іншого вікна	Window/List	Alt +0
Закрити поточне вікно	Window/Close	Alt+F3
Вийти з середовища	File/Exit	Alt+X

Практичне завдання

Відкрийте існуючий в каталозі системи програмування файл Primerl.pas. В програмі Primerl використані основні елементи процесу опрацювання даних у відповідності з принципом IPO (input - processing - output), тобто введення - опрацювання - виведення.

Для повного розуміння цього принципу необхідно познайомитися з операторами введення-виведення та присвоювання.

```
Program Primerl; {додавання двох цілих чисел}
var a,b,s: integer; begin
write (' введіть перше число'); readln(a);
write ( введіть друге число'); readln(b);
s=a+b;
writeln ( 'сума  A+B=' ,s)
end.
```

1. Виконайте програму декілька разів для різних A і B.
2. Створіть exe-файл. Для цього виконайте послідовність дій:
   - увійдіть в головне меню, виберіть меню Compile.
   - виберіть і виконайте команду Destination - Memory, яка після натискання клавіші Enter змінюється на Destination - Disk.
   - Натисніть комбінацію клавіш Alt+F9.
3. Вийдіть з середовища ТП 7.0. Виконайте файл Primerl.exe.

Підсумок уроку.

Домашнє завдання:

вивчити означення, що прочитані на лекції (що таке програма, класифікація мов програмування, що таке транслятор, типи трансляторів).

Урок 10. Основні поняття мови Turbo Pascal 7.0

Мета: дати поняття про загальну характеристику мови Паскаль; алфавіт мови, основні поняття мови: ідентифікатор, величина, оператор, вираз; правила лінійного запису арифметичних виразів. Навчити записувати математичні вирази в лінійній формі, обчислювати поточне значення величини та остаточний результат.

Тип уроку: вивчення нового навчального матеріалу.

Хід уроку.

І. Актуалізація опорних знань учнів.

Запитання для перевірки знань учнів.

1. Для чого створено мови програмівання?
2. Що можна назвати програмою?
3. Для чого програми перекладають в машинні коди?
4. Чим компілятор відрізняється від інтерпретатора?
5. Які мови програмування ви знаєте? Наведіть приклади.

ІІ. Викладання нового навчального матеріалу.

Мова програмування — це один із способів подачі опису алгоритму, що розрахований на виконавця — комп'ютер.

Будь-яка мова програмування характеризується трьома основними складовими: алфавітом, синтаксисом і семантикою. Сукупність символів, які дозволяється використовувати під час побудови опису алгоритмів мовою програмування, називають **алфавітом** цієї мови.

Сукупність правил опису алгоритмів мовою програмування називають **синтаксисом** мови програмування.

Правила **семантики** пояснюють, яке смислове значення має кожний елемент мови. У будь-якій мові програмування можна виділити чотири типи елементів, що використовуються при побудові програм: символи, слова, вирази, команди (оператори). **Символи мови** — це основні знаки, якими описуються команди і дані.

Слова мови — структури, що утворені із символів алфавіту мови програмування і мають певний зміст. Слова — це імена змінних та констант, числа, службові слова та ін.

Вираз — це правило (формула) обчислення значення однієї величини. Якщо одержуване значення числове, то вираз називають арифметичним, якщо значення логічне, то вираз називають логічним або бульовим, якщо значення—текст, то вираз називають літерним.

Команда — це вказівка на виконання деякої дії. При написанні програм команди називають операторами, а величини, що використані команді — операндами.

Скінченна послідовність виконуваних почергово команд називається **серією команд**. Серія може складатися з однієї команди і навіть бути порожньою.

Алфавіт і словник мови

Програма мовою Паскаль формується за допомогою набору знаків, що утворюють алфавіт мови, і складається з літер, десяткових і шістнадцяткових цифр і спеціальних символів.

У якості літер використовують великі та малі літери латинського алфавіту: ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklm nopqrstuvwxyz, а також _ (знак підкреслення).

У якості десяткових цифр: 1234567890. Шістнадцяткові цифри утворюються з десяткових цифр і літер від A до F (або від a до f).

При написанні програм застосовуються спеціальні символи:

+ плюс	- мінус
* зірочка (знак множення)	/ знак ділення
= дорівнює	> більше
< менше	# міжнародний комерційний номер
$ знак грошової одиниці	[] квадратні дужки
() круглі дужки	{} фігурні дужки
. крапка	, кома
: двокрапка	; крапка з комою
' апостроф	~ тильда
@ ет (або комерційне а)	пробіл.

Комбінації спеціальних символів можуть утворювати складені символи, які сприймаються комп'ютером як один символ:

:= присвоювання	<> не дорівнює
… діапазон значень	>= більше або дорівнює
(..) альтернатива [ ]	<= менше або дорівнює.
(* *) альтернатива {}	

Неподільні послідовності символів утворюють слова, що несуть певний зміст у програмі. Слова відділяються розділовими символами, такими як: пробіл, кома, символ кінця рядка, коментар. Слова поділяються на: стандартні, зарезервовані, ідентифікатори користувача. *Зарезервовані* слова є складовою частиною мови, мають фіксоване написання і назавжди визначений зміст. Наприклад: ***begin, else, function, write, end, program*** та ін.

Стандартні слова призначені для заздалегідь визначених розробником мови типів даних, констант, процедур і функцій (наприклад, ***sin, cos, Pi***). Зарезервований ідентифікатор можна перевизначити, але це може призвести до помилки, тому краще цього не робити.

Ідентифікатори користувача використовують для позначення констант, змінних, процедур і функцій, що визначені самим програмістом. Існують загальні правила написання ідентифікаторів:

1. Ідентифікатор починається тільки з літери або знака підкреслення.
2. Ідентифікатор може складатися з літер, цифр і знака підкреслення.
3. Між двома ідентифікаторами має бути хоча б один розділовий знак.
4. Максимальна довжина ідентифікатора 127 символів, але значущими є тільки перші 63 символи.
5. Ідентифікатор не може бути зарезервованим словом.

У написанні програм можна використовувати як великі, так і малі літери. Компілятор не визначає різниці між ними.

Правила оформлення програм (пунктуації):

1. Крапку з комою можна не ставити після begin і перед end, тому що ці слова є операторними дужками, а не операторами.
2. Крапка з комою розділяє оператори. її відсутність між операторами викликає помилку компіляції. Наявність між операторами кількох крапок з комою не є помилкою, тому що компілятор сприймає їх як ознаку наявності порожніх операторів.
3. При використанні вкладених структур може виникнути ситуація:

```
                end;
        end;
    end
```

Крапку з комою можна ставити як після кожного, так і після останнього end. А наприкінці програми можна ставити крапку.

4. В операторах циклу крапка з комою не ставиться після while, repeat, do і перед until.
5. В умовних операторах крапка з комою не ставять після then і перед else.

III. Закріплення теоретичного матеріалу.

Опитування за питаннями для перевірки засвоєння знань

1. Що називають алфавітом мови програмування?
2. Що називають серією команд?
3. Чи можна змінити зміст зарезервованого слова?
4. Для чого використовують ідентифікатори користувача? .
5. Чи може бути ідентифікатор користувача зарезервованим словом?

Підсумок уроку.

Домашнє завдання:

вивчити означення, що прочитані на лекції.

Урок 11. Величини. Прості типи мови Паскаль.

Мета уроку: ознайомлення з поняттями величини, типів даних, форму вання навичок визначення та опису стандартних типів даних; виховання інформаційної культури.

Тип уроку. Вивченя нового навчального матеріалу.

Хід уроку.

I. Актуалізація опорних знань учнів.

- Сформулюйте загальні правила написання ідентифікаторів.
- Сформулюйте правила оформлення програм.

II. Викладання нового матеріалу.

1. Поняття величини.

У своїй роботі програміст має справу з таким поняттям, як величина.

Що ж таке величина? З точки зору програмування *величини* — це дані, що обробляються програмами. Мова Паскаль інтерпретує дані, як *а. константи* або *змінні.* Як перші, так і другі визначаються ідентифікаторами (іменами), за допомогою яких можна звертатися для одержання *х* відповідних значень. *Константами* називають такі дані, яким присвоюються значення в описовій частині програми, й у процесі виконання програми їх змінювати заборонено. Для визначення констант служить зарезервоване слово *const.*

Формат опису:

Const < ідентифікатор > = < значення константи >;

Приклад: **Const Max=1000;**

Vxod='сегмент 5';

Є константи, до значень яких можна звертатися без попереднього опису.

Ідентифікатор	Тип	Значення	Опис
True	Boolean	True	Істина
False	Boolean	False	Хибність
Maxint	integer	32767	Максимальне

Змінні, на відміну від констант, можуть змінювати свої значення в процесі виконання програми. Кожна змінна і константа належать до визначеного типу даних. Тип констант визначається компілятором автоматично. Тип змінних обов'язково вказується перед тим, як їх використати. Для опису змінних призначено зарезервоване слово *var.*

Формат опису:

Var <ідентифікатор> : <тип даних>;

Приклад: Var Suml, Sum2 : real;

2. Типи даних.

Тип даних — це сукупність властивостей певного набору даних, від яких залежать: діапазон значень, якого можуть набувати ці дані, а також сукупність операцій, які можна виконувати над цими даними.

Усі типи даних у мові програмування Паскаль розділяють на дві групи: скалярні (прості), структуровані (складені). *Скалярні* типи, у свою чергу розділяють на *стандартні* та *типи користувача.* Стандартні типи користувачам пропонують розробники системи Turbo Pascal. Типи користувача — розроблюють програмісти самостійно.

До стандартних скалярних типів належать типи: цілі, дійсні, літерні, булівські. Величини *цілих* типів можна подавати як у десятковій, так і в шістнадцятковій системах числення. Якщо число подане в шістнадцят-ковій системі, перед ним без пробілу записується знак *$.* Діапазон зміни шістнадцяткових чисел від $0000 до *SFFFF.*

Цілі типи даних являють собою значення, що можна використовувати в арифметичних виразах.

Стандартні цілі типи наведено в таблиці:

Тип	Діапазон	Необхідна пам'ять (байт)
Byte	0...255	1
Shortint	-128... 127	1
Integer	-32768 ...32767	2
Word	0... 65535	2
Longint	-2147483648 ...2147483647	4

Дійсні типи даних являють собою дійсні значення, що використовуються в арифметичних виразах і займають у пам'яті від 4 до 10 байт. У програмі мовою Паскаль можна подавати дані дійсних типів у вигляді як із плаваючою, так і з фіксованою крапкою.

Дійсні *десяткові числа з фіксованою крапкою* записуються за звичайними правилами арифметики. Зверніть увагу, що ціла частина від дробової відокремлюється десятковою крапкою, а не комою. Якщо десяткова крапка відсутня, число вважається цілим. Перед числом може знаходитися знак «+» або «-». Якщо знак відсутній, то число вважається додатнім.

Дійсні *десяткові числа у форматі з плаваючою крапкою* подаються в експоненціальному вигляді: *mE+p,* де *m*-мантиса (ціле або дробове число з фіксованою десятковою крапкою), *E* - означає «десять у степені», *p* -порядок (ціле число). Мантиса має бути нормалізованою, тобто поданою у вигляді числа з діапазону від 0 до 9 (це означає, що крапка завжди знаходиться після першої значущої цифри числа). Однак можна записати мантису у вигляді будь-якого дробового числа з фіксованою крапкою. Нормалізація при цьому виконується системою автоматично. Наприклад:

Число у форматі з плаваючою	Значення числа
0.4500E+02	$0.45*10^2=45$
-2.600E05	$-2.6*10^5 = -260000$
+0.45670E-02	$0.4567*10^{-2}= 0.004567$

Стандартний (найчастіше використовуваний) дійсний тип даних наведений у таблиці:

Тип	Діапазон значень	Мантиса (кількість	Необхідна пам'ять
Real	±2.9*10E-39 .. 1.7*10E38	11-12	6

Літерний (символьний) тип може набувати значень ASCII-таблиці комп'ютера. Символьній змінній в пам'яті виділяється один байт, тому можна зберегти тільки один символ ASCII-таблиці.

Булівський тип подається двома значеннями: ***True*** *(істина)* або ***False*** *(хибність).* Цей тип застосовують у логічних виразах і виразах відношення.

Примітка: *Для розв'язування майже всіх задач шкільного курсу інформатики учням цілком достатньо використання стандартного дійсного типу даних, але якщо вчитель вважає потрібним, то він може дати учням інші чотири дійсних типа* ***(single, double, extended, comp).***

Структуровані типи у своїй основі мають один або кілька скалярних типів даних. До структурованих типів даних належать рядки, масиви, файли, записи і т.д. їх ми розглянемо пізніше.

Перетворення типів:

Ціле значення можна перетворити в дійсне, присвоївши дійсній змінній ціле значення: A:=3+5.

Дійсне значення можна переворити в ціле за допомогою стандартних функцій: TRUNC(X) – ціла частина аргументу; ROUND(X) – аргумент, округлений до найближчого цілого (за правилами математики); INT(X) – аргумент, округлений до найближчого меншого цілого.

Сумісність типів:

При виконанні оператора присвоювання обчислюється вираз, що стоїть в правій частині, і його значення присвоюється змінній в лівій частині. При цьому тип виразу повинен відповідати типу змінної. Для стандартних типів це означає, що вони повинні збігатися. Допускається присвоювання змінній дійсного типу значення цілого типу. Присвоювання ж змінній цілого типу виразу дійсного типу заборонено!

Змінні і константи всіх типів використовуються у **виразах**

Вираз задає порядок виконання дій над даними і складається з операндов (констант, змінних, звертань до функцій), круглих дужок і знаків операцій. Круглі дужки ставлять, як і в математиці, для керування порядком виконання операцій. Якщо дужки відсутні, то операції виконуються залежно від їх пріоритетів, про що буде сказано далі.

У мові Паскаль є такі операції: арифметичні; відношення (порівняння); логічні. Операції можуть бути *унарними* та *бінарними.* У першому випадку операція стосується одного операнду і завжди записується перед ним, у другому операція виражає відношення між двома операндами і записується між ними. ***Арифметичні операції*** задають арифметичні дії у виразах над значеннями операндів цілих та дійсних типів. Найчастіше використовуються арифметичні операції, що подані в наступній таблиці:

Операція	Дія	Тип операндів	Тип результату
Бінарні			
+	Додавання	Цілий	Цілий
		Дійсний	Дійсний
	Віднімання	Цілий	Цілий
		Дійсний	Дійсний
*****	Множення	Цілий	Цілий
		Дійсний	Дійсний
/	Ділення	Цілий	Дійсний
		Дійсний	Дійсний
Div	Ділення націло	Цілий	Цілий
Mod	Залишок від ділення	Цілий	Цілий
Унарні			
+	Збереження знака	Цілий	Цілий
		Дійсний	Дійсний
-	Інверсія знака	Цілий	Цілий
		Дійсний	Дійсний

Операції відношення виконують порівняння двох операндів і визначають значення виразу є істинним або хибним. Результат завжди має булівський тип одного з двох значень: ***True*** *(істина)* або ***False*** *(хибність).*

Операція	Назва	Вираз	Результат
=	Дорівнює	A=B	True, якщо A дорівнює B
<>	Не дорівнює	A<>B	True, якщо A не дорівнює B
>	Більше	A>B	True, якщо A більше B
<	Менше	A<B	True, якщо A менше B
>=	Більше або	A>=B	True, якщо A більше або дорівнює B
<=	Менше або	A<=B	True, якщо A менше або дорівнює B

Результатом виконання *логічного (булівського)* ***виразу*** є логічне значення ***True*** або *False.*

Виконання кожної операції відбувається з урахуванням її пріоритету. Пріоритети операцій зазначені в наступній таблиці:

Операція	Приоритет	Вид операції
Not, унарні «-»і «+»	перший (вищий)	Унарна операція
*, /, div, mod, and	другий	Операції типу
+, -, or	третій	Операції типу
=, <>, <, >, <=, >=	четвертий (нижчий)	Операції відношення

Арифметичні вирази у якості операндів можуть містити імена функцій. Про поняття *функції* мова буде йти пізніше в курсі програмування, але стандартні функції *(cos, sin, x^2* та інші) знайомі учням зі шкільного курсу математики, і використання таких виразів у курсі інформатики відрізняється тільки правилами запису (синтаксису). Так, наприклад, у програмуванні аргумент функції обов'язково береться в круглі дужки.

3. Команда присвоювання

Команда присвоювання має вигляд

<ім'я змінної> := <вираз>;

Дія команди. Обчислюється вираз і його значення надається змінній. Вираз призначений для описування формул, за якими виконуються обчислення. Вираз може містити числа, змінні, сталі, назви функцій,з'єднані символами операцій.

Змінна і вираз мають бути одного типу або узгодженими: змінним дійсного типу можна надавати значення виразів цілого типу, а змінним рядкового типу присвоювати значення виразів символьного типу, але не навпаки.

Приклад. Розглянемо дію команд присвоєння в програмі Trykutnyk: p:=a+b+c; p:=p/2; s:=sqrt(p*(p-a)*(p-b)*(p-c)). Тут обчислюється значення периметра і воно надається змінній p, півпериметра (надається також змінній p) та площі (надається змінній s).

Основні стандартні функції та процедури

Функція	Тип аргументу	Тип результату	Математичний запис, коментар
abs(x)	integer, real	integer, real	$\|x\|$
arctan(x)	integer, real	real	arctgx
cos(x)	integer, real	real	cosx
sin(x)	integer, real	real	sinx
exp(x)	integer, real	real	ex
ln(x)	integer, real	real	lnx
sqrt(x)	integer, real	real	$\sqrt{x}$
sqr(x)	integer, real	integer, real	X2
ord(x)	char упорядкова-ний	integer	ASCII-код симв., номер елемента
succ(x)	упорядкований	упорядкований	повертає наступне значення x
pred(x)	упорядкований	упорядкований	Повертає попереднє значення x
round(x)	real	integer	заокруглює число x до цілого
trunc(x)	real	integer	відкидає дробову частину числа x
int(x)	real	real	відкидає дробову частину числа x
frac(x)	real	real	дробова частина чис-
odd(x)	integer	boolean	true (x - непарне), false (x - парне)
random (x)	integer	integer	Генерує випадкове число з діапазону від 0
upcase(x)	char	char	замінює малу літеру латинської абетки на велику
процедури:			
inc(x.y)	integer	integer	збільшує x на y
inc(x)	integer, char	integer, char	Збільшує x на 1
dec(x.y)	integer	integer	Зменшує x на y
dec(x)	integer, char	integer, char	Зменшує x на 1

Розглянемо приклади значень функцій і виконання процедур:

round(2.1)=2, int(2.1)=2.0, x:=l; inc(x,5); (x=6),
round(6.8)=7, int(6.8)=6.0, x:=V; inc(x); (x='b'),
trunc(2.1)=2, frac(2.1)=0.1, x:=7; dec(x,3); (x=4),
trunc(6.8)=6, frac(6.8)=0.8, x:='d'; dec(x); (x=V).

$$x^{\alpha} = e^{\alpha \ln x}, \quad \log_b a = \frac{\ln a}{\ln b}$$

Це відповідає такому запису в Pascal:

exp (ln (x)* α).

Правила лінійного запису арифметичних виразів:

1. вираз повинен бути записаний у вигляді лінійного ланцюжка символів, що обумовлено конструкцією пристрою введення інформації;
2. не можна опускати знак операції множення;
3. порядок виконання операцій одного пріоритету регулюється дужками;
4. аргументи функцій записуються в круглих дужках.

Перестановка змінних. Добре іллюструє роботу із змінними наступне завдання - поміняти місцями значення двох змінних a й b (тобто, щоб в a з'явилося те, що було раніше в b, і навпаки). Здавалося б, можна записати так:

a := b; b := a;

однак тут криється помилка. Давайте перевіримо це на конкретному прикладі. Нехай a дорівнювало 10, а значення b дорівнювало 7. Після виконання оператора a := b в b залишиться 7; a дорівнюватиме 7, і тепер, якщо виконати оператор присвоювання b := a, там також виявиться 7. Ми втратили одне зі значень! Як же бути? Вихід полягає у використанні додаткової проміжної змінної t, в якій ми збережемо значення змінної a до того, як його буде треба замінити новим:

t := a; a := b; b := t;

Корисно перевірити виконання даного фрагменте програми на конкретному прикладі:

	a	b	t
Значення до	10	7	-
Після t:=a	10	7	10
Після a:=b	7	7	10
Після b:=t	7	10	10

III. Формування практичних вмінь та навичок.

Виконайте завдання

1. Скільки арифметичних операцій містить вираз:

1)(x + 1/2)*(y + 7/10) -3/4;
2) (x + y)/(x - y)/x*y;
3) (x + y) mod (x - y) div 10;
4) sqr (x mod y div y)/x - y;
5) trunc (sin ((x - y)/x*y) div 10?

2. Записати арифметичний вираз у вигляді звичайної алгебраїчної формули:

1) a*b*c - a/(n*m)/(a + k/2);
2) a*b/(c + d) + (c- d)/b*(a + 6);
3) (a/(j + k/(2*x)) + c)/(j - sqr (sqrt (a)));
4) x/(j + sqr (x)/(2 + x*sqr (x)/3));
5) (x*sin (y + z*(2 + k)) + 1)/(0.7 - x/(y + sqr (z)))*x/(a + b).

3. Вказати порядок виконання операцій під час обчислення значення простого арифметичного виразу:

1)sqr(x) + sqr(sqr(i/));
2) x*sqr (y)/sqr (2);
3) a*b/c*d;
4) sqrt (a) + 1 - sqr (b);
5) -x*sqr (y)/sqr (sqr (2)).

4. Нехай A= 5, B = 4, C = 3 та P = 0.5. Обчислити значення простого арифметичного виразу:

1) (A + B)/C*P; 2) (A + B)/C/P;
3) (A + B)/(C*P); 4) -A*B + sqr (C) + 0.5.

5. Перевести формулу у вигляд , доступний для програмування:

1) $a+bc$; 2) $x+\frac{2x-10}{y+2}$; 3) $10^5\alpha(\beta-10^{-3})$;

4) $\frac{x-x_0}{x_1-x_0}-\frac{y-y_0}{y_1-y_0}$; 5) $\frac{a}{x+\frac{b}{2+\frac{c}{5+d}}}$.

6. Нехай задані такі значення дійсних змінних x = 0.5 та h = 0.1. Які значення матимуть ці змінні після виконання таких операторів присвоювання:

1) x:=2.5; 2) x := x + 2*h; h:=h/2;
3) x:=x + h; 4) h := - h; x := x + h; h := h*21

7. Є три цілочислові змінні з поточними значеннями A = 3, B = 5, C = 7. Які значення матимуть ці змінні в результаті виконання таких послідовностей операторів присвоювання:

1) C :=A*B + 2; B := B + 1; A := C - sqr (B);
2) P := C; C := B; B := A; A := P;
3) B:=B+A; C:=C + B;
4) A :=A + 1;B:=A + B;C:=A + B?

8. Задані дійсні змінні x та y з поточними значеннями x = 0.8, y = - 2.1 та цілочислові змінні k та j із поточними значеннями k = 5, j = - 3. Знайти значення, якого набуде відповідна змінна в результаті виконання кожного з наведених нижче операторів присвоювання із зазначенням його типу:

1) k:=x; 2) k:=(k + 1)/10;
3) j:= 5 +y; 4) x :=(k + 1)/10;

5) k:=k/3; 6) j:=x + y;
7) x:=k; 8) x:=x + *y;*
9) y:=k + x; 10) j:=x + 0.9.
9. Чому дорівнюватимуть значення змінних x та y після виконання таких дій:
x:=2, y:=5, x:=y; y := x?

IV. Підсумок уроку

Домашнє завдання:

вивчити означення, що прочитані на лекції.

1. Задати за допомогою оператора присвоювання такі дії:
1) змінній z присвоїти значення, що дорівнює півсумі значень змінних x та y;
2) подвоїти значення змінної a;
3) значення змінної x збільшити на 0.1;
4) за нове значення змінної *a* прийняти її поточне значення, піднесене до квадрата;
5) змінити знак у значенні змінної *q*.

2. Змінній *a* присвоїти значення, що дорівнює сумі значень змінних x та *y*, а змінній *b* - подвоєному їх добутку.
3. Визначити, за допомогою якої послідовності дій можна поміняти місцями вміст змінних x та *y*:
1) x := x + y; *2)x:=y;* 3) *y* := x + *y;* *4)* x := x + *y;*
y:=y-x; x:=y - x; x := y - x; *y* := x - *y;*
x:=*y;* *y*:=x-*y;* *y* := y - x; x:=x-*y*.
4. Записати у вигляді оператора присвоювання обчислення виразу $y = 2x^3 + 3x^2 + x + 5$, не використовуючи при цьому операції піднесення до степеня.
5. Яке значення будуть мати змінні x та y після виконання операторів:
А) x:= 178 y:= x Div 10 e:= x Mod 10;
Б) x:= 123 y:= 1234 x:= (x Div 100)Mod 10 y:= (y Mod 1000) Div 100;
В) x:= 3456 y:=12345 x:=(x Div 100)Mod 10 y:=(y Mod 1000)Div 10;
Г) x:= 23456 y:= 12345 x:= (x Mod 10000) Div 10 y:= (x Mod 100) Div 10.
6. Обчислити значення виразів
а)trunc(6,9); б) round(6,9); в)tranc(6,2); г)round(6,2); д) trunc(-l,8); є) round(-l,8); ж) round(0,5); з)round(-0,5); і)20 div 6; к)123 div 0;
л) 3.0mod 3; м)20 mod 6; н) 2 div 5; о) 2 mod 5; п) 3 * 7div2 mod 7 / 3 - trunc(sin(1)); р) succ(round(5/2)-pred(3)).

Урок 12. Самостійна робота №1

Мета: перевірити знання учнів з тем «Основні поняття мови Паскаль» та «Величини. Прості типи мови Паскаль».

1. Визначити тип результату виразів, якщо I, j, k: integer; x, y, z: real;
1) i*i+j*j+2*k/3; 2) cos(x)+3*sin(y)-z; 3) i+sqrt(j)/
2. Обчислити значення виразів, якщо A=2.5; B=7.8; C=-17.3; M=5; X=8.7
1) (A+B)/C*M; 2) 2+X*X/(X+(A+B)/5); 3) (A<B) AND (X+A<B) OR (C<M)
3. Змінній A присвоїти дробову частину додатнього числа X.
4. Обчислити значення виразів:
а) trunc(7.8); б) round(7.8); в) trunc(7.2); г) round(7.2); д) trunc(-2.7); е) round(-2.7); ж) trunc(0.5); з) round(-0.5).
5. Обчислити значення виразів:
а) 30 div 6; б) 30 mod 6; в) 30 div 5; г) 30 mod 5; д) 3 div 7; е) 3 mod 7; ж) 137 div 0; з) 3.0 mod 6.
6. Вказати порядок виконання операцій у виразі:
-a mod b+a div b*c
7. Обчислити значення виразів:
а) 3*7 div 2 mod 7/3-trunc(sin(1)); б) succ(round(5/2)-pred(3)).
8. Визначити тип (цілий чи дійсний) виразу:
а) 1+0.0; б) 20/4; в) sqr(4); г) sqr(5.0); д) sqrt(25); е) sin(0); ж) succ(-3); з) trunc(-3.14).
9. Вкажіть тип констант:
а) 5; б) -4.0; в) $3.6\cdot10^6$; г) ‘567’; д) ‘ok’.
10. Запишіть наступні формули в лінійній формі запису:

1) $\frac{ab}{c}+\frac{b}{ac}+\frac{a}{bc}$; 2) $10^6 d+\frac{3}{7}$; 3) $\sqrt{x_1^2+5}$; 4) $\sqrt{|x+6|}$.

Урок 13. Створення лінійних програм

Мета: дати поняття про загальну структуру програми, призначення розділів опису; синтаксис та семантику операторів введення/виведення; правила форматування при виведенні; навчити *правильно описувати величини; організовувати введення та виведення даних; програмувати простіший діалог.*

Тип уроку: вивчення нового навчального матеріалу.
Хід уроку.
І. Організаційний момент.
ІІ. Викладання нового матеріалу.
1. Структура програми. Програма складається із заголовка
program <ім'я програми>;
розділів описової частини

uses label	— приєднання бібліотек та модулів; — оголошення міток (позначок);
const	— оголошення сталих;
type	— опис типів;
var	— оголошення змінних;
procedure function	— оголошення процедур користувача; — оголошення функцій користувача

та виконуваної частини
begin
<розділ команд>
end.

Заголовок та усі розділи, окрім останнього, є необов'язковими. Розділювачем між конструкціями (командами) програми є символ ";". У кінці програми завжди має стояти крапка.

Заголовок програмі надає програміст. В *іменах,* які користувач дає своїм програмам та змінним, великі і малі букви рівноправні: імена A та a (або MyName та myname) позначають один і той самий об'єкт.

У програму можуть входити коментарі. ***Коментар*** — фрагмент тексту програми, взятий в фігурні дужки або записаний так: (* коментар *). Коментар слугує для пояснення роботи програми і не впливає на виконання команд. Він може бути розташований у довільному місці програми.

2. Розділи оголошення сталих і змінних. Усі величини, які входять у програму, повинні бути описані у розділі сталих (констант), якщо вони не мінятимуть значення протягом виконання Програми:

const <стала 1> = значення 1>;

або у розділі оголошення змінних, якщо вони обчислюватимуться:

var <список змінних 1> : <тип змінних 1>;
<список змінних n > : <тип змінних n>;

Елементи списків записують через кому. Кутові дужки <...> — це засіб формалізованого описування конструкцій мови. У конкретних програмах їх не використовують.

3. Перша програма. Програма — це послідовність команд, за допомогою яких записують алгоритм розв'язування задачі. Програми (алгоритми) складають за таким принципом: ***вводять*** дані, ***визначають*** потрібне, ***виводять*** результати. Аналогічно розв'язують задачі з математики та фізики, але тут обчислення вручну виконувати не потрібно — їх виконає комп'ютер.

Розглянемо програму з назвою Trykutnyk для розв'язування задачі обчислення периметра *p* та площі s трикутника зі сторонами a = 5, b=3.6, c=4.2 за формулою Герона. Усі команди, наведені в програмі, будуть детально розглянуті нижче.

```
program Trykutnyk;
uses Crt;                          {Приєднуємо модуль Crt}
const a=5; b=3.6; c=4.2;           {Вводимо довжини сторін}
var p,s: real;                     {Оголошуємо змінні для]
begin                              {периметра та площі}
   clrscr;                         {Очищуємо екран}
   p:=a+b+c;                       {Обчислюємо периметр}
   writeln('p=', p:5:2);           {Виводимо значення периметра}
   p:=p/2;                         {Обчислюємо півпериметр}
   s:=sqrt(p*(p-a)*(p-b)*(p-c));   {Визначаємо площу}
   writeln('s=',   s:5:2);         {Виводимо значення площі}
writeln('Виконав Іванов П.');
readln
end.
```

Після виконання програми на екрані отримаємо:
p= 12.80 s= 7.43

Прості команди алгоритмів: присвоювання, введення і виведення.

4. Команди введення (read, readln) даних. Надавати значення змінним можна двома способами: за допомогою команди присвоювання, наприклад x:=5, або команд уведення даних з клавіатури. Другий спосіб робить програму більш універсальною, оскільки дає змогу розв'язувати задачі для різних значень змінних. Команда read має вигляд

Read(<змінна 1>,…<змінна n>);

Дія команди. Виконання програми зупиняється. Система переходить у режим очікування введення даних (екран темний, миготить курсор). Значення цих даних користувач набирає на клавіатурі через пропуск або натискає після кожного даного на клавішу вводу. У результаті виконання цієї команди відповідним змінним будуть присвоєні конкретні значення.

Команда readln має вигляд

readln(<змінна 1>,...,<змінна n>);

Вона діє як команда read з тою різницею, що зайві дані у рядку введення ігноруються. Наступна команда вводу читатиме дані нового рядка. Цю команду застосовують під час роботи з текстовими файлами.

Зауваження. Команду readln без параметрів часто використовують у середовищі TP для MS-DOS, щоб оглянути результати виконання програми на екрані. Щоб після цього перейти у режим редагування програми, потрібно натиснути на клавішу вводу. *Зауваження.* Значення змінних логічного й перерахованого типу вводити з клавіатури *не можна.*

5. Команди виведення (write, writeln) даних. Для виведення на екран повідомлень та результатів обчислень використовують команди write та writeln:

write(<вираз 1>,< вираз 2>..... <вираз n>);

У списку виведення можуть бути сталі, змінні або вирази.

Дія команди. Сталі, значення змінних та виразів виводяться на екран у вікно виведення, яке можна переглянути за допомогою комбінації клавіш Alt+F5. Команда

writeln(<вираз 1>,...,<вираз n>);

діє майже так само як і команда write; різниця така: наступна після неї команда write чи writeln буде виводити значення на екран у новому рядку.

Для переходу на новий рядок екрана чи для пропуску рядка використовують команду writeln без параметрів.

6. Форматний вивід. Команди write та writeln можуть здійснювати форматний вивід даних. Форматування — це подання результатів у наперед заданому користувачем вигляді. Для цього після виразу через двокрапку записують число (:n) — кількість позицій на екрані, які треба надати для виведення значення цього виразу. Формат :n застосовують для даних *цілого та рядкового* типів. Під час виведення даного *дійсного* типу зазначають загальну кількість позицій для всіх символів (n) та кількість позицій для дробової частини (m),тобто формат має вигляд :n:m.

Задача 1. Дано координати трьох вершин трикутника A(1;1), B(2;2) та C(-1;2). Обчислити медіану m_b та радіус описаного кола *r.*

```
program TrykutnykNew;
uses Crt;
var xl,yl,x2,y2,x3,y3,a,b,c,mb,r,x,y,p,s: real;
begin
      clrscr;
      writeln('введіть координати:');
      readln(xl,yl,x2,y2,x3,y3);
      a:=sqrt(sqr(x3-x2)+sqr(y3-y2));  {Обчислимо довжини}
      b:=sqrt(sqr(xl-x3)+sqr(yl-y3));  {сторін трикутника}
      c:=sqrt(sqr(xl-x2)+sqr(yl-y2));
      x:=(xl+x3)/2;                    {Обчислимо координати)
      y:=(yl+y3)/2;                    {середини сторони b)
      mb:=sqrt(sqr(x-x2)+sqr(y-y2));   {Обчислимо медіану mb}
      p:=(a+b+c)/2;                    {Обчислимо півпериметр}
      s:=sqrt(p*(p-a)*(p-b)*(p-c));    {Обчислимо площу}
      r:=a*b*c/(4*s);                  {Обчислимо радіус}
      writeln('mb=',mb:5:2);           {Виведемо результати}
      writeln('r=',r:5:2);             {Виведемо радіус}
      readln
end.
```

Задача 2. Написати програму визначення суми цифр тризначного числа.

```
program proba_l;
var x, rez: integer; begin
writeln('Задайте тризначне число:');
```

```
readin(x);
rez:=(x mod 10)+((xdiv 10) mod 10)+(x div 100);
writeln(' Сума цифр заданого числа: ', rez); readln;
end.
```

7. Зворотний запис числа. Дано тризначне число *x = abc (a, b, c* - його цифри). Потрібно одержати число, записане тими ж цифрами, але у зворотному порядку. Тобто, якщо дано число 128, то одержати треба 821. Перед нами встає завдання знаходження цифр числа — *a, b* і *c.* Розмістити ці цифри у зворотному порядку легко — результат буде дорівнювати *x* = 100 c + 10 *b* + *a.* Найпростіше знайти цифру одиниць с — вона буде дорівнювати залишку від ділення числа х на 10 (наприклад, якщо *x* = 128, те *x* mod 10 буде 8, що нам і потрібно). Отже,

c := x mod 10;

Щоб знайти *b,* спочатку знайдемо *ab,* рівне *x,* діленому націло на 10 (для 128 це буде 12), а потім уже визначимо *y* — цифру одиниць числа, що вийшло після останніх дій :

b := x div 10 mod 10;

Знайти *a* просто — це результат ділення *x* націло на 100:

a :.= x div 100;

Тепер можна написати всю програму:

```
program naoborot;
var x, a, b, з : Integer;
begin
Write('x==>');
ReadLn(x);
c := x mod 10;
b := x div 10 mod 10;
a := x div 100;
Write(100 * з + 10 * b + a) ;
end.
```

III. Формування практичних вмінь та навичок.

Задача 1. Скласти програму обчислення:

1) Середнього арифметичного чисел A, B, C;
2) Площі прямокутного трикутника з катетами A і B;
3) Суми цифр тризначного числа A.

Задача 2. Обчислити добуток трьох дійсних чисел.

Задача 3. Обчислити значення виразу $\frac{a^2 - 2ab - 3}{2}$

де *a* й *b* — дійсні числа.

Задача 4. Дано чотиризначне число *x = abcd.* Одержати число, записане тими ж цифрами у зворотному порядку *(y = dcba).*

Задача 5. Дано тризначне число *x = abc.* Знайти суму квадратів його цифр.

IV. Підсумок уроку.

Домашнє завдання.

Підготувати відповіді на запитання:

1. Яку структуру повинна мати правильно написана програма на мові програмування Паскаль?
2. Які блоки в програмі обов'язкові, а які ні?
3. Що таке лінійна програма?
4. Які ви знаєте вказівки введення та виведення інформації?

Скласти програму, задавши вхідні дані самостійно.

1. Радіус Місяця 1740км. Обчислити площу поверхні $S=4\pi r^2$ та об'єм планети $V=(4/3)\pi r^3$.
2. Обчислити кінетичну $E=mv^2/2$ та потенціальну $P=mgh$ енергії тіла заданої маси *m*, яке рухається на висоті *h* зі швидкістю *v*.
3. Скільки секунд мають доба, тиждень, рік?
4. Задано двозначне число. Визначити суму цифр цього числа.

Урок 14. Розв'язування задач на створення лінійних програм

Мета: формування практичних навичок складання програм з лінійними алгоритмами; виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

І. Актуалізація опорних знань учнів.

Повторення понять величини, типів величин, команди присвоювання, правил оформлення виразів та пріоритетів виконання операцій мовою Паскаль.

Розв'язання прикладів на повторення:

Приклад 1. Якщо значення цілочисельних змінних таке: *X*= 15, *Y*= 25, Z= 8, то чому дорівнюватимуть значення цих змінних після виконання операцій присвоювання:

X := **sqrt(Y)*2;**
Y := **Y-2;**
Z := **Y div 2 mod 3;**

Очікувана відповідь: **X:= sqrt(25)*2** —тут присвоєння не може бути виконаним, оскільки значення квадратного кореня — дійсне число, а *X*—ціле число

Y:= **25-2** Y = 23
Z:= 23 **div** 2 **mod** 3 Z = 2

Приклад 2. Написати команду присвоєння, що надає значення середнього арифметичного змінних *X та* У змінній Z (Тип змінних дійсний).

Очікувана відповідь: **Z** : = **(X + У)** / **2**

Приклад 3. За допомогою яких операторів присвоєння можна поміняти місцями значення двох змінних *X та Y.*

Очікувана відповідь: Для цього необхідне використання третьої змінної того ж типу, що й змінні X та У, наприклад, Z: Z := X; X :=Y; Y := Z

II. Формування навичок.

Розв'язування задач

Задача 1.

Якщо на одну шальку терезів посадити Даринку, яка важить N кг, і Наталку, яка важить на 5кг менше, а на іншу насипати М кг цукерок, то скільки кілограмів цукерок доведеться з'їсти дівчаткам, щоб шальки терезів зрівноважилися?

Введемо змінні для зберігання результатів: *N*— вага Даринки; *M* — вага цукерок; *P*— вага цукерок, які необхідно з'їсти дівчаткам.

Тоді програма для розв'язання задачі буде такою:

```
Program Task_1;
Uses  crt;
Var M, N, P : real;
Begin
Clrscr;
Write('Введіть вагу Даринки'); Readln(N);
Write('Введіть вагу цукерок, що лежать на терезах'); Readln(M);
P :=N+N-5-M; {N-5- вага Наталки)
Writeln("Дівчаткам необхідно з'їсти ' ,P, 'кг цукерок.');
 Readln;{Процедура затримує зображення на екрані до натискання клавіші Enter}
End.
```

Задача 2

Визначити, яку платню одержить на фірмі сумісник за виконану роботу, якщо йому нараховано S грн., а податок становить 20%.

Необхідні змінні: *S* — сума нарахувань сумісника; *P*—реальна платня, яку він одержить у касі (за умовою вона становить 80 % від нарахувань).

Програма має наступний вигляд:

```
Program Task_2;
Uses crt;
Var  P,S  :  real;
Begin
Clrscr;
Write ('Введіть суму нарахувань робітника') ; Readln(S);
P  := S*0.8;
Writeln('Платня сумісника становить:',  P:8:2);
Readkey;
End.
```

Задача 3. Від міста A до B автомобіль їхав t_1 = 5 год з середньою швидкістю v_1 = 70 км/год, від B до C — t_2 = 4 год. зі швидкістю v_2 = 75 км/год. Визначити відстань між містами.

```
program Distance;
var t1, vl, t2, v2, ab, be, ac : integer;
begin
tl := 5 ; t2 := 4;  vl := 70 ; v2 := 75;
ab := vl * tl; be := v2 * t2; ac :== ab +bc;
writeln (ab:6, bc:6, ac:6);
readln
```

```
end.
```

Виконаємо програму і на екрані отримаємо: **350 300** 650.

Завдання. Модифікуйте програму на випадок чотирьох міст.

Задача 4. Автотранспортна фірма «Радар» купує п'ять (kl) мікроавтобусів «Пежо» по 32100 грн. (c1) за автобус і три (к2) мікроавтобуси «Івеко» по 29500 грн. (c2). Яку суму (suma) потрібно заплатити?

```
program Radar;
var kl, cl, k2, c2, suma : integer;
begin
kl := 5 ; k2 := 3;  cl := 32100 ; c2 := 29500;
suma := kl * cl + k2 * c2;
writeln (suma:8);
readln
end.
```

Виконаємо програму, і замість результату отримаємо повідомлення про помилку. Виявляється, що значення змінної suma є 249 000, і воно вийшло за допустимі межі, визначені для типу **integer.** Ось чому правильно оголосити змінні потрібно так:

var kl, k2, cl, c2 : integer; suma : longint; .

Отже, щоб розв'язати задачу, потрібна додаткова пам'ять, оскільки змінні типу **longint** займають удвічі більше пам'яті, ніж змінні типу integer.

Завдання. Виконайте програму Radar і поекспериментуйте з різними цінами. Модифікуйте програму, якщо купують три марки автобусів.

Задача 5. Ввести з клавіатури будь-яке тризначне число. Визначити суму його цифр і вивести цифри числа у зворотному порядку.

Нехай змінна a міститиме значення заданого числа. Цифри числа позначимо так: i — кількість сотень, j — кількість десятків, k — кількість одиниць, а їхню суму — s. Для визначення цифр деякого числа використовують операції **div** та **mod.**

```
program MyNumber; var a, i, j, k, s : integer;
begin
write('Введіть число a: ');read (a);
i := a div 100;    {Отримаємо кількість сотень}
j := a div  10 mod 10; {Отримаємо к-сть десятків}
k := a mod 10;  {Отримаємо кількість одиниць}
s := i + j + k;
writeln('Сума цифр числа a =  ', s);
writeln(k,j,i)
 end.
```

Виконаємо програму. Введемо число 235 - отримаємо результат суми цифр числа a = **10**

Вправи та задачі для самостійного розв'язання

1. Швидкість світла 299792 км/с. Яку відстань долає світло за хвилину, годину, добу?
2. Квіткова клумба має форму круга. Обчисліть її периметр і площу за заданим радіусом.
3. Тіло падає з прискоренням g. визначіть пройдений тілом шлях $h=gt^2/2$ після першої та другої секунди падіння.
4. Яка маса золотої кульки заданого радіуса, наприклад 0.02м, якщо густина золота 19300кг/м^3? ($M=\rho \cdot V$, $V=(4/3)\pi r^3$)
5. Яка маса зливка срібла у вигляді куба з заданою стороною, наприклад 6см, якщо густина срібла 10500кг/м^3?

III. Підсумок уроку.

Домашнє завдання. Виконайте вправи та задачі.

1. Нехай a=0. Якого значення набуде змінна a, якщо команду a:= a+2 виконати: а) один раз; б) два рази підряд; в) три рази підряд.
2. Нехай a=1. Якого значення набуде змінна a, якщо команду a:= a*2 виконати: а) один раз; б) два рази підряд; в) три рази підряд.
3. Які значення матимуть змінні (та якого вони є типу) після виконання команд присвоювання, якщо раніше були виконані команди A:=2; B:=5; C:=0:

а) A1 := (2*A - 3*B)/(11-2*B);
б) A2 := A/2 + B+5*A/(B+C);
в) A3 := 3 + 25 div 4 + 7 mod 3;
г) A4 := 110 div 100 +110 mod 100 + 110 mod 10;
д) A5 := B div A + B/A + B mod A (відповідь: 5.5)?

4. Уведіть тризначне ціле число. Визначте суму і добуток крайніх цифр.
5. Уведіть чотиризначне число. Виведіть його цифри а) у стовпчик; б) у зворотному порядку.
6. Уведіть п'ятизначне ціле число. Обчисліть суму його цифр.

7. Задано координати вершин трикутника у площині: $(x_1 ; y_1), (x_2; y_2), (x_3; y_3)$. Обчисліть довжини його сторін (складіть програму).

Довідка. Відстань між двома точками, які задані координатами $(x_1 ; y_1)$ та $(x_2; y_2)$, визначають так:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} .$$

8. Задано координати вершин чотирикутника у площині: $(x_1 ; y_1), (x_2; y_2), (x_3; y_3), (x_4; y_4)$. Складіть програму для обчислення довжини його сторін.

Урок 15-16. Практична робота №1 «Створення лінійних програм»

Мета: формування практичних навичок складання програм з лінійними алгоритмами; виховання інформаційної культури.

Тип уроку: практична робота

Хід уроку.

I. Повторення правил поведінки в комп'ютерному класі.

II. Актуалізація опорних знань учнів.

Дати відповіді на запитання:

1. Яку структуру повинна мати програма на мові Паскаль?
2. Які блоки в програмі обов'язкові, а які ні?
3. Що таке лінійна програма?
4. Які оператори використовують для введення/виведення інформації?

III. Виконання практичної роботи.

1. Яка маса Землі? Сучасні наукові дослідження супроводжуються обчисленнями, які не під силу виконати вручну одній людині чи навіть багатьом людям. Однак комп'ютер потрібний не лише для обчислень. Сьогодні він дає змогу відкрити нові явища в природі, уточнити наші знання про вже відомі фізичні факти тощо.

Розглянемо задачу визначення маси нашої планети: скласти алгоритм і програму для обчислення маси Землі (M) у кілограмах, а також її густини (p). У скільки разів *(n)* густина Землі більша, ніж густина води (густина води — 1000 *кг/м^3)?*

Розв'язування. У курсі фізики є дві формули для сил F_1 і F_2, які діють на тіло масою *m,* яке перебуває на Північному полюсі Землі:

$$F_1 = \gamma \frac{mM}{R^2}, \quad F_2 = mg$$

Такі сталі були визначені експериментально з деякою точністю:

γ — гравітаційна стала 6,67*10^{-11};

R — радіус Землі 6,37*10^6 *м;*

g — 9,8 *м/с^2.*

Згідно із законами фізики рівняння $F_1 = F_2$ є математичною моделлю задачі. З нього визначаємо *M,* записуємо формулу для об'єму Землі та її густини p і шуканого числа *n* — отримаємо метод розв'язування і алгоритм (як послідовність формул):

M= gR^2 /γ;π

$V = (4/3)\pi R^3$ {Це формула об'єму кулі}; ρ **= *M/V;*** *n* = ρ/1000. Використаємо латинські імена gamma та ro замість відповідних грецьких літер і складемо програму.

```
program MyPlanet;
const g = 9.8; gamma - 6.67E-11; r = 6.37E+6;
var M, V, ro, n : real;
begin
M : = g * r * r /  gamma; V := 4/3 * pi * r * r * r; ro := M / V;
n := ro /1000;
writeln('Маса =', M, ' густина =', ro);
writeln('густина Землі більша, ніж води в ',n:4:1, "разів') end.
```

Виконайте програму, отримайте результати і перепишіть їх у зошит. Яка маса Землі?

2.Складіть програми для обчислення значень виразів:

а) $5x^5 - 6,3x^2(\sin\pi/7 - tg3a) + 8,12$;

б) $\cos l/3\arcsin 2a + 4,5x^3 - 2,6x + 2,3$.

3. Складіть графічну схему алгоритму і програму для обчислення кінетичної $E=mv^2/2$ і потенціальної енергії тіла $P=mgh$ маси *m,* що рухається на висоті *h* зі швидкістю *v.* Вхідні дані задайте самостійно.

4. Складіть графічну схему алгоритму і програму для обчислення площі основи $S=\pi R^2$ і об'єму $V=1/3Sh$ конуса за відомими радіусом основи *r* і висотою *h.* Вхідні дані задайте самостійно.

Тексти програм і результати обчислень записати в зошит.

Урок 17. Модуль CRT. Графічний режим роботи

Мета: ознайомити з понятями графічного режиму роботи монітору, ініціалізації графічного режиму; формування навичок роботи з процедурами та функціями для побудови графічних зображень; виховання інформаційної культури.

Тип уроку: вивчення нового навчального матеріалу

Хід уроку.

І. Організаційний момент.

ІІ. Викладання нового матеріалу.

1. **Ініціалізація графічного режиму**

Після запуску інтегрованого середовища Турбо Паскаля включається текстовий режим і для використання графіки необхідно виконати ряд дій по переходу в графічний режим.

Насамперед необхідно підключити модуль Graph Турбо Паскаля. У цьому модулі описані процедури й функції, призначені для роботи із графічним екраном, а також деякі вбудовані константи й змінні, які можуть бути використані в графічних програмах. Для того щоб скористатися всіма можливостями модуля Graph, на початку програми (після її заголовка) необхідно розмістити оператор: uses Graph;

Основну частину модуля становлять процедури виводу основних графічних елементів, таких як точки, відрізки прямих ліній, дуги, цілі кола та ін. Такі елементи називаються *графічними примітивами.* Інша група процедур призначена для керування графічним режимом. Усього в бібліотеці модуля Graph перебуває більше 50 процедур і функцій для роботи із графікою.

Далі варто визначити тип відеоадаптера, установленого на комп'ютері. Відеоадаптером називають набір мікросхем, керуючих роботою конкретного дисплея. Можна доручити з'ясувати тип відеоадаптера програмі.

Як майже всякий фізичний пристрій комп'ютера, відеоадаптер може працювати тільки в тому випадку, коли завантажена програма, що управляє його роботою. Така програма називається *драйвером* пристрою (driver у перекладі з англійської мови — «шофер», так що драйвер є «водієм», що управляє роботою пристрою). Графічні драйвери містяться у файлах з розширенням **.BGI** (наприклад, **EGAVGA.BGI** або **IBM8514.BGI). У** Турбо Паскалі вже є необхідний набір .BGI-Файлів, тому програмістові залишається лише вказати у відповідному місці програми розташування каталогу, що містить ці файли.

Графічний режим спочатку треба задати. Це виконують так:

```
<розділи описів та оголошень конкретної програми>;
var driver, mode : integer;   {Для характеристик дисплея}
begin
   driver:= detect; {detect - стандартна стала}
   initgraph (driver, mode, ''); {Задания графічного режиму}
   if graphresult < > 0 then begin
writeln('графічний режим задати не вдалося');
halt            {Стоп}
end;
<текст конкретної програми з графічними командами>
end.
```

2. Процедури і функції для графічних побудов. Розглянемо процедури модуля **Graph,** призначені для графічних побудов.

initgraph (driver, mode, <шлях до драйвера>) — задає графічний режим. Шлях до драйвера зазначають (у лапках), якщо він не є в тому ж каталозі, що й файл turbo.exe;

detectgraph (<драйвер>, <режим>) — повертає значення характеристик дисплея;

setcolor (<колір>) — задає колір майбутнього зображення;

setbkcolor (<колір>) — задає колір тла;

putpixel (x, y, <колір>) — висвітлює точку (x,y) заданим кольором;

line (xl, yl, x2, y2) — рисує лінію між двома точками;

lineto (x, y) — рисує лінію від поточної точки до точки (x,y);

linerel (dx, dy) — рисує лінію від поточної точки з заданими приростами;

rectangle (xl, yl, x2, y2) — рисує прямокутник з заданими координатами діагонально протилежних вершин (лівої верхньої та правої нижньої);

setviewport (xl, yl, x2, y2, true) — задає координати нового графічного вікна. Логічна стала true задає режим відсікання зображення, яке виходитиме за межі вікна;

bar (xl, yl, x2, y2) — рисує зафарбований прямокутник; **bar3d** (xl, yl, x2, y2, <об'ємна глибина>, true) — рисує паралелепіпед;

circle (x, y, **R)** — рисує коло з радіусом R і центром у (x,y); **arc** (x, y, <початковий кут>, <кінцевий кут>, <радіус>) — рисує дугу; **pieslice** (x, y, <початковий кут>, <кінцевий кут>, <радіус>) — рисує зафарбований сектор;

ellipse (x, y, <початковий кут>, <кінцевий кут>, <горизонт. радіус>, <вертик. радіус>) — рисує еліпс чи дугу еліпса;

setfillstyle (<заповнення>, <колір>) — задає спосіб заповнення замкнутої області залежно від значення параметра заповнення: 0 — заповнення кольором фону, 1 — суцільне заповнення, 2 — заповнення товстими горизонтальними лініями, 3 — заповнення нахиленими лініями, ..., 10 —заповнення точками, 11— щільне заповнення точками;

floodfill (x, y, <колір межі>) — заповнює замкнену область, що містить точку (x,y);

closegraph — закриває графічний режим;

outtext (<текст>) — виводить заданий текст з поточної позиції;

outtextxy (x, y, <текст>) — виводить текст у заданому місці;

settextstyle (<шрифт>, <напрям>, <розмір>) — задає вигляд символів, напрямок виведення: 0 — горизонтально чи 1 — вертикально, і розміри символів: 1, 2, 3.

***3. Розглянемо деякі функції модуля* Graph.**

graphresult — повертає код помилки, якщо неможливо задати графічний режим, і 0 — у разі задания;

getmaxx — повертає значення розміру екрана уздовж осі OX; **getmaxy** — повертає значення розміру екрана уздовж осі 0У; **getcolor** — повертає значення поточного кольору; **getcolor(x.y)** — повертає значення кольору точки (x,y); **getx, gety** — повертають координати поточного пікселя.

4. Кольори. Кольори задають числами або англійськими назвами:

black=0 - чорний;	darkgray=8 - темно-сірий;
blue=l - синій;	lightblue=9 - яскраво-синій;
green=2 - зелений;	lightgreen=10 - яскраво-зелений;
cуап=3 - блакитний;	lightcyan=ll - яскраво-блакитн.;
red=4 — червоний;	lightred=12 — яскраво-червоний;
magenta=5 - фіолетовий;	lightmagenta=13 - яскраво-фіол.;
brown=6 — коричневий;	yellow=14 - жовтий;
lightgray=7 - світло-сірий;	white=15 - білий.

Задача 1. Нарисувати різними кольорами десять концентричних кіл, які мають спільний центр по середині екрана, тобто в точці з графічними координатами (320; 240), і описати навколо кіл червоний прямокутник.

```
program Circle10;
uses Crt, Graph;
var driver, mode, r : integer;
begin  clrscr;
    driver := detect; initgraph( driver, mode, '');
    r := 10;       {Радіус першого кола 10 пікселів)
    while r <= 100 do
    begin
    setcolor(r div 10); circle(320, 240, r);
    r := r + 10 end; setcolor(red);
rectangle(220, 140, 420,340); readln
end.
```

Задача 2. Нарисувати емблему. У верхній лівій частині графічного екрана на чорному фоні нарисувати блакитний квадрат, а в ньому - чорне коло, зафарбоване жовтим кольором. У центрі емблеми чорними літерами написати слово "Балта".

```
program Emblema;
uses Crt, Graph;
var driver, mode: integer;
begin  clrscr;
driver:—detect;
initgraph(driver,mode,'');
setbkcolor(O);
setcolor(3);
rectangle(100,0,300,200);
setfillstyle(l,3);
floodfill(200,100,3);
setcolor(14);
circle( 200,100,100);
setfillstyle(l,14);
floodfill(200,100,14);
setcolor(O);
```

```
circle(200,100,100);
settextstyle(0,0,3); outtextxy(135,95, 'Балта'); readln end.
```

Побудова графіків функцій. Графічні команди використовують також для рисування графіків функцій.

Задача3. Зобразити на екрані графік функції y=sinx.

```
program Grafiksin;
uses crt, graph;
var driver, mode, i, x1, y1 : integer; x, y : real;
begin   clrscr; driver := detect;
initgraph(driver, mode,'');
setcolor(4); setbkcolor(11);
setlinestyle(0,1,3);
line(20, 240, 450, 240); line (60, 340, 60, 120);
x:=0;   x1:=60; y1:=240;
Moveto(x1, y1); setcolor(8);
while x<=2*pi+0.1 do  {розгл. відрізок [0;2π]}
begin
y:= sin(x);
y1:= - trunc(100*y) + 240;
lineto(x1, y1);
x:=x+0.1; x1:=x1+5
end;
settextstyle(0, 0, 1); outtextxy(60, 245, '0');
outtextxy(360, 245, '6.3'); settextstyle(0, 0, 2);
outtextxy(80, 100, 'Графік функції sin(x)')
 end.
```

Задача 4. Різнобарвні смуги. Намалювати 14 різнобарвних вертикальних смуг, пофарбованих 14 кольорами (крім білого й чорного), можна за допомогою процедур малювання ліній і установки кольору:

```
program stripes;
uses Graph;
 var gd,  gm,  c,  x,  y,  i  :  Integer;
begin
gd  := Detect;
InitGraph(gd, gm, '');
SetBkColor(white);
ClearDevice;
x := 0;
for c := 1 to 14 do
begin
x := x + 35;
SetColor(c) ;
Line(x, 0, x, 400);
for i :=1 to 5 do
Line(x + i, 0, x + i, 400) ;
{Малюємо 6 смуг}
end;
Readln;
CloseGraph;
end.
```

Задача 5. **Малюємо Королеву Краси.** Ну от, тепер можна спробувати намалювати царівну Будур або хоча б схожого на неї зображення.

```
program man; uses Graph;
var gd,  gm  :  Integer;
begin
  gd  :=  Detect;
  InitGraph(gd,  gm,  ' ');
  SetFiUStyleU,  Green);{Трава}
  Bar(0,  350,  639, 479);
  SetFillStyle(1, LightBlue);{Небо}
```

```
  FloodFill(0,   0, Green) ;
  SetColor(Red);
  Circle(320, 200, 19);{Голова}
  SetLineStyle(0, 0, 3);
  Rectangle(300, 220, 340, 300);{Тулуб}
 Line(320, 300, 300, 350);{Ноги}
  Line(320, 300, 340, 350);
  Line(300, 240, 250, 250); {Руки}
  Line(340, 240, 390, 250);
  SetFillStyle(1, Red);
 FloodFill(320, 200, Red)
 FloodFill(320, 230, Red)
 SetColor(Yellow) ;
 Circle(315, 190, 2):, {Ліве око}
 Circle(325, 190, 2); {Праве око}
 Line(315, 210, 325, 210) {Рот}
 Readln;
 CloseGraph;
 end.
```

Ну як? Намальована програмою дівчина... звичайно, далеко не прекрасна дочка султана! Але в Турбо Паскалі є досить засобів і можливостей для того, щоб створити по-справжньому гарну картинку.

Задача 6. **Павутина.** Програма web («павутина») виводить на екран зображення, складене з відрізків прямих і концентричних кіл. Загальний центр точки перетину відрізків і кіл знаходиться в центрі екрана. Тут використовуються функції GetMaxX і GetMaxY. Графічні координати правого нижнього кута екрана рівні (GetMaxX, GetMaxY).

У програмі використовується процедура Delay модуля Crt. Ця процедура припиняє виконання програми на зазначену кількість мілісекунд.

Програма припиняє свою роботу при натисканні клавіші **Enter.**

```
  program web;
  uses CRT, Graph;
  var i : Word; gd, gm : Integer;
  begin
  gd := Detect;
  gm := 0;
  InitGraph(gd, gm, '');
  SetBkColor(Blue);
  SetColor(LightCyan) ;
  Line(0, 0, GetMaxX, GetMaxY);
  Delay(1000);
  SetColor(Yellow);
  Line(0, GetMaxY, GetMaxX, 0);
  Delay(1000);
  SetColor(LightGreen) ;
  Line(0, GetMaxY div 2, GetMaxX, GetMaxY div 2);
  Delay(1000);
  SetColor(LightGray) ;
  Line(GetMaxX div  2,   0,   GetMaxX div  2, GetMaxY);
  Delay(1000);
  SetColor(LightRed);
  for   i:=2   to 20 do
  begin SetColor(16  -  i   div 2);
  Circle(GetMaxX div  2,   GetMaxY div  2, GetMaxY div  i);
  Delay(500  -   15   *   i);
  end;
  ReadLn;
   CloseGraph;
   end.
```

Підсумок уроку.

Домашнє завдання.

Побудувати трикутник з вершинами в точках (100, 100), (150, 100), (80, 70). Колір фону – сірий, колір ліній – червоний.

Урок №18. Елементи комп'ютерної графіки.

Мета: ознайомлення з функціями роботи в графічному режимі; формування навичок роботи в графічному режимі екрану; виховання інформаційної культури.

На цьому уроці пропоную розв'язати цікаві задачі із застосуванням графічного режиму роботи монітору.

Тип уроку: формування навичок.

Хід уроку.

I. Актуалізація опорних знань учнів

- Як задати графічний режим?
- Як нарисувати пряму лінію?
- Як нарисувати коло?

II. Формування практичних навичок та вмінь.

Розв'язування задач.

Задача 1.

Умова: *Скласти програму, яка при натисканні клавіші Д (день) малює сонце, а при натисканні клавіші Н (ніч) малює місяць.*

Розв'язування: По-перше, для вибору малюнку (день чи ніч) введемо символьну змінну *Ch,* залежно від значення якої і будемо малювати сонце чи місяць. По-друге, малювання сонця складається з малювання зафарбованого кола процедурою *FillEllipse* (ця процедура малює зафарбований еліпс, але, якщо еліпс має однакові радіуси по осям, то він перетворюється на коло) та кількох прямих (променів), а місяць можна отримати, якщо накласти одне на одне два кола різних кольорів (жовтого та чорного) з деяким зміщенням. Програма має вигляд:

```
Program Example_1;
Uses graph,crt;  {Підключення бібліотек}
Var GraphDriver,GraphMode:integer;
Ch:char;
Begin  Clrscr;
Writeln('Введіть Ваш вибір:  Д - день,  Н - ніч.');
Readln(ch);
GraphDriver:=VGA;  {Ініціалізація графічного режиму}
GraphMode:=VGAHi;
InitGraph(GraphDriver,GraphMode,'' );
 if  (Ch='Д')  or  (Ch='д')   then
 begin setfillstyle(I,yellow);
setcolor(yellow);
fillellipse(100,80,50,50);  {Малювання сонця}
{Малювання променів}
line(100,80,250, 80) ;   line(100,80,240,30) ;
line(100,80,200,250);  line(100,80,230,180);
line(100,80,150,250);  line(100,80,100,300);
line(100,80,50,380);  line(100,80,20,280) ;
line(100,80,0,150);  line(100,80,0,80);
line(100,80,0,30);  line(100,80,10,0) ;
line(100,80,50,0);  line(100,80,100,0);
line(100,80,150,0);
 end
else
if  (Ch='H')  or  (Ch='H')   then
begin
setfillstyle(1,yellow);  setcolor(yellow);
fillellipse(100,80,50,50);  setfillstyle(1,black);
setcolor(black);  fillellipse(130,80,50,50);
end
else writeln (' Ви помилилися!') ;
Readkey;     Closegraph;
End.
```

Задача 2. ***Умова:*** *«Зоряне небо». Заповнити екран монітора різнокольоровими точками, кількість яких, колір та координати визначаються випадково.*

Розв'язання: Для вибору випадковим чином вказаних величин скористуємось функцією *Random,* що вибирає числа із заданого діапазону, д причому врахуємо, що, якщо в дужках після функції вказане ціле число, то будуть генеруватися цілі числа в діапазоні від 0 до вказаного числа. Зверніть увагу на те, що всього можливих кольорів 16 (від 0 до 15), але на чорному тлі чорний колір (з нульовим номером) не видимий, тому можна скористатися такою формулою для отримання ненульових цілих чисел в діапазоні від 1 до 15: random (14) +1

Аналогічно можна вибрати координати та кількість «зірок» (точок) на екрані, причому відслідкувати, щоб кількість ніколи не була нульовою. Сама «зірка» (точка) на екрані може бути отримана процедурою *Putpixel,* що задає колір та координати точки виведення. Програма має вигляд:

```
Program Example_2;
Uses  graph;
Var GraphDriver,GraphMode:integer;
x,y,color,N:integer;   {x,y - координати точки  -   'зірки', color - колір точки,  N - кількість точок)
i:integer;   {i - змінна циклу)
Begin Randomize;
GraphDriver:=VGA;
GraphMode:=VGAHi; InitGraph(GraphDriver,GraphMode,' ') ;
{Генерується кількість точок в діапазоні від 200 до 1200} N:=random(1000)+200;
for i:=1  to N do
begin x:=random(640);   y:=random(480);   color:=random(14)+1; putpixel (x,y,color) ;  {Виведення
піксела заданого кольору
color у задані координати екрану x та y}
 end;
Readkey;     Closegraph;
End.
```

Запитання для перевірки засвоєння знань

1. Які процедури визначають тип ліній?
2. Як скористатися процедурою малювання еліпса для малювання кола?
3. Що задає процедура ***PutPixel?***
4. Якими процедурами визначається тип зафарбування?
5. Які процедури дають змогу намалювати пряму лінію?

III. Підсумок уроку

Домашнє завдання:

Побудувати і замалювати зеленим кольором круг радіусом 100, центр якого співпадає з центром екрана дисплея. Колір фону – малиновий.

Урок 19-20. Практична робота №2

Мета: формування навичок роботи з діловою графікою засобами мови Паскаль на прикладах розв'язання задач; виховання інформаційної культури.

На цьому уроці пропонується показати можливості мови Паскаль при побудові графіків функцій та різного виду діаграм.

Тип уроку: формування навичок

Хід уроку.

I. Повторення правил ТБ

II. Практична робота

Виконайте завдання.

Умова: *Зобразити на екрані монітора декартову систему координат, початок якої збігається з центром екрана.*

Розв'язування: Для малювання осей *x* та *y* слід скористатися процедурою *line,* причому координати початку та кінця цих прямих обчислити неважко, тому що вони мають розміщуватись в центрі екрану. Градуювання осей робиться теж за допомогою коротких відрізків довжиною 8 пікселів, що розташовані з кроком *step* пікселів (крок в програмі заданий у вигляді константи, хоча можна його задавати і іншим методом). Підписи на осях можна зробити таким чином: число, що треба написати, переводиться в рядок процедурою *str,* а потім виводиться на екран процедурою *OutTextXy.* Зверніть увагу на те, що на від'ємному проміжку вісі до числа ліворуч дописується знак «-» командою *S := '-' + S,* де S - рядок, що містить підпис під поділкою. Для якісного оформлення малюнку використовується процедура *settextjustify (1,1),* що забезпечує відцентроване виведення тексту у вказану позицію. Програма, що реалізує алгоритм, має вигляд:

```
Program Example_1;
Uses graph; {Підключення бібліотек}
```

```
Const Step=25; {Крок між поділками на осях}
Var GraphDriver,GraphMode:integer;
x,y:integer; {x,y - координати центру дехарт.сист. коорд.}
r:integer;  {r - відстань від центру координат до чергової поділки}
S:string;   {S - рядок, де зберігається символьне значення підпису для поділки}
Begin   Randomize;
GraphDriver:=VGA;  {Ініціалізація графічного режиму}
GraphMode:=VGAHi;
InitGraph(GraphDriver,GraphMode,'');   {Малювання осей}
line(0,240,640,240);
 line(320,0,320 , 480) ;
 {Малювання стрілочок на кінцях осей}
line(630,235,640,240); line(630,245,640,240);
line(315,10,320,0); line(325,10,320,0);
{Підписи на осях)
outtextxy(330,5,'Y'); outtextxy(630,220,'X') ;
x =320; y:=240; r:=0;  {Малювання та підпис поділок на вісі X}
while x+r<640 do
  begin
   line(x+r,y-4,x+r,y+4); line(x-r,y-4,x-r,y+4) ;
   r:=r+step;  str(r div step, S);
   settextjustify (1,1);
outtextxy(x+r,y+10,S); J   s:='-'+S; outtextxy(x-r,y+10,S);
  end;
  r:=0; {Малювання та підпис поділок на вісі Y}
  while y+r<480 do
  begin
line (x+4 , y+r, x-4 , y+r) ; line (x+4 , y-r, x-4, y-r) ;
   r:=r+step;
str(r div step, S) ;
settextjustify   (1,1);   outtextxy(x-10,y-r,S); s:='-'+S;
outtextxy(x-10,y+r,S);
end;
Readkey;  Closegraph;   {Закриття графічного режиму}
End.
```

Задача2. *Умова: Стовпчаста діаграма - це послідовно зображені прямокутники однакової ширини, що розташовані на одному горизонтальному рівні. Висота прямокутників пропорційна значенням деякої числової послідовності. Побудувати стовпчасту діаграму за даними п цілими значеннями. Для наочності стовпчики зафарбувати різними кольорами.*

Розв'язання: Для побудови діаграми, по-перше, необхідно задати кількість стовпчиків, тобто кількість числових значень, по яких буде будуватися діаграма, а, по-друге, - самі значення. В даному алгоритмі всі ці величини вводяться з клавіатури, хоча можна передбачити і інші методи, наприклад, заповнення генератором випадкових чисел. Після введення числових даних слід розрахувати коефіцієнти пропорційності по осях X та У, щоб отримати малюнок на весь екран. Врахуємо, що максимальний розмір по осі *X*- 640 пікселів, а по осі У - 480 пікселів. Тоді коефіцієнт по осі *X* можна обчислити за формулою

$$SizeX = \frac{640}{N} - 5$$

Де $SizeX$ коефіцієнт пропорційності, *N*— кількість стовпчиків на діаграмі.

Константне значення 5 від дробу віднімається, щоб розділити стовпчики між собою хоча б на 5 пікселів (це значення можна змінити).

По осі У знайти коефіцієнт пропорційності важче, тому що для цього слід спочатку визначити максимальне значення, що використовується для побудови діаграми. Для знаходження максимуму використовуємо стандартний алгоритм. *Тут можна запропонувати дітям згадати цей алгоритм самостійно.* Після знаходження максимуму знаходимо коефіцієнт пропорційності по осі Y за очевидною формулою $SizeY = \frac{480}{Max}$

де $SizeY$ - шуканий коефіцієнт, *Max* - максимальне значення з масиву.

Обидва шукані коефіцієнти пропорційності округлюємо функцією *round,* тому що екранні координати не можуть бути дробовими, а після цього нормалізуємо значення масиву множенням на коефіцієнт $SizeY$.

Після підготовки даних можна побудувати стовпчасту діаграму. Кожен її елемент будується процедурою *Bar,* що малює зафарбований прямокутник, колір якого задається процедурою *SetFillStyle.* Очевидно, що ширина кожного стовпчика буде дорівнювати $SizeX$ - *5,* тому початкова координата по осі X дорівнює $(i-1)*SizeX$, а кінцева — $i*SizeX-5$, де *i* - змінна циклу, що рахує номер чергового стовпчика. Початкова координата по осі У буде максимальною, тобто 480, а кінцева координата дорівнювати різниці між 480 та елементом масиву. Програма має наступний вигляд:

```
Program Example_2;
Uses graph;
Var GraphOrivar,OraphMode:integer; N,i,Max :integer;
A:array [1. .100] of integer; {Масив мишень побудови діаграми}
SizeX,  SizeY  :  integer;
{Коефіцієнти пропорційності по відповідний осях)
Begin ClrScr;
Write  ('Введіть кількість стовпчиків у діаграмі:  ');
ReadLn  (N) ;  {Введення значень для побудови діаграми}
for i:=1 to N do
begin Write  ('Введіть A[',i,']:');  ReadLn  (A[i]) ;  end;
Randomize;
GraphDriver:=VGA;  {Ініціаліваціа графічного режиму} GraphMode:=VGAHi;
InitGraph(GraphDriver,GraphMode,'') ;
SizeX:=round(640/N-5);    {Пошук максимального значення в
масиві для   побудови діаграми на весь екран}
Max:=A[1];
 for i:=2 to N do
if A[i]>Max than Max:=A[i];
SizeY:=round(480/Max);  {Перетворення масиву значень у
відповідності з коефіцієнтом пропорційності}
 for i:=1 to N do A[i]:=A[i]*SizeY;
 for i :=1 to N do
begin  {Встановлення випадковим чином кольору зафарбування
стовпчиків діаграми}
SetFillStyle (1,random(14)+I) ;
Bar  ((i-1)*SizeX,480,i*SizeX-5,480-A[i]) ;
End;
Readkey;
CloseGraph;
end.
```

Задача 2. *Секторною діаграмою називають круг, площі секторів якого пропорційні відповідним числовим величинам, узятим з деякої послідовності. Для заданої послідовності з п дійсних чисел побудувати секторну діаграму. Для наочності сектори діаграми зафарбувати різними кольорами.*

Розв'язування: Кількість елементів діаграми в цій задачі вводиться так само, як і в попередньому випадку, тобто з клавіатури, а заповнення масиву даними зробимо генератором випадкових чисел. Далі, як і в попередній задачі, необхідно промасштабувати початкові значення для виведення їх на екран у вигляді кругової діаграми. Для цього спочатку знаходимо суму всіх елементів масиву, а потім масштабуємо їх за формулою

$A_i = \frac{A_i \cdot 360}{Sum}$ де *Sum* - загальна сума елементів масиву, 360 - кількість градусів у повному колі.

Сама діаграма будується за допомогою процедури *PieSlice,* що виводить на екран зафарбований сектор круга. Колір зафарбування задається процедурою *SetFillStyle,* а початковий та кінцевий кути сектора обчислюються від поточного кута *Angle* з урахуванням значення елементу масиву. Центр кола, на якому будується кругова діаграма, завжди константний (320; 240). Програма, що реалізує описаний алгоритм, має вигляд:

```
Program Example_2;
Uses graph;  {Підключення бібліотек}
Var GraphDriver,GraphMode:integer;   N,  i  :  integer;
Sum, Ang : real;
A : array [1..100] of real; S : string;
```

```
Begin
ClrScr; Randomize;
Write ('Введіть кількість елементів діаграми: ');
ReadLn (N) ; {Введення значень для побудови діаграми}
for i:=1  to N do A [i] :=random*200;
GraphDriver:=VGA;
GraphMode:=VGAHi;
InitGraph(GraphDriver,GraphMode,») ;
Sum: =0 ;
for i:=1 to N do Sum: =Sum+A[i] ;
for i:=1 to N do A[i] :=A[i] *360/Sum;
Ang:=0;
for i:=1 to N do
begin
SetFillStyle (1,Random(14)+1) ;
PieSlice (320,240,round (Ang), round (Ang+A[i]),230);
{Виведення на діаграмі числових значень}
Str (A[i]*Sum/360:3:0,S) ;
OutTextXY(round(320+120*cos((2*Ang+A[i]))*Pi/360),
round(240-120*sin((2*Ang+A[i])*Pi/360)),S);
Ang:=Ang+A[i];
end; Readkey;   CloseGraph;  {Закриття графічного режиму}
end.
```

Додаткове завдання. В надану програмою «Павлин» Внести доповнення, які б дозволяли уповільнити появу зображення на екрані.

```
program pavlin;
uses Graph,  Crt;
var gd, gm. xl, x2, a, y2 : Integer;
begin
gd := Detect;
InitGraph(gd, gm, '');
Randomize;
x1 := 0; a : = 1;
while x1<= 600 do
begin
x2 := Round(320 + 140 * Sin(xl / 30));
y2 := Round(240 + 140 * Cos(xl / 30));
Delay(100);
Line(x1, 240, x2, y2);
X1 := x1 + 2;
a := a + 1;
if a > 14 then a := 1;
end;
Readln;
CloseGraph;
end.
```

Запитання для перевірки засвоєння знань

1. Як вивести осі координат на екран?
2. Як «підписати» осі координат?
3. Якою процедурою можна «відцентрувати» текст?
4. Як можна створити стовпчикову діаграму?
5. Навіщо потрібна ініціалізація графічного режиму?

Урок 21 Тематична атестація з теми «Основні поняття мови Паскаль»

Дайте відповіді на запитання:

1. Яке призначення виразів?
2. Які є типи виразів?
3. Який тип дужок використовують у функціях?
4. Які операції визначені над числовими даними?
5. Які операції визначені над числовими цілими даними?
6. Яка різниця між операціями ділення і цілочислового ділення?
7. Сформулюйте правило пріоритетів.
8. Сформулюйте правило дужок.
9. Сформулюйте правило лінійного запису виразів.
10. Сформулюйте правило коректних імен.

Виконайте завдання:

1. Яких значень набудуть такі функції та вирази:
а) abs(-5) (відповідь: 5); б) sin(0); в) sqrt(36); г) 8 **div** 5; д) 9 **mod** 2. Яких значень набудуть такі функції та вирази:
а) 9 **div** 2 / 2 (відповідь: 2.0); б) abs(2-sqrt(9)); в) cos(l **div** 2); г) sqr(5 - 3); д) sqrt(abs(-9)).
3. Напишіть за правилами мови такі вирази:
а) $7x^2 - 3x + 4,5\text{tg}2x$ (відповідь: 7*x*x - 3*x + 4.5*sin(2*x)/cos(2*x));
б) $4x^2 - 8x + \cos 3x$; в) $\text{tg}2x-\sin 3x+2\cos^2(2x^2-l,4)$;
4. Напишіть мовою Паскаль такі вирази:
а) $ax^2 + bx + c$ (відповідь: a*sqr(x) + b*x + c);
б) $a+b\sin x + \cos 3,2x + \cos 2,5x$;
в) $\sin 5c - 6,5d + 4ac$;
5. Які помилки допущені в записах арифметичних виразів:
а) sin(2x) + cosx (має бути так: sin(2*x) + cos(x));
б) 5*x + 6y/(5*x - 2*y; в) sin3*x; г) sin(abs(3*x); д) 2,51x + 5a?
7. Складіть програму для обчислення периметра та площі будь-якого трикутника за відомими сторонами, використовуючи формулу Герона.
8. Виберіть з таблиці свою власну функцію, де *n* — ваш номер у журналі. Складіть програму для обчислення її значння в деякій точці. Обчисліть значення функції в точках 1, 2, 3, 4, для чого виконайте програму чотири рази. Результати запишіть у зошит.

n	Функція $f_n(x)$
1	$9,2\cos x^2-\lvert\sin x/1,1\rvert$
2	$12,4\sin\lvert x/2,1\rvert-8,3\cos 1,2x$
3	$\lvert\cos x/2,7\rvert+9,1\sin(l,2x+l)$
4	$\lvert\sin x/3,12+\cos x^2\rvert -8,3\sin 3x$
5	$\cos\lvert 2x\rvert/l,12-\cos(3x-2)+6,15$
6	$\sin x\cos x^2\sin(x+l,4)+5,14$
7	$\lvert\sin(2x-l,5)+3\sin x^2\rvert+2,38$
8	$\cos x^2\sin(2x-l)+4,29$
9	$\cos(x^2+l)-\lvert\sin 2x-5,76\rvert$
10	$\sin x-\cos x^3\sin(x^2-4,2)+4,27$
11	$\lvert\sin l2x\cos\lvert 2x\rvert/3\rvert+4,21$
12	$\cos x^3/2,l+\cos x^2/l,l-8.3\sin(3x+l)$
13	$\sin x^2\cos x^3$—$\sin x+5,2$
14	$2\sin x\sin(2x-l,5)\cos(2x+l,5)-6$
15	$\lvert\cos x^2-0,51\rvert\sin(3x-4)-4,44$
16	$\cos 2,lx\sin\lvert x\rvert /0,15-5,8$

Скласти і виконати програму, задавши вхідні дані самостійно.
1. Квіткова клумба має форму круга. Обчислити її периметр і площу за заданим радіусом.
2. Обчислити довжину кола і площу круга за заданим діаметром.
3. Ділянка лісу має форму рівнобічної трапеції. Обчислити її периметр і площу за заданими сторонами.
4. Ресторан закуповує щодня масло m_1 кг по 8.50 грн. за кілограм, сметану m_2 кг по 2.40 грн., вершки m_3 кг по 4.10 грн. Визначити суми, потрібні для купівлі окремих продуктів, і загальну суму.
5. Скільки секунд мають доба, тиждень, рік?
6. Обчислити кінетичну $E=mv^2/2$ та потенціальну $P=mgh$ енергії тіла заданої маси *m,* яке рухається на висоті *h* зі швидкістю *v.*
7. Ціни на два види товарів зросли на *p* відсотків. Вивести старі та нові ціни.
8. Обчислити площу поверхні $S=4\pi r^2$ та об'єм $V=4\pi r^3/3$ сфери за заданим радіусом *r.*
9. Швидкість світла 299792 км/с. Яку відстань долає світло за годину, добу?
10. Обчислити об'єм та площу бічної поверхні куба, якщо відоме ребро.

11. Яку площу і периметр матиме квадрат, описаний навколо круга заданої площі S.

3. Організація розгалужень
Урок 22 Логічні операції та вирази

Мета: ознайомлення з поняттями про прості та складені умови, навчити записувати умову логічної задачі у вигляді системи логічних виразів, виховання інформаційної культури.

Тип уроку: вивчення нового навчального матеріалу

Хід уроку.

І. Організаційний момент.

ІІ. Викладання нового матеріалу.

1. Логічні *вирази. Обчислення значень логічних виразів*

Крім арифметичних виразів, у Pascal існує ще один тип виразів – логічний.

Логічним виразом називається такий вираз, внаслідок обчислення якого одержується логічне значення типу true або **false («істина» або «хиба»).**

Із стандартним типом змінних **Boolean,** які можуть набувати лише двох значень **True** або **False,** ви вже ознайомилися. Отже, саме такий тип і мають результати обчислення логічних виразів.

Логічні вирази поділяються на ***прості*** та ***складені.***

Простим логічним виразом називається вираз, який записується за допомогою знаків співвідношень <, >, <=, >=, =, та <>.

Приклади простих логічних виразів можуть здатися вам простими:

a + b > c + d, п > m, x = y.

Порівняйте тепер призначення символів «:=» та «=»! Зверніть увагу на те, що спочатку виконуються арифметичні дії, а вже потім порівняння отриманих результатів.

Складеним логічним виразом називається вираз, в якому використовуються логічні операції and, or, not («так», «або», «ні»).

Наведемо приклади. З математики вам відомі такі записи: $x \in [a, b]$ та $x \notin [a, b]$.

Спробуємо записати їх у вигляді логічних виразів

(x >= a) **and** (x <= b),

(x<a)or(x>b).

Під час запису складених логічних виразів прості логічні вирази обов'язково слід брати у круглі дужки!

Чи можна записати простий логічний вираз n<> m у вигляді складеного? Виявляється, можна:

not (n = m).

Визначимо правила, за якими обчислюються значення складених логічних виразів. Для цього існують таблиці істинності, в яких цифра 0 означає **false,** а цифра 1 - **true.** Наведені таблиці можна перефразувати таким чином.

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

***Логічна операція* and *дає результат* true *тоді і тільки тоді, коли обидва операнда мають значення* true.**

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

***Логічна операція* or *дає результат* true *тоді, коли хоча б один операнд має значення* true.**

A	not A
0	1
1	0

Логічна операція* not *завжди дає результат, протилежний значенню її операнда.

Вираз	Значення	Вираз	Значення
not true	false	**not** false	true
true **and** true	true	true **or** true	true
true **and** false	false	true **or** false	true
false **and** true	false	false **or** true	true
false **and** false	false	false **or** false	false

Приклад. Нехай x=3, y= -9. Розглянемо деякі логічні вирази та їхні значення.

Прості вирази	Значення	Складені вирази	Значення

x=3	true	not(y - 50)	true
x>y	true	(1<x) and (x<5)	true
7 mod 3=1	true	(x>4) or (y<-15)	false
y div 2=4	false	(x>4) or (y>-15)	true

Подвійну нерівність 1<x<5 як складений логічний вираз записують так: (1<x) and (x<5). Сукупність нерівностей вигляду x<1; x>5 — так: (x<1) or (x>5). *Прості логічні вирази, які входять у складені, завжди беруть у дужки.*

2. Логічні змінні

Логічні змінні. Значення логічних виразів можна надавати логічним змінним. Це скорочує текст програми.

Для **роботи** з логічними змінними є тип даних **boolean.** Нагадаємо, що логічних сталих є лише дві: **true** і **false.**

Логічні змінні треба описувати у розділі оголошення змінних так:

var <список імен змінних>: **boolean;**

Наприклад, **var** z, z1, z2: **boolean.**

Нехай x = 2. Якого значення набуде змінна z2 після виконання таких трьох команд присвоювання:

z := x > 5; z1:= **not** z; z2 := z **or** z1 ?

Відповідь: тут **z=false, a zl=true,** тому змінна z2 матиме значення «істина» **(true).**

Довідка. Логічним змінним не можна надавати значення в діалоговому режимі командою **read,** однак їх можна виводити на екран командою **write.**

Задача. Нехай *a, b, c* — коефіцієнти квадратного рівняння $ax^2+bx+c=0$. Відповісти на запитання: вислів «Квадратне рівняння має два дійсні різні корені» є істинний чи хибний?

Розглянемо програму Lohika.

```
program Lohika;
var a, b, c : real; L : boolean;
begin
write ('Введіть числа a,b,c: '); readln (a, b, c); L :== b*b-4*a*c> 0;
writeln ('Квадратне рівняння має два дійсні різні корені — ', L)
end.
```

Якщо під час виконання програми ввести три числа, наприклад, 2.5 8.1 -2.9, то на екрані отримаємо:

Квадратне рівняння має два дійсні різні корені — TRUE.

Висновок

Таблиця пріоритетів арифметичних і логічних операцій має такий вигляд:

1 ( ) — спочатку виконуються дії в дужках
2 Функції, логічна операція **not**
3 *, /, **div, mod, and**
4 +, -, **or**
5 >, <, >=, <=, =, <>

Умовою безпомилкового виконання таких операторів є збігання типів, тобто змінні в лівій частині цих операторів повинні бути описані типом **boolean.**

По-друге, результат обчислення логічних виразів **true** та **false** можна ще трактувати як «так» та «ні». Це наводить на думку про використання логічних виразів для визначення оцінки деякої ситуації, що склалася, і прийняття рішення про те, що робити далі.

III. Закріплення теоретичного матеріалу

Дайте відповіді на запитання

I. Для чого використовують логічні вирази?
2. Що таке простий логічний вираз?
3. Які є символи відношень між величинами?
4. Що таке складений логічний вираз?
5. Які є логічні операції?
6. Дайте означення логічної операції **not.**
7. Дайте означення логічної операції **and.**
8. Дайте означення логічної операції **or.**
9. Який пріоритет логічних операцій?

IV. Формування практичних навичок

Розвязати вправи та задачі

1. Усно. Чи істинний простий логічний вираз x > 10, якщо:

а) x=0 (відповідь: ні); б) x=2; в) x=10; г) x=5; д) x=15?

2. Усно. Чи буде хибним вираз x >= 10, якщо:

а) x=1 (відповідь: так); б) x=3; в) x=10; г) x=12; д) x=25?
3. Чи істинний складений логічний вираз (x > 1) **and** (x < 5), якщо:
а) x=0 (відповідь: ні); б) x=2; в) x=10; г) x=5; д) x=15?
4. Чи істинний складений логічний вираз (x <= 8) **and** (x > 3), якщо:
а) x=0 (відповідь: ні); б) x=2; в) x=10; г) x=5; д) x=15?
5. Якого значення **(true** чи **false)** набуде вираз (x <= 2) or (x > 5), якщо:
а) x=0 (відповідь: **true);** б) x=2; в) x=10; г) x=5; д) x=15?
6. Запишіть логічні вирази для нерівностей:
а) 0<x<10 (відповідь: (x>=0) **and** (x<10)); б) -5<x≤8; в) 2<x≤7; г) x≤1 або x>9; д) x≤2, x>12; є) x≤*0* і *y≥0*.

Підсумок уроку.

Домашнє завдання.

Вивчити теоретичний матеріал уроку.
1. Запишіть логічний вираз для визначення, чи точка *x* належить відрізку:
а) [0; 3) (відповідь: (x>=0) **and** (x<3)); б) [-5; 5); в) [10; 20];
г) [2;14] або [20;25]; д) [4;10] і [8; 12].
2. Запишіть нерівності, які відповідають логічним виразам:
а) (x>2) **and** (x<=20) (відповідь: 2<x<20);
б) (x<-2) **or** (x>15); в) (x>=5) **and** (x<25);
г) (x>3) **and not** (x>5); д) (x>=-5) **and** (x<5) **or** (x>0).

Урок 23. Обчислення значень логічних виразів

Мета: формувати практичні навички та вміння обчислювати значення логічних виразів.

Тип уроку: формування навичок

Хід уроку.

I. Актуалізація опорних знань учнів

Фронтальне опитування.

1. Для чого використовують логічні вирази?
2. Що таке простий логічний вираз?
3. Які є символи відношень між величинами?
4. Що таке складений логічний вираз?
5. Які логічні операції ви знаєте?
6. Дайте означення логічної операції not.
7. Дайте означення логічної операції and.
8. Дайте означення логічної операції or.
9. Який пріоритет логічних операцій?

II. Формування практичних навичок

Розвязування вправ та задач.

1. Запишіть умову того, що число *a* є: а) парне; б) ділиться без остачі на 3; в) не ділиться без остачі на 3; г) ділиться на 3 і на 5; д) ділиться на 3 або на 5.
2. Запишіть умову того, що деякий день року припадає на 6 травня.
3. Ціну товару позначено змінною с. Яка умова того, що ціна: а) не перевищує 20 грн.; б) є більшою, ніж 10 і меншою, ніж 15 грн.?
4. Складіть логічні вирази для перевірки, чи є точка *(x;y):*
а) У другій чверті (відповідь: (x < 0) **and** (y > 0));
б) на координатних осях;
в) у другій або третій чверті;
г) у квадраті зі стороною, що дорівнює 1, побудованому на координатних осях у першій чверті;
д) у крузі одиничного радіуса з центром у початку координат (підказка: умова належності точки кругу така: $x^2+y^2<1$).
5. Поставити у відповідність розташованим ліворуч виразам вирази, що розташовані праворуч:

l) **not**(x = y),	а) $x \in [0,1]$,
2)(x<y)**or**(x = y),	*b)* $x \neq y$,
3) (x< 0)**or**(x> 1),	*c)*$x \leq y$,
4) (x >= 0) **and** (x <= 1),	d) $x \notin [0, 1]$.

6. Обчислити значення логічних виразів:
1) x < y при *x* = 2.5, *y* = **0.1;**
2) a **and not** (b = c) при *a* = **false, *b* = false, *c* = true;**
3) not (a and b) or b = *a* при ***a* = true, *b* = false;**
4) (not a and (x < y)) or (x < 0) при ***x* = -0.1,** *y* **= 0.7, *a* = true.**

7. Записати у вигляді логічних виразів висловлювання, наведені нижче:

1) значення *x* не належить інтервалу (0; 1);
2) значення *x* належить відрізку [- 1; 0] або відрізку [2; 5];
3) точка *M(x; y)* лежить у другій чверті координатної площини;
4) точка *M(x; y)* лежить усередині або на межі одиничного круга з центром у початку координат;
5) точка *M(x; y)* не лежить на одиничному колі з центром у початку координат;
6) 0<A< 1,5;
7) 3 >B>C> 0,1;
8) 5<A<6<B<7,6.

. 8. Математична логіка - це розділ математики, який вивчає методи встановлення істинності або хибності висловлювань. У логіці визначені такі операції: «і» (∧) - кон'юнкція, або логічне множення; «або» (∨) - диз'юнкція, або логічне додавання; «не» (¬) - інверсія, або заперечення. Ці логічні операції повністю відповідають логічним операціям **«and»,** «ог», **«not»,** що використовуються в логічних виразах. Представити у вигляді логічних виразів такі записи:

$1) x \vee \neg y \wedge (x \vee y) \wedge y = z;$

$2) \neg x \wedge (a * b - c) b + d \le \frac{a(b+d)}{c} \vee x;$

$3) 5 - 2a \ne -2 \wedge x \wedge y \vee \neg((x \vee y) \wedge z);$

$4) 3{,}47 \frac{b-c}{a} + b \ge (a - 8{,}96d)^2 \vee a^3 - c < 3b;$

$5) \neg(x \vee y \vee z) \wedge a > \sin c;$

Підсумок уроку.

Домашнє завдання

Розвязати задачу: Олексій, Борис і Григорій знайшли в землі амфору. Кожний з них висловив по два припущення:
А: Це амфора грецька, V століття.
Б: Це амфора фінікійська, III століття.
Г: Це амфора не грецька, IV століття.
Вчитель історії сказав хлопцям, що кожний з них висловив правильну думку тільки в одному з двох своїх припущень.
Де й у якім столітті була виготовлена амфора?

Урок 24 Вказівка розгалуження

Мета: ознайомити учнів з поняттями про структурні оператори, вказівку розгалуження (повну та скорочену форми) і поняттями про прості і складені умови; формувати навички використання вказівки розгалуження; виховувати інформаційну культуру учнів.

Тип уроку: вивчення нового матеріалу.

Хід уроку.

І. Актуалізація опорних знань учнів.

Розвяжіть задачу:

Віктор, Роман, Леонід та Сергій зайняли на математичній олімпіаді чотири перших місця. Коли їх запитали про розподіл місць, вони дали три відповіді:

- Сергій – перший, Роман – другий;
- Сергій – другий, Віктор – третій;
- Леонід – другий, Віктор – четвертий.

Відомо, що у кожній відповіді тільки одне твердження вірне. Як розподілилися місця?

II. Викладання нового навчального матеріалу.

1. *Оператор умовного переходу. Повна та скорочена форми*

Уявіть собі, що ви за кермом автомобіля і перед вами стоїть вибір дальшого руху: або їхати поганою, але коротшою дорогою, або ж гарною, але довшою. Звичайно, що ваш вибір буде залежати від певних умов: по-перше, чи є у вас зайвий час, по-друге, хто господар автомобіля?

Подібну проблему завжди вирішують оператори розгалуження.

Загальний вигляд повної форми оператора умовного переходу:

if <логічний вираз> **then P1 else P2,**

де логічний вираз - це вираз, який може набувати одного з двох значень **true** або **false,** P1 та P2 - це оператори або процедури.

Робота оператора умовного переходу не викликає ніяких труднощів. Цей оператор використовує результат обчислення логічного виразу для вибору того чи іншого шляху наступного виконання алгоритму - виконання оператора P1 або оператора P2. Після цього робота алгоритму продовжується далі за вказаними операторами.

Після аналізу значення логічного виразу буде вибраний лише один з наступних напрямків виконання алгоритму (P1 або P2), після чого цей алгоритм буде виконуватися далі.

Загальний вигляд скороченої форми оператора умовного переходу:

if <логічний вираз> **then** P,

де значення вказаних параметрів такі самі, як і в повній формі.

Відмінність між двома формами умовного оператора: в першій - повній - незалежно від значення логічного виразу якісь дії обов'язково будуть виконані, а вже потім продовжено виконання алгоритму далі, у другій - скороченій - у випадку, коли логічний вираз набуде значення **true,** будуть виконані якісь дії, а потім продовжено виконання алгоритму, а у випадку, коли логічний вираз набуде значення **false,** алгоритм відразу буде продовжено далі.

2. Складений оператор

Розширимо поняття оператора в Pascal. Справа в тому, що ми з вами поки що мали справу лише з *простими* операторами -присвоювання і умовного переходу. А що робити, коли після службових слів **then** або **else** нам потрібно вказати не один такий оператор, а кілька? Для такого випадку в Pascal введене поняття *складеного* оператора.

Складеним оператором називають послідовність кількох операторів або викликів процедур, розділених символом «;» та взятих в операторні дужки **begin** ... **end.**

Складений оператор — це конструкція такого вигляду:

```
begin
    <команда 1>;
    ...
    <команда n>;
end;
```

Складена команда трактується як одна команда.

III. Формування практичних навичок

Тепер уже час переходити до прикладів. Розглянемо алгоритм пошуку найбільшого з двох заданих чисел A та *B.*

```
program max_A_B;
var a, b, max: real;
begin
write ('Задайте два будь-які числа: ');
readln (a, b);
if a > b then
begin
writeln ('Перше число більше за друге.');
max := a
end
 else
begin
writeln ('Друге число більше або дорівнює першому.');
max := b
end
writeln ('Це число -' , max:10:5);
readln
end.
```

Розглянемо детальніше наведену програму. *По-перше,* у ній чітко спостерігається принцип «вкладеності» операторів, тобто сходинкова структура. Наочність такої програми явно виграє! Для цього в ній навіть з'єднані вертикальними лініями оператори різного рівня. *По-друге,* перед закриваючою операторного дужкою **(end)** не стоїть символ «;». І справді, ви ж не ставите кому в тексті перед закриваючою дужкою, коли перелічуєте в ньому в дужках декілька слів. Хоча в Pascal це помилкою. Саме через це не поставлено і символ «;» після останньої процедури **readln.**

Спробуйте відповісти на запитання: при виконанні якої умови буде виконано оператор max := b? Дійсно, за умовою, протилежною a > b, тобто при виконанні умови a <= b!

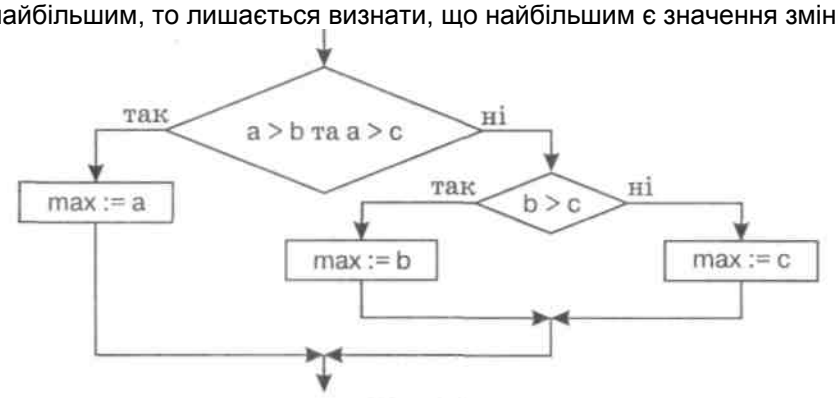
Розглянемо пошук найбільшої з трьох попарно різних величин *a, b, c*

```
program max_A_B_C;
vara, b, c, max: real;
 begin
write ('Задайте три будь-які числа: ');
 readln (a, b, c); if (a > b) and (a > c)
then
```

```
begin
writeln ('Перше число більше за інші два.');
 max := a
end
else
if (b > c) then
 begin
writeln ('Друге число більше за інші два.');
max := b
end
else
 begin
writeln (Третє число більше за інші два.');
max := c
end;
writeln ('Це число - ', max:10:5);
readln
end.
```

У цьому прикладі ми бачимо використання двох вкладених умовних операторів.

Внутрішній оператор умовного переходу буде виконано тоді і тільки тоді, коли значення логічного виразу зовнішнього умовного оператора матиме значення **false.** Це означатиме, що значення змінної a не є найбільшим, тому у вкладеному умовному операторі перевіряється лише значення змінної *b.* Якщо ж і її значення не є найбільшим, то лишається визнати, що найбільшим є значення змінної c. Розглянемо схему цього алгоритму.



Спробуємо записати інший варіант алгоритму пошуку найбільшої з трьох величин.

```
program max ABC;
vara, b, c, max: real; begin
write ('Задайте три будь-які числа: ');
readln (a, b, c); if (a > b) and (a > c) then
begin
writeln {'Перше число більше за інші два.');
max := a end;
if (b > a) and (b > c) then
begin
writeln ('Друге число більше за інші два.');
max := b
end;
if (c > a) and (c > b) then
begin
writeln (Третє число більше за інші два.');
 max := c
end;
writeln ('Це число - ', max:10:5);
readln
end.
```

Отже, у першому алгоритмі внутрішній оператор розгалуження виконається тільки тоді, коли нас «пропустить» до нього зовнішній оператор розгалуження. Цей принцип можна порівняти із ситом.

У другому варіанті алгоритму всі оператори розгалуження є послідовними. Тобто всі вони виконуватимуться, перевірятимуть значення своїх логічних виразів і вирішуватимуть, виконувати чи ні вказані в них дії. Крім того, треба зауважити, що використані тут оператори розгалуження мають скорочену форму.

Звичайно, що з погляду ефективності виконання алгоритму перевагу має перший варіант - у ньому може бути виконано менше перевірок для досягнення необхідного результату. Але, зрештою, все залежить від конкретної умови задачі і вашого бачення її реалізації. Десь ви виграєте на етапі виконання програми, але програєте в її написанні, а десь програма буде зрозумілішою, але кількість виконуваних дій у ній буде більшою.

Розглянемо приклад. 1.

Обчислити і вивести значення складеної функції y.

$$y=\begin{cases}\ln|x|, & x<-1,\\ \sin(x), & -1\le x<1,\\ \cos(x), & x\ge 1.\end{cases}$$

```
program Myfunction;
uses Crt;
var x,y:real;
begin clrscr; writeln('Введіть x'); readln(x);
if x < -1 then y:=ln(abs(x)) else
if (x>=-l) and (x<l) then y:=sin(x) else y:=cos(x);
writeln('x=',x:5:2,'    y=',y:5:2);
readln
 end.
```

2. Визначити, чи є задане тризначне число *паліндромом* (паліндром читається однаково ліворуч праворуч і праворуч ліворуч, наприклад, паліндромами є числа 121, 282, слово «наган»). Для розв'язання цієї задачі можна використати просту умову - перша цифра числа повинна рівнятися останній:

```
program palindrom;
var x  :  Integer;
begin
Write ('Уведіть ціле число:');
readln(x) ;   {Уведення числа  x}
if  x mod  10 = x div  100   then
Write('Уведене число  є паліндромом')
else   Write('Уведене  число  не
є  паліндромом');
end.
```

Підсумок уроку.

Запитання для перевірки засвоєння знань

1. *Який алгоритм називається розгалуженим?*
2. *Які умови називаються складеними?*
3. *Які логічні операції ви знаєте?*
4. *Як записують скорочену форму команди розгалуження?*
5. *Які операції відношення можна записувати в умові?*

Домашнє завдання

1. Уведіть вік користувача і його стать(1 - жіноча, 2 - чоловіча). Складіть
2. програму, за допомогою якої красиво привітайтеся з користувачем залежно від його віку й статі.
3. Уведіть дійсне число x. Складіть програму, за допомогою якої визначите, чи є воно коренем рівняння:
   а). 15x + 100 = 0;
   б). 105 x - 10,18 = 0;
   в). 12 x^3 -2x^2 + 6x + 1 = 0;
   г). 17 x^4 -12x^2 + 2x-1 = 0.
4. Задано вік двох приятелів. Складіть програму, за допомогою якої визначите, хто з них старше.
5. Задано вік трьох друзів. Складіть програму, за допомогою якої визначите, хто із друзів молодше.
6. Задано число x. Складіть програму, за допомогою якої збільшить число x на 10, якщо воно від'ємне, у всіх інших випадках зменшить задане число на 5.
7. Задано число x. Складіть програму, за допомогою якої збільшить число x на 15, якщо воно від'ємне.

Урок 25 Повне розгалуження

Мета уроку: формування навичок складання алгоритмів з використанням команди розгалуження та запису їх мовою програмування; виховання інформаційної культури.

Тип уроку: формування навичок

Хід уроку.

І. Актуалізація опорних знань учнів.

На початку уроку бажано зробити експрес-опитування за матеріалом попереднього уроку (поняття умови, умови прості та складені, поняття команди розгалуження, її форми, запис мовою програмування та мовою блок-схем). Далі пропонується розглянути типові задачі з використанням команди розгалуження.

ІІ. Формування навичок

Задача 1

Умова: Дано значення дійсних величин a, b, c Знайти:

$$\min((a+b+c)/2, \sqrt{1/(a^2+1)+1/(b^2+1)+1/(c^2+1)})$$

```
Program Example_1;
Uses crt;
Var a,b,c  :  real;
Rez1, Rez2, Min  :  real;  {a,b,c - вхідні дані;  Rez1,  Rez2 -проміжні обчислення; Min - результат виконання програми}
Begin
Clrscr;  {Очищення екрану}
Write('Введіть числа a,b,c:  ') ;
Readln(a,b,c);
Rez1:=(a + b +  c)  /2;
Rez2:=sqrt(1/(sqr(a)+1)+ 1/ (sqr (b)+1)+1/(sqr(c)+1)) ;
If Rez1<Rez2 Then Min:=Rez1
Else Min:=Rez2;
Writeln('Min=',Min:8:2);
Readkey;  {Затримка зображення на екрані}
End.
```

Задача 2

Умова: *Дано значення дійсної величини x. Визначити:*

$$\frac{x-5}{x^3+x-2}$$

Примітка: *На перший погляд учні можуть не зрозуміти, навіщо у цій задачі команда розгалуження. Треба їм нагадати відоме правило: ділити на нуль неможна. І тоді розв'язання стає очевидним.*

```
Program Example_2;
Uses  crt;
Var X,Rezultat:real;
Begin
Clrscr;     {Очищення екрану}
Write('Введіть значення X: ');
Readln(X);
If  X*X*X+X-2<>0 Then
begin
Rezultat:=(X-5)/(X*X*X+X-2);
Writeln('Rezultat=',Rezultat:8:2);
end
Else
Writeln('Обчислення неможливі - ділення на нуль!');
Readkey;
End.
```

Задача 3 ***Умова:*** *При даному значенні x обчислити:* $\sqrt{x^3-\sqrt{x-1}}$

Для розв'язання цієї задачі необхідно пам'ятати, що не можна знайти квадратний корінь з від'ємного числа *(зверніть увагу учнів на те, що у прикладі присутні два квадратних кореня).*

```
Program Example_3;
Uses crt;
```

```
Var X, Rezultat: real;
Begin
Clrscr;
Write('Введіть значення X:  ') ;
Readln(X);
If (X>=1) and (X*X*X-sqrt(X-1)>=0) Then
begin
Rezultat:=sqrt(X*X*X-sqrt(X-1));
Writeln('Rezultat=',Rezultat:8:2);
end
else
 Writeln('Обчислення неможливі - від' ємний підкореневий вираз!');
Readkey;
End.
```

Задача 4

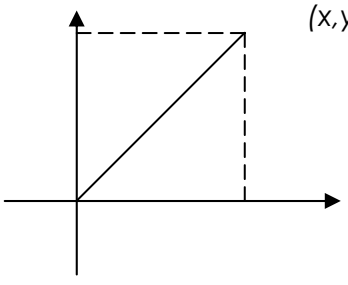
За рейтинговою системою оцінка визначається таким чином: якщо загальний бал учня становить не менше 92% від максимального, то виставляється оцінка 12, якщо не нижче 70%, то — оцінка 8, якщо ж не нижче 50%, то — оцінка 5, в інших випадках - оцінка 2, Визначте оцінку учня, якщо він набрав N балів, а максимальне значення загального балу становить S. У цій задачі можна використати повну або скорочену форму команди розгалуження. Пропонується розв'язування зі скороченою формою.

```
Program Example_4;
Uses crt;
Var N,S,Grade:integer;
{N - бали, що набрав учень; S - максимальне значення сумарного балу;
Grade - оцінка учня}
Begin
Clrscr ;
Write('Введіть максимальне значення сумарного балу, що може набрати учень: ');
Readln(S);
Write('Введіть кількість балів, що отримав учень: ');
Readln(N);
If (S<=0) or (N<=0) or (N>S)
Then writeln('Помилка вхідних даних')
Else
Begin
N:=round(N/S*100); {Знаходження %-відношення балів учня до максимально можливого}
If N>=92 then Grade:=12;
If (N<92) and (N>=70) then Grade:=8;
If (N<70) and (N>=50) then Grade:=5;
If (N<50) then Grade:=2;
Writeln(' Учень отримав оцінку - ' , Grade);
End;
Readkey;
End.
```

Задача 5

Умова: На площині дано дві точки (x_v y_t) та (x_2, y_2). Визначити, яка з них знаходиться далі від початку координат.

Для розв'язання цієї задачі необхідно скористатися теоремою Піфагора для знаходження відстані від початку координат до заданої точки (див. рисунок):



0 *X*

Очевидно, що відстань від початку координат до точки з координатами *(x,y)* буде обчислюватись наступним співвідношенням:

$$S = \sqrt{x^2 + y^2}$$

Зверніть увагу: через те, що кожна з координат у формулі підноситься до квадрату, неважливо, в якій чверті координатної площини буде знаходитись точка. Програма для розв'язання даної задачі має вигляд:

```
Program Example_5;
Uses  crt;
Var X1, Y1, X2, Y2:real;
{X1,Y1, X2, Y2 - координати даних точок}
S1, S2:real;
{S1, S2 - відстані відповідно до першої та другої точки}
Begin
Clrscr;
Write('Введіть координати першої точки: ');
Readln(X1,Y1);
Write('Введіть координати другої точки: ');
Readln(X2,Y2);
S1:=sqrt(sqr(X1)+sqr(Y1));
S2:=eqrt(sqr(X2)+sqr(Y2)) ;
If  S1<S2
Then Writeln('Друга точка далі від початку координат')
Else Writeln('Перша точка далі від початху координат') ;
Readkey;
End.
```

Запитання для перевірки засвоєння знань

1. *Опишіть правила оформлення складних виразів.*
2. *Назвіть пріоритети виконання операцій в програмах?*
3. *Що таке вкладеність операторів?*
4. *У яких випадках використовують логічні оператори or, а у яких and?*
5. *Поясніть відмінність у застосуванні команд Write і Writeln.*

Підсумок уроку.

Домашнє завдання

1.Задано число *p.* Складіть програму, за допомогою якої, якщо *p* менше нуля, значення *y* обчислите по формулі $y = p^3$. Якщо *p* більше або дорівнює нулю, то значення *y* обчисліть по формулі *y* = 1 - *p.*

2.Відомі: денний заробіток двірника й заробіток садівника за тиждень. Складіть програму, за допомогою якої визначите, хто з них більше заробляє в рік.

3.Відомо, скільки в середньому за день заробляє інженер і скільки в середньому за місяць заробляє лікар. Складіть програму, за допомогою якої визначите, хто з них більше заробляє грошей за рік.

4.Задано дійсне число x. Складіть програму, за допомогою якої перевірте виконання нерівності **3/4 < X < 7/8.**

Урок 26 Розвязування задач на використання вказівки розгалуження

Мета: формування навичок створювання математичних моделей задач, складання алгоритмів з використанням команди розгалуження та запису їх мовою програмування; виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

І. Актуалізація опорних знань учнів.

На початку уроку повторити поняття умови, умови прості та складені, поняття команди розгалуження, її форми, запис мовою програмування та мовою блок-схем. На цьому уроці пропонується розглянути задачі з використанням команди розгалуження. Зверніть увагу на те, що перед розв 'язанням цих задач необхідно з учнями розібрати математичну модель задачі, довівши математичні співвідношення, що приведуть до отримання бажаного результату.

ІІ. Формування навичок.

Розв'язування задач.

Задача 86 (задачник Т.П.Караванової)

Чебурашка вирішив купити килими, щоб застелити кімнату, в якій він мешкав разом з Геною. Їхня прямокутна кімната виявилася розмірами A x B, де A та B — цілі числа. Коли Чебурашка запитав у магазині, які килими є у продажу, то продавець повідомив, що є квадратні килими зі стороною C, де C — ціле число. Яку кількість килимів потрібно придбати Чебурашці, щоб накрити максимальну площу кімнати? Килими неможна накладати та підгинати. Визначити, яка площа кімнати буде не накритою килимами. Передбачити ситуацію, коли розміри килиму перевищують розміри кімнати.

Очевидно, що якщо довжина сторони килиму більша за будь-яку зі сторін кімнати, то застелити її цими килимами неможливо. Крім того, для знаходження кількості килимів, що вміщуються по одній зі сторін кімнати без їх підгинання, необхідно поділити націло довжину кімнати на довжину килиму. Загальна кількість килимів знаходиться за формулою: $K=K_1 \cdot K_2$, де K_1 та K_2 — кількості килимів, що вміщуються вздовж двох суміжних сторін кімнати.

Площа, що не закрита килимами, визначається як різниця між площею кімнати та площею всіх куплених килимів,

Використані змінні: A, *B* — розміри кімнати; C—розмір килиму; K_1, K_2 - кількість килимів вздовж однієї та другої стінки відповідно; K - загальна кількість килимів; S —площа кімнати, що не накрита килимами.

Програма, що реалізує алгоритм розв'язання даної задачі, має вигляд:

```
Program Example_86;
Uses ort;
Var  a,b,c,S:word; K,K1,K2   :   word;
Begin
  Clrscr;   {Очищення екрану}
  Write ( 'Введіть розміри кімнати:   ') ;,
  Readln(a,b);
  Write('Введіть розміри килима: ');
  Readln(c);
  If   (c > a)   or   (c > b)
  Then writeln( 'Кімнату неможливо накрити такими килимами')
  Else
  Begin
  K1:=a div c;  K2:=b div c;
  K   := K1*K2;       S   := a*b - K*c*c;
  Writeln( 'Кількість  куплених килимів   ',   K);
  Writeln('Площа кімнати,   що не накрита килимами  ',   S);
   End;
  Readkey;
   End.
```

Задача 89

Від річкового вокзалу відійшли одночасно у протилежних напрямках теплохід та турист. Теплохід рухався зі швидкістю V_1 км/год, а турист по стежці вздовж річки зі швидкістю V_2 км/год. Якщо через N годин турист передумає і вирішить попливти річкою назад за теплоходом зі швидкістю V_3 км/год, то чи встигне він підсісти на теплохід, який має за графіком зупинку через Y годин після початку руху і стоїть на цій зупинці Z годин? Вважати на те, що всі події відбувалися протягом однієї доби.

Якщо турист протягом N годин рухався в протилежному напрямку від теплоходу, то відстань між ними в той момент, коли турист вирішив наздогнати теплохід, була наступна:

$S=(V_1 + V_2) \cdot N$ де V_1 та V_2 — швидкості теплоходу та туриста відповідно.

Швидкість, з якою турист почне наздоганяти теплохід, — $(V_3 – V_1)$ км за годину, де V_3 — швидкість, з якою турист попливе навздогін теплохода. Час, який буде у туриста для наздоганяння, $(Y-N + Z)$ годин, тому що зупинка в теплохода буде за розкладом через У годин після початку руху, але *N* годин він уже плив, а Z годин теплохід буде стояти на цій зупинці. Тоді за цей час турист пройде відстань: $S_t=(V_3 – V_1) \cdot (Y – N + Z)$

Отже, турист встигне підсісти на теплохід тільки в тому випадку, якщо відстань S_t буде не менше, ніж відстань, на яку теплохід перегнав туриста. Програма, що реалізує запропонований алгоритм, має вигляд:

```
Program Example_89;
 Uses crt;
Var V1,V2,V3:real;  N,Y,Z   :   real;
Begin Clrscr;
Write('Введіть швидкості теплоходу та туриста:   ') ;
Readln (VI, V2) ;
Write ('Введіть час, черев який турист підсів на теплохід:')
 Readln(N);
Write('Введіть швидкість,   з якою турист плив за теплоходом, час зупинки теплоходу, та тривалість
зупинки:')
Readln(V3,Y,Z);
If  (V1<=0)or(V2<=0)or(V3<=0)or(N<=0)or(Y<=0)or(Z<=0) Then
 writeln('Помилкові вхідні дані')
 Else
```

```
Begin
S:=(V1+V2)*N;
St:=(V3-V1)*(Y-N+Z);
If  St >= S
Then writeln('Турист встигне на теплохід.')
Else
writeln('Турист не встигне на теплохід.');
End;
Readkey;
End.
```

Задача № 90

Жили собі дід і баба, і був у них город прямокутної форми. Довжина городу була А м, а ширина складала В м. Якось дід посварився з бабою і вирішив поділити город порівну. Тепер у діда квадратний город зі стороною С м, відрізаний скраю, а решта дісталася бабі. Визначити, чи не залишилася баба ошуканою та якої форми дістався їй город - прямокутної чи квадратної?

Взагалі задача має дуже простий розв'язок: адже бабуся не буде ошуканою в тому випадку, якщо площа городу, що залишилася для неї, не буде меншою, ніж площа дідусевого городу, тобто $C^2 <= A \cdot B - C^2$. Та це тільки на перший погляд. Насправді в даній задачі може бути велика кількість винятків.

Наприклад, якщо дідусь захоче відрізати собі город зі стороною більшою, ніж сторона всього городу, то це неможливо зробити взагалі. Якщо ж він відріже, то город, що залишиться, може мати квадратну, прямокутну або іншу форми.

Програма, що реалізує запропонований алгоритм, має вигляд:

```
Program Example_90;
Uses crt;
Var A,B,C:real;
Begin
Clrscr;
Write('Введіть розміри городу:   ');
Readln(A,B);
Write('Введіть довжину сторони дідусевого городу: ');
Readln(C);
If (A<=0)or(B<=0)or(C<=0) then writeln ('Помилкові вхідні дані')
Else
Begin
If (C>A) or (C>B)
then writeln('Дідусь не зможе відрізати город такого розміру')
else
begin
If A*B-sqr(C)<=sqr(C) then writeln('Бабуся ошукана.')
else writeln('Бабуся не ошукана.');
If (A<>C) and (B<>C)
Then writeln('Город залишився іншої форми')
Else
If ((A=C)and(B/2=C))or((B=C)and(A/2=C))
 then writeln('У бабусі квадратний город.')
else writeln('У бабусі прямокутний город.') ;
end;
 End;
 Readkey;
End.
```

Задача №91

Умова Трьом Товстунам подали на десерт кремові тістечка. Маса одного тістечка — Хкг, а маса Товстунів відповідно X_1 кг, X_2 кг та X_3 кг. Перший Товстун з'їв N тістечок. Кожний наступний Товстун з'їдав у два рази більше від попереднього, але при цьому він не міг з'їсти більше половини своєї власної ваги. Скільки тістечок було з'їдено Товстунами за обідом?

Зверніть увагу на те, що другий та третій Товстуни за умовою можуть з'їсти тістечок у два рази більше, ніж попередній Товстун, але не можуть з'їсти більше половини своєї ваги. Тому фактично в задачі необхідно перевірити, чи не перевищує кількість тістечок, що може з'їсти кожний Товстун, дозволену масу, і у відповідності до цього підрахувати кількість тістечок, що були з'їдені.

Наприклад, якщо другий Товстун може з'їсти 2 • N тістечок, то вага цієї їжі буде *2·N·X кг.* Але за умовою він не може з'їсти більше половини своєї ваги, тобто більше ніж $X_1/2$ кг. Тому якщо вага тих тістечок, що Товстун може з'їсти, не перевищує поріг $X_1/2$ кг, то ми до загальної кількості тістечок додаємо всі можливі, тобто *2·N,* якщо ж перевищує, то ми додаємо тільки ту кількість тістечок, що не дає змоги перевищити припустимий поріг, тобто $X_1/2/X$ (дозволена вага їжі поділена на вагу одного тістечка). Якщо в цьому випадку число вийде нецілим, то це означає, що Товстун з'їв тістечко не повністю. Щоб такого не трапилось, ми робимо відкидання дробової частини після ділення за допомогою функції *trunc.*

Програма, що реалізує цей алгоритм, має вигляд:

```
Program Example_91;
Uses crt;
Var X,X1,X2,X3:real; N,Counter : integer;
{N - кількість тістечок, що з'їв перший Товстун; Counter -загальна кількість з'їдених тістечок}
Begin
Clrscr;
Write('Введіть вагу тістечха:  ');
Readln(X);
Write('Введіть вагу Товстунів  (1-го,  2-го та 3-го):  ');
Readln(X1,  X2,  X3);
Write('Введіть кількість тістечок, що з'їв перший Товстун ') ;
Readln(N);
If  (X<=0)or(X1<=0)or(X2<=0)or(X3<=0)or(N<=0) Then
 writeln('Помилкові вхідні дані')
Else
 Begin
Counter:=N;  {З'їв перший Товстун)
If N*2*X<=X2/2    Then Counter:=Counter+2*N
Else
 Counter := Counters- trunc(X2/2/X);
If N*4*X<=X3/2   Then Counter:=Counter+4*N
Else
Counter:= Counter* trunc(X3/2/X);
Writeln('Кількість з'їдених тістечок:  ' , Counter);
End;
Readkey;
End.
```

Запитання для перевірки засвоєння знань

1. *Що таке глобальний блок програми?*
2. *Поясніть особливості використання локальних змінних.*
3. *Які операції можна виконувати зі змінними цілого типу?*
4. *Для чого в програмах використовують коментарі?*
5. *Поясніть використання функції* ***trunc.***

Підсумок уроку.

Домашнє завдання:

1. Задано довжини сторін трикутника. Складіть програму, за допомогою якої визначіть, чи є цей трикутник рівнобедреним, рівностороннім чи різностороннім.
2. Задано три числа a, b, c. складіть програму, за допомогою якої визначіть, чи існує трикутник з такими довжинами сторін. Якщо да, то знайдіть його периметр.

Урок 27 Розвязання задач на використання вказівки розгалуження

Мета: формування навичок складання алгоритмів з використанням команди розгалуження та запису їх мовою програмування; виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

І. Актуалізація опорних знань учнів

Фронтальне опитування

1 *Що називають логічними виразами? На які два типи вони поділяються?*
2 *Які знаки співвідношень використовуються для запису простих логічних виразів?*
3 *Назвіть логічні операції.*
4 *Поясність використання логічних операцій за таблицями істинності.*
5 *Що називають складеним оператором? Які правила його запису?*
6 *Яким чином організоване розгалуження у Паскалі?*
7 *Чим відрізняються повна та скорочена форми оператора умовного переходу?*

8 Запишіть загальний вигляд повної форми розгалуження.
9 Запишіть загальний вигляд скороченої форми розгалуження.
10 Намалюйте схеми алгоритмів обох варіантів розгалуження.

II. Формування навичок

Розв'язування задач

Задача 120 (задачник Т.П.Караванової)

Дано натуральне число N (N k 1000). *Визначити суму першої і останньої цифр даного числа.*

Для розв'язання цієї задачі ми скористаємося стандартними операціями цілочисельного ділення та остачі від ділення цілих чисел (операції div та mod). Нагадаємо, що результатом ділення числа націло на 10 буде ефект відкидання «молодшої» цифри числа (відповідно при діленні на числа 100,

1000,10000 тощо будемо «відкидати» дві, три або чотири цифри числа). Результатом ж операції знаходження залишку від ділення на 10 буде остання цифра числа (відповідно при знаходженні залишку від ділення на 100, 1000, 10000 будемо отримувати дві останні, три останні, чотири останні цифри числа).

Наприклад:

234 div 10 = 23
9213 div 100 = 92
52 mod 10 = 2
2845 mod 1000 ■ 845.

Виходячи з усього сказаного, програма буде мати вигляд: Program Example_120_2;

```
Uses crt;
Var N,  First,  Last  :  word;
{First - перша цифра числа;  Last - остання цифра числа)
Begin
Clrscr;
Write('Введіть число: ');
Readln(N);
Last := N mod 10;
If (N>=0) and (N<10) then First:=0;
If (N>=10) and (N<100) then First:=N div 10;
If (N>=100) and (N<1000) then First:=N div 100;
If (N=1000) then First:=1;
Writeln ('Сума першої та останньої цифр дорівнює' , First+Last) ; Readkey;
End.
```

Задача.

Популярність діалогових програм, що дозволяють зробити вибір із запропонованого користувачу «меню», всім відома. Розглянемо задачу, яка за генератором випадкових чисел визначає виграш в лотереї таким чином: 1 - відеомагнітофон; 2 -музичний центр; 3 - комп'ютер; 4 - комп'ютерний клас.

Основою даного алгоритму є перевірка значення цілої змінної, наприклад, ch, яке визначає порядковий номер виграшу. Наведемо словесний вигляд алгоритму:

- Якщо значення змінної ch дорівнює 1, то виграно перший за номером приз і переходимо до п.5. У протилежному випадку переходимо до п.2.
- Якщо значення змінної ch дорівнює 2, то виграно другий за номером приз і переходимо до п.5. У протилежному випадку переходимо до п.3.
- Якщо значення змінної ch дорівнює 3, то виграно третій за номером приз і переходимо до п.5. У протилежному випадку переходимо до п.4.
- Якщо значення змінної **ch** дорівнює 4, то виграно четвертий за номером приз і переходимо до п.5.
- Кінець алгоритму.

```
 program prize;
uses CRT; < var n: byte;
ch: char;
begin
writeln ('Призи лотереї:');
writeln ('1. Відеомагнітофон');
writeln ('2. Музичний центр');
writeln ('3. Комп'ютер');
 writeln ('4. Комп'ютерний клас');
writeln ('Натисніть будь-яку клавішу для запуску лототрона');
repeat until KeyPressed;
ch:=ReadKey; randomize; n:=random(3)+l;
write ('Ваш виграш під номером ', п,'. Це - ');
```

```
if n=l then writeln ('Відеомагнітофон')
else if n=2 then writeln ('Музичний центр')
else if n=3 then writeln ('Комп'ютер')
else writeln ('Комп'ютерний клас');
repeat until KeyPressed
end.
```

Прокоментуємо деякі моменти програми. Послідовність викликів стандартних процедури і функції **randomize; n:=random(3)+1** дозволяють за допомогою генератора випадкових чисел змінній n надати будь-якого випадкового значення в інтервалі від 1 до 4. Підключення модуля **CRT** дозволяє використати функції **KeyPressed** та **ReadKey.** За допомогою першої ми можемо затримати виконання програми до натискання користувачем будь-якої клавіші на клавіатурі. Але, оскільки код цієї клавіші буде залишатися в буфері клавіатури і це стане на заваді повторного використання цієї ж функції в кінці програми, то операція **ch:=ReadKey** допоможе «вичитати» цей код із буфера і спорожнити його.

Задача 130

Дано дійсні додатні числа a, b, c, x, y. Визначити, чи пройде цеглина з ребрами a, b, c у прямокутний отвір зі сторонами x та y. Проштовхувати цеглину дозволяється лише так, щоб кожне з її ребер було паралельним чи перпендикулярним кожній зі сторін отвору.

Для розв'язання цієї задачі пропонується впорядкувати розміри отвору та розміри цеглини за зростанням, тобто досягти того, щоб було *a<=b<=c* та x <= y. Тоді перевірка зведеться до порівняння розмірів отвору з найменшими розмірами цеглини (адже ми можемо цеглину розвернути будь-яким боком, щоб проштовхнути її у отвір).

```
Program Example_130;
 Uses crt;
Var a,b,c,x,y,S:real;
{S - допоміжна змінна для обміну місцями значень двох змінних}
Begin Clrscr;
Write ( 'Введіть розміри цеглини: ') ; Readln (a,b,c) ;
Write (‘ Введіть розміри отвору: ') ; Readln (x, y) ;
If (a<=0)or(b<=0)or(c<=0)or(x<=0)or(y<=0)
Then writeln ('Помилка вхідних даних.')
Else
Begin {Впорядкування розмірів цеглини)
If a>b  Then Begin S:=a; a:=b; b:=S; End;
If a>c  Then Begin S:=a; a:=c; c:=S; End;
If b>c  Then Begin S:=b; b:=c; c:=S; End;
{Впорядкування розмірів отвору}
If x>y  Then Begin S:=x; x:=y; y:=S; End;
If (a<=x) and (b<=y) Then writeln( 'Цеглина пройде у отвір.')
else writeln('Цеглина не пройде у отвір.')
End;
Readkey;
End.
```

Запитання для **перевірки** засвоєння знань

1. *Чим відрізняється застосування функцій **Read** і **Readln.***
2. *Як можна перетворити змінну дійсного типу у змінну цілого типу?*
3. *Над якими типами даних можна виконати операцію div?*
4. *Чим відрізняються операції **div** і mod?*
5. *Коли використовують підпрограми?*

Підсумок уроку.

Домашнє завдання:

Складіть програму, за допомогою якої обчисліть значення функції:

$$a)\ y=\begin{cases}1+3x^2, & x<-2;\\ 1-2x^3, & x\ge -2;\end{cases}\quad b)\ y=\begin{cases}x^2+2, & x\le 1;\\ x^2-2x+4, & x>1;\end{cases}$$

$$c)\ y=\begin{cases}\dfrac{4x^2-15}{2x^3-5}, & x\ge 1;\\ \sqrt{2x-5}, & 0\le x<1;\\ |x^3-16|, & x<0;\end{cases}\quad d)\ y=\begin{cases}\sqrt{x^2+5}, & x>0;\\ |2x^3-2x^2|, & -10<x\le 0;\\ \dfrac{5x-1}{x^4+11}; & x\le -10.\end{cases}$$

Урок 28. Практична робота №3 «Програми з розгалуженнями»

Мета: закріплення навичок складання програм з розгалуженнями.

Запитання для перевірки володіння теоретичним матеріалом:

1 У чому полягає відмінність виконання вкладених та послідовно розташованих розгалужень?

2 Яку роль відіграють складені логічні вирази в розгалужених алгоритмах?
3 Чи можна алгоритм у прикладі 3 записати за допомогою послідовних розгалужень? Якої форми вони будуть?
4 Чи можна алгоритм у прикладі 4 записати за допомогою послідовних розгалужень? Якої форми вони будуть?
5 Наведіть власний приклад алгоритму з використанням послідовних розгалужень.
6 Наведіть власний приклад алгоритму з використанням вкладених розгалужень.

За наведеним сценарієм виконайте завдання по створенню та налагодженню розгалужених алгоритмів та програм.

1) *Визначити кількість і типи вхідних даних відповідно до умови задачі.*
2) *Визначити кількість і типи вихідних даних відповідно до умови задачі.*
3) *Визначити тип алгоритму, побудувати математичну модель.*
4) *Визначити необхідність введення проміжних даних, їх кількість та типи.*
5) *Розробити словесний опис алгоритму та побудувати його схему.*
6) *Визначити форми розгалуження, які необхідно застосувати для створення алгоритму відповідно до умови задачі.*
7) *Визначити послідовність введення початкових та виведення результуючих даних з використанням відповідних коментарів і форматування.*
8) *Визначити коректність використання послідовних та вкладених розгалужень.*
9) *Записати алгоритм мовою програмування.*
10) *Набрати текст програми, використовуючи середовище програмування та виконати налагодження програми, виправивши синтаксичні помилки.*
11) *Виконати програму, підготувавши систему тестів.*

Завдання для практичної роботи:

1. Задано значення x. Скласти програму, за допомогою якої обчисліть значення функції:

$$a)\ y=\begin{cases}1+x, & x\le -2;\\ 1-x, & x>-2\end{cases};\quad b)\ y=\begin{cases}\dfrac{2x^2+15}{x^3-1}, & x\ge 1;\\ \sqrt{7x+5}, & x<1;\end{cases}$$

$$c)\ y=\begin{cases}\dfrac{5x^2+12}{2x^2+12}, & x\ge 15;\\ \sqrt{4x+3}, & 0\le x<15;\\ |x^3-17|, & x<0.\end{cases}$$

2. Задано сторону квадрата і радіус круга. Складіть програму, за допомогою якої визначіть, що більше – площа квадрата чи площа круга.

Тексти програм та результати обчислень записати в зошит.

Урок 29 Самостійна робота №2

***Мета:** перевіри вміння учнів застосовувати вказівку розгалуження для розв'язування задач.*

Запитання

1. Який загальний вигляд має команда повного розгалуження?
2. Яка дія команди повного розгалуження?
3. Чи можна ставити ";" перед словом else?
4. Що таке складена команда і коли її застосовують?
5. Що означає удосконалити програму?
6. Чому важливо зменшувати кількість змінних під час складання програм?

Складіть програми, використовуючи повну команду if.

1. Уведіть значення аргументу і обчисліть значення складеної функції

$$y=\begin{cases}x^2, & x<5;\\ x-5, & x\ge 5\end{cases}$$

3. Уведіть число. Виведіть повідомлення чи число кратне 9.
4. Уведіть два числа. Менше замініть сумою цих чисел, більше — їх різницею. Виведіть результати.
5. Уведіть два числа. Чи належить більше число проміжку [10; 20]?
6. Уведіть значення аргументу і обчисліть значення складеної функції

$$y=\begin{cases}x^2, & x<0;\\ x-5, & x>5;\\ 0, & y\ всіх\ інших\ випадках.\end{cases}$$

7. Уведіть значення аргументу і обчисліть значення складеної функції

$$a)\ y=\begin{cases}x^3-5, & x<-2;\\ x+6, & -2\le x<5;\\ x+\dfrac{x^3+7}{x^4+1}, & y\ всіх\ інших\ випадках\end{cases}$$

$$б)\ y=\begin{cases} x^2+8x-6, & x<-2 \text{ або } x>2; \\ x^3+3x^2+8, & x=-2 \text{ або } x=2; \\ x^2\sin 2x, & -2<x<2 \end{cases}$$

8. Уведіть число. Виведіть повідомлення: число додатне, від'ємне чи нуль.

Урок 30 Оператор вибору

Мета: ознайомлення з поняттями вказівки вибору; формування навичок використання вказівки вибору; виховання інформаційної культури.

Тип уроку: вивчення нового матеріалу.

Хід уроку.

І. Організаційний момент.

ІІ. Викладання нового матеріалу.

1. Вказівка вибору

Вказівка вибору це оператор який є узагальненням оператора *if* і дає змогу зробити вибір із довільного числа наявних варіантів. Він складається з виразу, що називається *селектором,* і списку параметрів, кожному з яких передує список констант вибору (список може складатися і з однієї константи). Як і в операторі *if,* тут може бути присутнім слово *else,* що має той же зміст. Формат опису:

case < вираз-селектор > of
< список констант вибору1 > : < оператор 1 >;
< список констант вибору 2 > : < оператор 2 >;
< список констант вибору п > : < оператор п >
[else < оператор п+1 >] end;

Оператор *case* працює наступним чином: спочатку обчислюється значення виразу-селектора, потім забезпечується реалізація того оператора, константа вибору якого дорівнює поточному значенню селектора. Якщо жодна з констант не дорівнює поточному значенню селектора, виконується оператор, що знаходиться за словом *else.* Якщо слово *else* відсутнє, активізується оператор, що знаходиться за словом *end,* тобто перший оператор за межею дії *case.* Селектор має належати до одного з перелічувальних типів (цілого, булівського або літерного). Дійсні та рядкові типи використовувати в якості селектора заборонено. Список констант вибору складається з довільної кількості значень або діапазонів, відділених один від одного комами. Межі діапазону записуються двома константами через складений символ діапазону «..». Тип констант у будь-якому випадку повинен збігатися з типом селектора. Щоб краще зрозуміти використання оператора вибору, розглянемо кілька типових задач.

Задача 134

Розробити діалогову програму, яка запитує вік користувача і визначає, до якої вікової категорії він належить:

1) від 1 до 10 років - дитина;
2) від 11 до 15 років - підліток;
3) від 16 до 20 років - юнак (юнка);
4) від 21 до ЗО років - молода людина;
5) після 31 року - доросла людина.

Особливих пояснень ця задача не потребує, адже її можна розв'язати і за допомогою команди розгалуження. Однак зробимо її за допомогою команди вибору, причому, щоб скористатися гілкою *Else,* будемо вважати,що людина може мати вік не більше 150 років (не зафіксовано рекордів коли людина прожила понад 150 років). Якщо ж користувач введе число, що не входить у дозволений діапазон, будемо вважати, що він пожартував.

```
Program Example_134 ;
Uses crt;
Var Years:byte;  {Years  - вік користувача}
Begin
Clrscr;  {Очищення екрану}
Write('Введіть Ваш вік:  ') ;
Readln(Years);
Write('Ви ');
Case Years of
0..10:  Writeln (' - дитина.') ;
11..15:  Writeln('- підліток.');
16..20:  Writeln('- юнак  (юнка).');
21..3O:  Writeln('- молода людина.') ;
31..150: Writeln('- доросла людина.')
```

```
  Else writeln(', пожартували? Людина стільки не живе!') ;
  End;
  Readkey;  {Затримка зображення на екрані}
  End.
```

Задача№151

Розробити програму виведення інформації про день тижня, (вихідний чи робочий), якщо задано його номер від 1 до 7 (1 - понеділок).

```
 Program Example_151;
  Uses  crt;
  Var Day:byte;  {Day - номер дня тижня}
  Begin
  Clrscr;
  Write('Введіть номер дня тижня:  ') ;
  Readln(Day);
  Case Day of
  1..5:  Write('Це робочий день  ');
  6,7:   Write('Це вихідний день ')
  Else    write('Це не день  ');
  End;
  Writeln('тижня.');
  Readkey;
  End.
```

Розглянемо ще один приклад застосування команди вибору.

Нехай населені пункти позначені номерами від 1 до 8. Вартість одного квитка до конкретного пункту *k* визначається так:

$$Cina = \begin{cases} 22, & k = 1, \\ 25, & k = 2,3,4, \\ 30, & k = 5,6, \\ 35, & k = 7,8. \end{cases}$$

Скільки коштуватимуть *m* квитків до населеного пункту, номер якого вводять з клавіатури?

```
  program Kvytky; uses Crt;
  var k,m,cina:integer;
  begin
  clrscr;
  writeln('Введіть номер пункту та кількість квитків:');
  readln(k.m);
  case k и
  1          : cina:=22;
  2..4       : cina:=25;
  5,6        : cina:=30
  else         cina:=35
  end;
  write(m,  ' квитків до пункту ', k,  ' коштують ');
  writeln(m*cina);
  readln
  end.
```

Якщо під час виконання програми ввести дані так: 3 5, то на екрані отримаємо: 5 квитків до пункту 3 коштують 125.

III. Перевірка засвоєних знань.

Дайте відповідь на запитання:

1. Чим константи відрізняються від змінних?
2. Який тип даних в програмах використовують найчастіше?
3. Коли можна використовувати діапазон даних?
4. Яке призначення алгоритмічної конструкції вибір?
5. Який вигляд має команда case і як вона працює?

Виконайте вправи:

1. Запишіть у вигляді діапазону: а) 2, 3, 4, 5, 6; б) 121, 122, 123, 124; в) ‘a’, ‘b’, ‘c’; г) ‘а’, ‘б’,’в’,’г’.
2. Запишіть діапазон у вигляді списку: а) 10..15; б) -5..2; в) ‘k’..’m’.
3. Для заданого номера дня тижня вивести його назву.

IV. Підсумок уроку.
Домашнє завдання:
Вивчити теоретичний матеріал уроку.
Задайте відстані до міст A, B, C, D. Нехай на 100 км потрібно 7 л бензину. Комп'ютер запитує водія про пункт призначення (треба буде ввести одну букву) і повідомляє про небхідну кількість бензину.

Урок 31. Розв'язування задач на використання оператора вибору.

Мета: формування навичок використання вказівки вибору, виховання інформаційної культури.
Тип уроку: формування навичок
Хід уроку.
І Актуалізація опорних знань учнів.

1. Напишіть інструкцію CASE, яка виводить на екран назву пори року. Змінна month містить номер місяця.
2. Замініть інструкцію

```
if day=7
then write('Неділя') else
if day=6
then write('Субота')
else      write('Робочий день');
```

інструкцією CASE. Відповіді:

1.
```
case month of
  12,1,2:write ('Зима! ' ) ; 3..5:write('Весна!');
  6. .8:write('Літо! ') ;
  9. .11:write('Осінь!') ; end;
```

2.
```
case day of
  7:write('Неділя');
  6:write('Субота');
  else write('Робочий день');
  end;
```

ІІ. Формування навичок.
Розв'язування задач.
Задача1. Ціна купальника у травні становить *a* грн. У червні ціна буде збільшена на *p%,* у липні дорівнюватиме травневій, а в серпні буде зменшена на *q%* порівняно з травневою. Скласти програму, яка після введення номера літнього місяця (month) визначатиме ціну (c) купальника.
Розв'язок задачі запишемо у такому вигляді:

$$c = \begin{cases} a(1+p/100), & якщо \quad month = 6, \\ a, & якщо \quad month = 7, \\ a(1-q/100), & якщо \quad month = 8. \end{cases}$$

Розглянемо програму. Змінну month оголосимо не як integer, а як діапазон 6..8 — звуження типу integer згідно з умовою задачі.

```
program Shopping;
var month : 6..8; a,p,q,c : real; when:string;
begin
write('Введіть ціну, відсотки p та q, номер місяця: ');
readln(a,p,q,month);
case month of
6: begin c := a * (1 + p/100); when :='червні" end;
7: begin c := a; when := 'липні' end;
8: begin c := a * (1 - q/100); when := 'серпні' end end;
writeln('Купальник у  ', when, ' коштує   ', c)
end.
```

Виконаємо програму. Нехай нас цікавить ціна купальника в серпні (month=8). Введемо дані так: 9.8 20 30 8.
Отримаємо результат: Купальник у серпні коштує 6.86.
Задача 2. Виконайте програму Dialog. Уведіть число — кількість років — і прочитайте відповідне повідомлення.

```
program Dialog; var Vik : integer; begin
write ('Скільки Вам років? ');
read(Vik);
```

```
case Vik of
12,13:writeln('Вам ще рано читати цей розділ');
14,15:writeln('Вам ще не можна дивитися фільми для
дорослих');
16,17:writeln('Добре вчіться - батьки будуть пишатися Вами'); 18,19:writeln('Мінздоров'я
попереджає...');
20..23:writeln('Паскаль вивчати вже пізно - пора заміж!')
else
writeln('Закрийте цю книжку! Читайте Н.Вірта!');
end ; readln
end.
```

У програмі Dialog використано діапазон значень 20..23 замість списку чисел 20, 21, 22, 23.

Завдання. Поекспериментуйте з однією з наведених вище програм, модифікувавши її на свій розсуд та ввівши свої дані.

Спробуйте розв'язати самостійно:

1. Дано ціле число *n* (1 < *n* < 12), яке визначає порядковий номер місяця в році. За введеним значенням *n* надрукувати назву відповідного місяця.
2. Розробити програму виведення кількості днів у місяці, якщо останній задається цілим числом від 1 до 12.
3. Розробити програму видачі текстового варіанта шкільних оцінок:
   1) 1, 2, 3 - початковий рівень;
   2) 4, 5, 6 - середній рівень;
   3) 7, 8, 9 - достатній рівень;
   4) 10, 11, 12 - високий рівень.

Підсумок уроку.

Домашнє завдання:

1. Дано ціле число *n* (1 < *n* < 4), яке визначає порядковий номер кварталу року (січень, лютий, березень -1 квартал і т. д.). За вказаним значенням *n* надрукувати перелік місяців, які відносяться до цього кварталу.
2. За даним цілим значенням змінної *k* (1 < *k* < 6), яка визначає день тижня, надрукувати свій розклад уроків.
3. Дано ціле число га (1 < n < 4), яке визначає пору року. За вказаним значенням *n* надрукувати перелік місяців, які відносяться до цієї пори року.

Урок 32. Розв'язування задач на використання оператора вибору.

Мета: формування навичок використання вказівки вибору, виховання інформаційної культури.

Тип уроку: формування навичок

Хід уроку.

І Актуалізація опорних знань учнів.

- Запишіть загальний вигляд повної форми оператора вибору.
- Коли доцільне використання оператора вибору?

ІІ. Формування навичок.

Розв'язування задач.

Задача 165

Дано натуральне число N (N < 100), яке позначає вік людини. Додати до цього числа відповідно слова: «рік», «роки», «років», наприклад: 1 рік, 12 років, але 43 роки.

Очевидно, що для того, щоб правильно дописати відповідне слово, необхідно виділити останню цифру числа, що позначає вік людини. Тоді, якщо це цифра «1», то дописується слово «рік», якщо цифри «2», «3» або «4» - дописується слово «роки», а в усіх останніх випадках - дописується слово «років». Виключенням являється діапазон між 10 та 20 роками: в цих випадках завжди пишеться слово «років».

```
Program Example_165;
Uses crt;
Var Years:byte;  {Years - вік людини}
Begin Clrscr; Write('Введіть Ваш вік:  ') ;
Readln(Years);
If Years>100 Then writeln('Помилкові вхідні дані.')
Else
Begin
Write('Вам  ',Years);
If  (Years>=10)  and  (Years<=20)  Then writeln('років')
Else
```

```
Case Years mod'10 of
1:  writeln('рік.');
2..4 :  writeln('роки.');
0,5..9:  writeln('років.');
End;
End;
Readkey;
End.
```

Розв'яжіть самостійно:

1. Введемо такі позначення для відмінків в українській мові: «називний» - «н» або «Н»;

«родовий» - «р» або «Р»; «давальний» - «д» або «Д»; «знахідний» - «з» або «З»; «орудний» - «о» або «О»; «місцевий» - «м» або «М»; «кличний» - «к» або «К».

Розробити програму, яка за введеним позначенням відмінка видаватиме запитання, на які відповідає іменник у вказаному відмінку, наприклад:

«називний» - «хто?, що?».

2. Використовуючи позначення відмінків із попередньої за дачі, розробити програму, яка за введеним позначенням відмінка видаватиме запитання, на яке відповідає прикметник у вказаному відмінку, наприклад:

«називний» - «який?».

3. Введемо позначення для визначення родів в українській мові:

чоловічий рід - «ч»;

жіночий рід - «ж»;

середній рід - «с». Розробити програму, яка за введеним позначенням роду видаватиме закінчення відповідних йому слів у називному відмінку, наприклад: «жіночий рід» - «-а, -я».

Підсумок уроку.

Домашнє завдання:

Розробити програму-довідник, яка за введеним значенням радіуса *R* пропонуватиме користувачу послуги в обчисленні:

1)1- довжини кола;
2) 2 - площі круга;
3) 3 - об'єму кулі;
4) 4 - площі поверхні кулі.

Урок 33. Практична робота №4

Мета: закріплення навичок складання програм з використанням вказівки вибору.

Запитання для перевірки володіння теоретичним матеріалом:

1.Які числа входять у діапазон 4..9?
2.Які символи належать діаазону 'в '.. 'з '?
3.Який загальний вигляд має діапазон?
4.Коли доцільно використовувати діапазон?
5.Яке призначення алгоритмічної конструкції вибір?
6.Який вигляд має команда вибору case?

За наведеним сценарієм виконайте завдання по створенню та налагодженню розгалужених алгоритмів та програм.

8) *Визначити кількість і типи вхідних даних відповідно до умови задачі.*
9) *Визначити кількість і типи вихідних даних відповідно до умови задачі.*
10) *Визначити тип алгоритму, побудувати математичну модель.*
11) *Визначити необхідність введення проміжних даних, їх кількість та типи.*
12) *Розробити словесний опис алгоритму та побудувати його схему.*
13) *Визначити послідовність введення початкових та виведення результуючих даних з використанням відповідних коментарів і форматування.*

12) *Визначити коректність використання команди вибору.*
13) *Записати алгоритм мовою програмування.*
14) *Набрати текст програми, використовуючи середовище програмування та виконати налагодження програми, виправивши синтаксичні помилки.*
15) *Виконати програму, підготувавши систему тестів.*

Завдання для практичної роботи:

1.Розробити програму видачі номера кварталу, до якого відноситься місяць, заданий числом від 1 до 12.
2.Розробити програму виведення назви дня тижня (понеділок, вівторок тощо), якщо він заданий цілим числом від 1 до 7.
3.Розробити програму виведення інформації про день тижня - вихідний він чи робочий, якщо задано його номер від 1 до 7.
4.За даним порядковим номером *n* ($1 < n < 10$) надрукувати прізвище учня вашого класу з класного журналу.
5. Розробити «меню»-орієнтований алгоритм, який за введеними початковими даними і порядковим номером виводить таку інформацію:

1) прізвище учня;
2) ім'я;
3) номер школи;
4) клас;
5)номер телефону.

Урок 34.Оператор безумовного переходу. Мітки

Мета: *Ознайомити учнів з оператором безумовного переходу, навчити використовувати його при розвязуванні задач; виховання інформаційної культури.*

Тип уроку: вивчення нового навчального матеріалу.

Хід уроку.

І. Організаційний момент.

ІІ. Викладання нового матеріалу.

1. Оператор безумовного переходу.

Для визначення поняття оператора безумовного переходу ознайомимося з поняттям мітки.

Міткою називається описаний в розділі міток **Label** ***ідентифікатор або беззнакове число від 0 до 9999.***

Мітками помічаються виконувані у програмі дії або службові слова **end.** Після кожної мітки повинен стояти символ «:». Кожна мітка у програмі повина бути унікальною. До речі, мітки 0010 та 10 вважаються однаковими!

Наведемо приклади міток:

Label StopLabel, 10, 909, Label_1, Label_2.

Тепер розглянемо **загальний вигляд оператора безумовного переходу:**

goto <мітка>.

Використання оператора безумовного переходу можна продемонструвати на прикладі захисту програми від уведення недопустимих даних.

Нехай за умовою задачі треба задавати лише додатні значення змінних *x, y, z.* Фрагмент програми виглядатиме приблизно так:

```
……
label myjabel;
 var x, y, z: real;
……..
begin
myjabel: write (' Задайте три додатні числа:');
readln (x, y, z);
if (x <= 0) or (y <= 0) or (z <= 0) then begin
writeln (' Ви помилилися, повторіть: ') goto myjabel;
end; { продовження програми}
```

У цьому прикладі використано можливість запису в програмі своїх власних коментарів. Для цього використовуються фігурні дужки «{...}», тобто все те, що взято у фігурні дужки, у Pascal-програмі не обробляється, ігнорується. Цією можливістю буде зручно користуватися згодом під час налагодження програм для тимчасового відключення виконання деяких її фрагментів.

2. Порожній оператор.

Загальний вигляд порожнього оператора: ;.

Тобто якщо ви в програмі випадково поставите підряд «;;», то це компілятором Pascal не буде вважатися помилкою, оскільки між цими двома символами він розпізнає порожній оператор! Але хіба лише для цього потрібен порожній оператор? Інколи виникає необхідність передати керування на неви-конувану частину програми, якою є, наприклад, службове слово **end.** А оскільки мітку можна ставити лише на виконуваній частині, то для цього існує порожній оператор: **label** myjabel;

```
begin
{початок програми }
goto my_label;
{ продовження програми }
My_label: ;
end.
```

ІІІ. Формування навичок.

З *а д а ч а* 1. Обчислити периметр (p) і площу трикутника (s) за трьома відомими сторонами a, b, c Програма повинна перевіряти правильність вхідних даних, тобто, чи існує трикутник.

```
program Trykutnyk;
label 222;
var a, b, c, p, s, piv, z : real;
```

```
begin
222 : write (‘Введіть значення сторін:');
readln (a,b,c);
p := a+b+c; piv := p/2;
z := piv*(piv-a)*(piv-b)*(piv-c);
{z, piv - додаткові змінні, тут використано формулу Герона}
 if z>0 then
begin
s := sqrt(z);
writeln ('Периметр=', p:8:2, ' Площа=', s:8:2)
end;
if z<=0 then
begin
writeln ('Трикутник не існує. Введіть інші дані');
 goto 222
end; readln
end.
```

Виконаємо програму. На запит комп'ютера введемо такі дані: 18, 25, 6 (дані треба вводити через пропуск: 18 25 6). Отримаємо: Трикутник не існує. Введіть інші дані Введіть значення сторін: 18 25 10 Периметр= 53.00 Площа= 74.67

Завдання. Поекспериментуйте з програмою, ввівши дані так: 3 5 7. *Зауваження.* Концепція сучасного (структурного) програмування не рекомендує використовувати команду переходу **goto** в програмах. Цю команду можна замінти іншими алгоритмічними конструкціями, наприклад, командою while, яку вивчатимемо пізніше.

IV. Закріплення матеріалу.

Дайте відповіді на запитання:

1. Яка дія і призначення команди неповного розгалуження?
2. Який вигляд має команда неповного розгалуження?
3. Яка дія і призначення команди переходу?
4. Який загальний вигляд має команда переходу?
5. Що таке мітка?
6. Як оголошують мітки?

Підсумок уроку.

Домашнє завдання.

1. Уведіть два числа і більше замініть сумою цих чисел.
2. Уведіть ціле число з діапазону 2..5. Виведіть його значення словом.
3. Уведіть число — номер дня тижня. Виведіть слово — назву цього дня.
4. Розгляньте 4-5 номерів поїздів і пункти їх призначення. Уведіть номер поїзда. Виведіть на екран назву кінцевого пункту.
5. Створіть словник для перекладу 5-6 іншомовних слів. Уведіть іншомовне слово й отримайте переклад.

Урок 35. Самостійна робота №3

Мета: перевірити знання учнів з теми «Організація розгалужень та вибору»

Тип уроку: перевіка знань

Хід уроку.

Виконайте завдання:

1. Записати алгоритм обчислення значення функції. Y вивести з точністю до 2-х знаків.

$$y = \frac{b + \cos(c)}{3b + 4\sqrt{a} + c} - 6c$$

2. Дано дійсні числа a, b, c. Записати у вигляді умовного оператора дії: а) перевірити, чи є серед них рівні (вивести відповідь «так» чи «ні»); б) менше з чисел a і b замінити півсумою цих чисел, а більше – їх добутком.
3. Написати програму для обчислення значень функції:

$$y = \begin{cases} \dfrac{5x^2 + 1}{\sqrt{x}} + 6, & x < 0.5 \\ \sqrt{x} + 6 + \dfrac{1}{x}, & x = 0.5 \\ \cos(x) + 6x^2, & x > 0.5 \end{cases}$$

Урок 36. Розв'язування задач на використання вказівок розгалуження та вибору

Мета: формувати практичні навички , підготувати учнів до тематичної атестації.
Тип уроку: формування навичок.
Хід уроку.
І. Організаційний момент.
ІІ. Формування навичок.
Розв'яжіть задачі:

1. Дозування ліків для дітей (кількість таблеток денно) залежить від віку (з п'ятирічною градацією). Скласти таблицю доз для ліків. Задум: комп'ютер запитує вік дитини і після введення відповіді повідомляє допустиму дозу ліків.
2. На складі є комп'ютери шести моделей. Кожна модель має номер від 1 до 6, назву і ціну. Задати номер моделі й отримати назву комп'ютера і його ціну.
3. Оплата міжміської телефонної розмови залежить від відстані. Скластиорієнтовну таблицю цін за 1 хв розмови з трьома містами. Задум: комп'ютер запитує першу букву назви міста, повідомляє про тривалість розмови і суму до оплати.
4. Безпечні дози роботи за комп'ютером протягом дня для дітей орієнтовно такі: 6-8 років - 20 хв, 9-11 років - 30 хв, 12-15 років - 60 хв, 16-17 років - 90 хв. Сюжет алгориту: комп'ютер запитує вік та час, уже проведений за комп'ютером, і повідомляє, скільки хвилин залишилося до закінчення сеансу роботи.
5. Складіть програму, де потрібно ввести два цілі числа, менше замінити добутком цих чисел, більше — їх сумою, а якщо числа одинакові, то вивести повідомлення про це.

Підсумок уроку.
Домашнє завдання.

1. Які значення набуватиме логічний вираз (x<5) **or** (x>=10), якщо: а) x=2; б) x=5; в) x=7.
2. Складіть логічний вираз, який перевірятиме чи у вашому класі є учні ріст яких більший, ніж 170 см, вага менша, ніж 70 кг і які займаються у футбольній або волейбольній секції.
3. Уведіть номер місяця. Виведіть кількість днів у ньому.
4. Уведіть два різні цілі числа. Виведіть повідомлення, чи належить більше число проміжку [10; 20], а менше — [5;8].

Урок 37-38. Тематична атестація з теми «Організація розгалужень»

Дайте відповіді на запитання:

1. Що називають логічними виразами? На які два типи вони поділяються?
2. Назвіть логічні операції.
3. Поясніть використання логічних операцій за таблицями істинності.
4. Яким чином організоване розгалуження у Pascal?
5. Чим відрізняються повна і скорочена форми оператора умовного переходу?
6. Намалюйте схеми алгоритмів обох варіантів розгалуження.
7. Що називають складеним оператором? Які правила його запису?
8. У чому полягає відмінність виконання вкладених та послідовно розташованих розгалужень?
9. Запишіть загальний вигляд і схему алгоритму повної форми оператора вибору.
10. Запишіть загальний вигляд і схему алгоритму скороченої форми оператора вибору.
11. Коли доцільне використання оператора вибору?
12. Що називають мітками, які правила їх запису, для чого вони використовуються?
13. Запишіть загальний вигляд оператора безумовного переходу і поясність принцип його роботи.
14. Що таке порожній оператор і в яких ситуаціях його використовують?

Записати за допомогою умовного оператора виконання таких дій:

1) менше з двох дійсних значень *x* та *y* (або будь-яке з них, якщо вони рівні) замінити нулем;
2) найбільше з трьох попарно різних дійсних значень *x, y, z* зменшити на 1/2;
3) присвоїти змінній *x* значення 0, якщо її початкове значення належало відрізку [0; 2];
4) якщо значення змінної *x* від'ємне, то залишити його без змін, у протилежному випадку зробити його таким, як і значення *y;*
5) якщо поточне значення змінної j не більше за 10, то значення j збільшити на одиницю, в протилежному випадку продовжити виконання дій алгоритму.

Використовуючи логічні величини, записати за допомогою умовного оператора виконання таких дій:

1) якщо A < C і B < C, то виконується умова *A = B;*
2) якщо 1 < B < 3 або -3<B<-1,то виконується умова $B^2 = 4$;
3) A ≤ B тільки за умови, що B ≤ 5;
4) *A = B* = 5 тільки за умови, що *C* < 3 < *E;*
5) *A* ≠3 тільки за умови, що B≠5та6≥C>3.

Складіть прогами для розв'язання задач:

1. На одному маленькому квадратному безлюдному острові зі стороною *a* м мешкали *k* потерпілих корабельну аварію Робінзонів. Чи не порушені їхні права на житло, якщо на кожного Робінзона повинно припадати *S* $м^2$? Скільком Робінзонам ще вистачить місця на острові, якщо поблизу трапиться нова аварія?
4. Іван Петрович одягнув нові штани і сів на щойно пофарбовану табуретку. На його штанах з'явилася квадратна

пляма зеленого кольору, довжина сторони якої становила *a* см. Виявилося, що у хімчистку беруть одяг, плями на якому не більші за S $см^2$. Визначити, чи вдалося Іванові Петровичу врятувати свої штани.

5. Щоб бути завжди чистою, людині необхідно *x* (24 < *x* < 50) шматків мила на рік. Якщо мити лише п'яти, то мила знадобиться у 12 разів менше, а якщо мити лише вуха - ще на один шматок менше. Скласти програму, яка за вибором користувача давала б відповідь, яку кількість шматків мила необхідно закупити на *n* років уперед, щоб:

1) митися повністю;
2) мити лише п'яти;
3) мити лише вуха;
4) мити п'яти і вуха.

4. Дано значення цілих величин *x* та *y*. Знайти:

1) max (x, *y)* + min *(x, y);*
2) max^2 (я, *y)* - min^2(x, *y);*
3) max (x^2, *y)* + max (x, y^2*);*
4) min (x + *y, x - y).*

5. Дано значення дійсних величин *a, b, c.* Знайти:

1) max *(a + b + c, a*b*c);*
2) min ((a + *b* + *c)/2,* $\sqrt{1/(a^2+1)+1/(b^2+1)+1/(c^2+1)}$;
3) max (a + *b, b + c)* + min (a + *c, b).*

6.Дано ціле число *n* (1 < *n* < 12), яке визначає порядковий номер місяця в році. За введеним значенням *n* надрукувати назву відповідного місяця.

7.Дано ціле число *n* (1 < *n* < 4), яке визначає порядковий номер кварталу року (січень, лютий, березень - I квартал і т. д.). За вказаним значенням *n* надрукувати перелік місяців, які відносяться до цього кварталу.

8.Розробити програму видачі номера кварталу, до якого відноситься місяць, заданий числом від 1 до 12.

9.Розробити програму виведення назви дня тижня (понеділок, вівторок тощо), якщо він заданий цілим числом від 1 до 7.

10. Розробити програму виведення інформації про день тижня - вихідний він чи робочий, якщо задано його номер від 1 до 7.

11. Розробити програму виведення кількості днів у місяці, якщо останній задається цілим числом від 1 до 12.

12. Дано ціле число *n* (1 < *n* < 4), яке визначає пору року. За вказаним значенням *n* надрукувати перелік місяців, які відносяться до цієї пори року.

13. Розробити програму видачі текстового варіанта шкільних оцінок:

1) 1, 2, 3 - початковий рівень;
2) 4, 5, 6 - середній рівень;
3) 7, 8, 9 - достатній рівень;
4) 10, 11, 12 - високий рівень.

14. За даним цілим значенням змінної *k* (1 < *k* < 6), яка визначає день тижня, надрукувати свій розклад уроків.

3. Організація циклів

Урок 39. Вказівка повторення. Організація циклів.

Мета: ознайомлення з поняттями типів циклів; формування навичок використання типів циклів та їх запису мовою програмування Паскаль і мовою блок-схем; виховання інформаційної культури.

Тип уроку: вивчення нового навчального матеріалу

Хід уроку.

І. Мотивація навчання.

Давайте спробуємо на природних явищах і прикладах з повсякденного життя визначити поняття циклу.

Якщо вас запитають, що таке цикл, то, напевно, ви, не замислюючись, відповісте, що це повторюваність чогось.

І це абсолютно правильно!

Визначити повторюваність пір року в природі — це цикл, кругообіг води в природі—це цикл, зміна дня і ночі—це цикл і багато інших процесів у природі повторюються, утворюючи циклічність.

Цикли в математиці — це явище, яке дуже часте зустрічається.

Наприклад, поки натуральні числа менші 10, то їх треба підсумовувати.

Іншими словами, ми знаходимо суму чисел від 1 до 10.

У цьому прикладі повторюється додавання натуральних чисел, поки виконується умова (числа менші за 10).

Такі цикли називаються циклом з передумовою, оскільки умова записується перед виконанням повторюваної групи операторів.

Увага! Цикл у програмуванні — це багаторазово виконувана група команд.

ІІ. Актуалізація опорних знань учнів.

Повторення загальних відомостей про цикли.

ІІІ. Викладання нового навчального матеріалу.

Як же ж цикли полегшують життя програмістам! Уявіть собі на хвилинку, що вам довелося б писати повторення одних і тих самих фрагментів програм багато разів! У Pascal передбачено три різновиди операторів циклу. Всі вони різні за своїм записом і застосуванням.

Загальний вигляд оператора циклу з передумовою:

while <логічний вираз> do P,

де логічний вираз приймає одне з двох значень **true** або **false,** P - простий чи складений оператор. Цикл з передумовою працює за таким принципом: *повторення оператора* P *буде виконуватися доти, поки логічний вираз в операторі циклу отримує значення* true. *Якщо тільки на деякому кроці циклу логічний вираз набуде значення* **false,** *цикл припинить свою роботу.*

Одне важливе зауваження: оскільки виконувані дії знаходяться за всіма службовими словами оператора циклу з передумовою, то не можна забувати про операторні дужки у випадку, коли тіло циклу складається з кількох таких дій.

Оператор дуже простий і зрозумілий. Схема алгоритму оператора повторення з передумовою не викличе у вас ніяких непорозумінь

Приклад 1 програми з оператором *While*

Обчислити суму S=1+1/2+1/3+…+1/50, використовуючи оператор While

```
Program Example_1 ;
VAR S: REAL; N:INTEGER;
BEGIN
S:=0; N:=1;
WHILE N<=50 DO
   BEGIN
       S:=S+1/N;
  N:=N+1;
   END;
WRITELN(' S=',S);
END.
```

Приклад 2 *Дано натуральне число N. Визначити кількість цифру числі.*

```
 Program Example_2 ;
Uses crt;
Var N: longint; Counter:  integer;
Begin
Clrscr;
Write ('Введіть число: '); Readln(N);
Counter := 0;
While N > 0 do
Begin
Counter:=Counter+1;     {Підрахунок кількості цифр)
N:=N div 10; {Відкидання останньої цифри}
End;
Writeln('Кількість цифр у заданому числі дорівнює', N);
Readkey;
End.
```

Загальний вигляд оператора циклу з післяумовою:

repeat P until <логічний вираз>;

де значення всіх параметрів такі самі, як і в попередньому операторі.

Цикл з післяумовою працює таким чином: *повторення оператора* **P** *відбувається доти, поки логічний вираз отримує значення* **false.** ***Якщо тільки на деякому кроці циклу****логічний вираз набуде значення* **true,** *цикл завершить свою роботу.*

Чи відчули ви відмінність між операторами **while ... do** та **repeat ... until?** Перший оператор спочатку перевіряє значення логічного виразу, а потім вирішує, виконувати йому оператор, чи ні, а другий, навпаки, спочатку виконує вказаний оператор, а потім перевіряє, чи треба його виконувати ще раз.

Розглянемо приклад: Обчислити суму S=1+1/2+1/3+…+1/50, використовуючи оператор Repeat

```
 Program Example_3 ;
VAR S: REAL; N:INTEGER;
BEGIN
S:=0; N:=1;
REPEAT
       S:=S+1/N;
        N:=N+1;
UNTIL N>50;
WRITELN(' S=',S);
END.
```

Ще один приклад застосування оператора Repeat

Перевірка коректності введення. Дано три числа, що задають міри кутів трикутника. Визначити, чи можна побудувати трикутник, що має задані кути. Якщо ні, користувач має ввести інші дані.

```
Program Example_4 ;
Uses  crt;
Var a,b,c:   integer;
Begin
Clrscr;
Repeat
Write('Введіть міри кутів трикутника:   ' ) ;
Readln(a,b,c);
Until  (a>0)and(b>0)and(c>0)and(a+b+c)=180;
End.
```

Загальний вигляд оператора циклу з параметром:

for <параметр циклу> := $X_{поч}$ **to (downto)** $X_{кінц}$ **do** *P,*

де параметр циклу - змінна зчисленого типу, $X_{поч}$ - початкове значення параметра циклу, $X_{кінц}$ - кінцеве значення параметра циклу, *P* - простий чи складений оператор.

Службове слово **to** означає, що зміна значення параметра циклу йде від $X_{поч}$ до $X_{кінц}$ в порядку збільшення, а **downto** - в порядку зменшення.

Приклад програми з оператором *For*

Знайти суму всіх натуральних чисел від 1 до 100.

```
Program Example_5;
 Uses crt;
Var Sum,   i:   integer;
Begin
Clrscr;
Sum:= 0;
 For i:= 1  to  100 do  Sum:» Sum + i;
Writeln('Sum =',  Sum);  Readkey;
End.
```

Розглянемо ще один приклад: обчислити суму S=1+1/2+1/3+…+1/50, використовуючи оператор For

```
Program Example_6;
VAR S: REAL; N:INTEGER;
BEGIN
S:=0;
FOR I:=1 TO 50 DO
     S:=S+1/I;
WRITELN(' S=',S);
END.
```

Підсумок уроку.

Запитання для перевірки засвоєння знань

1. Що називається циклом?
2. Які типи циклів вам відомі?

Домашнє завдання.

Вивчити теоретичний матеріал уроку.

1. Скласти програму, яка виводить на екран 50 чисел. Кожне число в окремому рядку.
2. Вивести на екран квадрати чисел від 1 до 10. Скласти програму в 3-х варіантах, використовуючи різні оператори циклу.

Урок 40. Цикли з параметром (1)

Мета уроку: формування навичок використання циклу з параметром для розв'язування типових задач; виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

I. Актуалізація опорних знань учнів.

Інформаційний диктант.

1. Обчислювальний процес називається циклічним, якщо…
2. Цикл з передумовою призначений для… записується так… виконується…
3. Цикл з післяумовою призначений для… записується так… виконується…
4. Цикл з параметром призначений для… записується так… виконується…
5. Описати різницю між трьома видами циклів.

II. Формування практичних навичок.

1. Побудова таблиць. Розглянемо задачу: вивести 11 екран таблицю квадратів і кубів чисел від 20 до 3О

```
20      400     8000
21      441     9261
22      484     10648
…
30      900     27000
```

Цю задачу розв'язують за допомогою циклічного алгоритму, яки можна утворити засобами команд **if** і **goto** так:

```
label dali, stop; var i:integer;
begin
i := 20;
dali: if i > 30 then goto stop;
writeln(i:6, i*i:7, i*i*i:8);
i := i+1;
goto dali;
stop: end.
```

Виявляється, що задачу можна розв'язати значно простіше за допомогою команди **for:**

```
var i: integer; {Це готова до виконання програма}
begin
for i := 20 to 30 do writeln(i:6, i*i:7, i*i*i:8);
end.
```

Задача 2. Один долар коштує 5,43 гривні. Вивести у вигляді таблиці вартість 1, 2,..., 10 доларів.

```
program Bank;
var d: integer; gr : real;
begin
writeln('Долари      Гривні');
for d := 1 to 10 do begin
gr := 5.43* d; writeln(d:4, gr:15:2)
end; readln
end.
```

На екрані отримаємо таку таблицю:

Долари	Гривні
1	5.43
2	10.86
3	16.29
4	21.72
5	27.15
6	32.58

```
 7    38.01
 8    43.44
 9    48.87
10    54.30
```

Завдання 2. Виведіть таку ж таблицю для євро чи іншої валюти.

2. Обчислення елементів послідовності. Розглянемо випадок, коли елементи числової послідовності описуються формулою, нехай, a_i =3+cos2i.

Задача. Обчислити та вивести на екран номери і значення перших n=10 елементів.
Це можна зробити за допомогою наступної програми:

```
program Elements;
var i,n: integer; a : real;
begin
n := 10;
for i := 1 to n do begin
a := 3 + cos(2*i); writeln(i:4, a:15:2)
end
end.
```

3. Пошук потрібних елементів. Задача пошуку потрібних елементів з-поміж усіх елементів послідовності розв'язують ***методом перегляду і аналізу всіх елементів.*** Для відбору потрібних елементів використовують деяку умову (логічний вираз). Усе це роблять за допомогою команди **if,** яка знаходиться в тілі команди циклу.

З а д а ч а 3. Нехай елементи числової послідовності описуються формулою a=2-2cos3i, *i* = 1,2,...,12. Вивести на екран номери і значення лише додатних елементів.

```
program Find;
var i,n: integer; a : real;
begin
n := 12;
for i := 1 to n do begin
a := 2 - 2*cos(3*i);
if a > 0 then writeln(i:4, a:15:2)
end
        end.
```

Завдання. Виведіть на екран елементи, які більші, ніж 1 і менші, ніж 2.

Довідка 1. Якщо s — змінна типу char, то вивести на екран усі символи латинського алфавіту можна так: **for** S := 'A' to 'Z' **do** write(S);

Довідка 2. якщо крок зміни параметра циклу дорівнює -1, то форма написання команди циклу «для»має вигляд:

for <параметр> := <вираз 1> **downto** <вираз 2> **do** <команда>.

Наприклад, вивести на екран числа від 1 до **10** у зворотньому порядку: 10 9 8 ... 1 - можна так: **for** i := **10 downto** 1 **do** write(i:3).

Закріплення матеріалу.

1. Що таке цикли?
2. Який вигляд має команда циклу **for?**
3. Що таке параметр циклу?
4. Опишіть дію команди циклу **for.**
5. Як може змінюватися значення параметра в команді циклу **for?**
6. Чим відрізняється команда **for-to** від команди **for-downto?**

Вправи та задачі для самостійного розв'язання.

1. Визначте результати виконання таких команд (усно):

а) a:=5; **for** i:=l to 2 **do** a:=a*i-2; a:=a+l (відповідь: a=5);
б) **a:=1; for** i:=l **to** 3 **do begin** a:=a+i; a:=a-l **end;**
в) a:=0; **for** i:=l to 4 **do** a:=a+i; a:=a+2;
г) p:=1; **for** b:=8 **downto** 5 **do** p:=p+b; p:=p+l;
д) s:=0; **for** n:=7 **downto 4 do begin** s:=s+n; s:=s+l **end;**

Складіть програми до наведених задач.

1. Для чисел від 1 до 10 обчисліть квадратні корені, кубічні корені та корені четвертого степеня. Результати наведіть у вигляді таблиці.
2. Виведіть на екран десять рядків таблиці відповідності між двома наступними мірами, знаючи що а) 1 фунт = 0,45359237 кг; б) 1 стоун 6,35029 кг; в) 1 унція = 28,353495 г; г) 1 драхм = 1,77185 г; д) 1 карат = 0,2 г; є) 1 гран - 0,068 г.
3. Виведіть таблицю переведення температури з градусів за шкалою Цельсія (C) у градуси за шкалою Фаренгейта (F) для значень градусів від a до b з кроком 1 за формулою F=1,8C+32.
4. Див. умову задачі № 6. Переведіть градуси за шкалою Фаренгейта у градуси за шкалою Цельсія.
5. Виведіть на екран 15 перших елементів числової послідовності, починаючи з першого, загальний елемент якої має вигляд 4 - 2sin2i.

Підсумок уроку.

Домашнє завдання.

2. Визначте результати виконання таких команд:

а) a:=4; **for** i:=1 to 2 **do** a:=a*i-1; a:=a+2;

б) a: = **1; for** i:=1 **to** 3 **do begin** a:=a+2*i; a:=a-2 **end;**

в) a:=-1; **for** i:=1 to 4 **do** a:=2*a+i; a:=a+2;

г) p:=30; **for** b:=7 **downto** 4 **do** p:=p-b; p:=p+5;

д) s:=0; **for** n:=6 **downto 3 do begin** s:=s+2*n; s:=s-1 **end;**

3. Тіло вільно падає з висоти *h.* Виведіть таблицю значень висоти протягом перших *k* секунд падіння.

Урок 41. Цикли з параметром (2)

Мета уроку: формування навичок використання циклу з параметром для розв'язування типових задач; виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

I. Актуалізація опорних знань учнів.

Дайте відповіді на запитання:

1. Що називається циклом?
2. Які типи циклів ви знаєте?
3. В чому особливість циклу з параметром?

II. Формування практичних навичок.

Розв'язування задач.

Задача № 183

Компанія бабусь поїхала на мотоциклах на курси комп'ютерної грамотності. Попереду на мотоциклі без глушника їхала одна бабуся, за нею — дві, потім — три і т.д. Скільки бабусь їхало на заняття, якщо приголомшені пішоходи всього нарахували N рядів? Чи змогли бабусі зайняти всі місця у класі, якщо там стояло k рядів по L комп'ютерів у кожному? Скільки вільних місць залишилося?

Розв'язання: Зверніть увагу на те, що фактично ця задача зводиться до знаходження суми всіх натуральних чисел від 1 до *N.* У кінці задачі для повторення команди розгалуження учням пропонується визначити кількість зайнятих бабусями та вільних місць. Програма розв'язання даної задачі має такий вигляд:

```
Program Example_183;
 Uses  crt;
Var i,N,Sum:word;  {i - параметр циклу, N - хількість рядів мотоциклів,  Sum - кількість бабусь, що приїхали на курси}
Place,k,L:word; (k - кількість рядів у комп'ютерному класі, L - кількість комп'ютерів у кожному ряду. Place - кількість місць, якої вистачило для бабусь}
```

```
Begin
Clrscr;
Sum:=0;
Write('Введіть кількість рядів мотоциклів: ');
Readln(N);
For i:=1 to N do Sum:=Sum+i;
Writeln('Кількість бабусь, що приїхали на курси ' ,Sum);
Writeln('Кількість комп''ютерів на курсах ',к*L);
If Sun<k*L
Then writeln('Бабусі не змогли зайняти всі місця.')
Else writeln( 'Бабусі зайняли всі місця.') ;
Place:=Sum - к*L;
If P1ace>0
Then writeln ('Бабусям не вистачило ' , Place,' місць.') ;
Readkey;
End.
```

Зауваження. У будь-якій програмі цикл **for** можна замінити рівносильним циклом **while,** але не навпаки. Тому цикл **while** вважається більш універсальним.

Порада. Якщо вам наперед відома кількість повторень оператора циклу, то доцільніше використовувати цикл із параметром. Якщо кількість повторень залежить від виконання деякої умови і постановка задачі передбачає обов'язкове виконання циклу хоча б один раз, то в цьому випадку рекомендується лише цикл з післяумовою. Якщо ж наперед невідома кількість повторень циклу, і, за умовою задачі, можливе невиконання циклу жодного разу, то прогноз єдиний - вам підходить лише цикл з передумовою! У циклах з перед/післяумовою створені вами умови повинні містити хоча б одну змінну величину, значення якої змінюється в тілі цього циклу. Інакше ви маєте нагоду створити «вічний двигун».

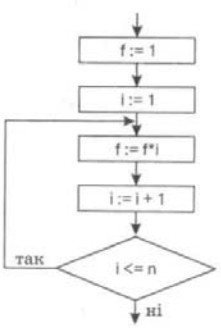
Розглянемо такий приклад. У математиці існує поняття факторіала. Факторіал обчислюється за такою формулою:

n! = 1*2*3*4*5* ... **n*. Тобто n! - добуток усіх натуральних чисел від 1 до *n.*

```
program factorial;
var i, f: integer;
begin
f:=1;
write ('Задайте значення числа n: ');
readln (n);
for i := 1 to n do f :=f*i;
writeln ('Значення факторіала числа ', n:5, 'дорівнює: ', f: 10)
end.
```

Накопичення добутку відрізняється від накопичення суми тим, що початкове значення обов'язково повинне дорівнювати 1!

Схема алгоритму цієї програми зображена на малюнку .



Якщо ми лише незначним чином змінимо нашу програму, ввівши виведення результату в тіло циклу, то отримаємо зовсім інший ефект. Результатом її виконання буде виведення значень факторіала на кожному кроці. Такий принцип називається *по-кроковим виведенням* результатів.

```
for i := 1 to n do begin
f := f*i;
write ('Значення факторіала числа ', i:5);
writeln (' дорівнює: ', f:10)
end;
```

Як розібратися, коли який оператор циклу використовувати? Саме в цьому й полягає майстерність програміста - оцінити постановку задачі, скласти алгоритм, залучити всі свої аналітичні здібності й остаточно вибрати оптимальний варіант циклічного оператора.

Розглянемо задачу: Дана послідовність з N чисел. Визначити середнє арифметичне непарних від'ємних чисел, добуток чисел, кратних 5 із проміжку [-20,20], кількість нульових чисел и замінити числа, менші ніж 3, на 100.

```
Program prim;
Const N=20;
Var Sr :real; S, kol, kolo, i, N, p, x: integer;
 Begin
S:=0; p:=1; kol:=0; kolo:=0;
 For i:=1 to n do
Begin
Writeln('введіть число последовності'); Readln(x);
If (x mod 2 <> 0) and (x<0) then Begin S:=s+x;  Kolo:=kolo+1; End;
If (x mod 5 =0) and (x >=-20) and (x <20) then p:=p*x;
If x=0 then kol:=kol+1;
Ifx<3 then Begin x:=100; Writeln('число змінилося' ,x); End;
End;
Sr:=s/kolo; Writeln('середнє=', sr:8:4, 'добуток = ', p, 'кількість нульовых чисел=', kol);
End.
```

Розглянемо задачу: Вивести на екран таблицю множення чисел (таблицю Піфагора)

```
Program Pifagor;
Var j, i: integer;
Begin
For i:=1 to 9 do
Begin
For j:=1 to 9 do
Write(i*j:3);
Writeln;
End;
Readln
End.
```

Підсумок уроку.

Домашнє завдання.

Розв'язати задачі:

1. Виведіть на екран у вигляді таблиці номери і значення перших десяти елементів числової послідовності, загальний елемент якої має вигляд $7 - 5\sin i^2$.
2. Серед перших 20 елементів числової послідовності $3 - 3\sin i^2$ виведіть на екран номери і значення лише від'ємних елементів, тут $i = 1,2,...20$.
3. Розгляньте елементи числової послідовності 5 - 3cos2i від 10-го до 20 го і виберіть серед них (виведіть їхні номери на екран) більші, ніж 4.
4. Перші десять елементів (i=1,2,...10) числової послідовності 2 — 2|sin3i| перетворіть за таким правилом: додатні елементи збільшить у два рази, від'ємні елементи зменшить на 2. Виведіть у вигляді таблиці номери елементів, їхні попередні та нові значення.

Урок 42. Обчислення суми, добутку та кількості.

Мета: формувати практичні навички та вміння розв'язувати задачі на обчислення суми, добутку та кількості.

Тип уроку: формування навичок.

Хід уроку.

І. Актуалізація опорних знань учнів.

Фронтальне опитування.

1. Що таке цикли?
2. Який вигляд має команда циклу for?

3. Що таке параметр циклу?
4. Опишіть дію команди циклу for.
5. Як може змінюватись значення параметра в команді циклу for?
6. Чим відрізняється команда for-to від команди for-downto?

II. Формування практичних навичок учнів.

1. Обчислення суми. Під час обчислення суми початкове значення змінної, де нагромаджуватиметься сума, наприклад s, має дорівнювати нулеві. Для цього використовують команду присвоєння s := 0.

З а д а ч а 1. Скласти програму для обчислення суми цілих чисел від 1 до **100.**

```
program Suma1;
var s, number : integer;
begin
s := 0;
for number := 1 to 100
do
s := s + number;
writeln("сума = ', s:5)
end.
```

Змінна number отримуватиме такі значення: 1, 2, 3, ...,100, а змінна s - такі: 0, 1, 3, 6, 10, 15, 21,…, 5050. *Відповідь:* 5050.

Завдання 1. Обчисліть суму чисел від **100** до **200.**

2. Обчислення добутку. Під час обчислення добутку початкове значення змінної, де нагромаджуватиметься добуток, наприклад d, має дорівнювати одиниці (d := 1).

З а д а ч а 2. Обчислити добуток перших п'яти елементів послідовності, які задані виглядом загального елемента $a = 2 + |\sin 3i|$, i=1,2,...,5.

```
program dobutok;
var i, n : integer; a, d : real;
begin
        d := 1; n :=5;
        for i := 1 to n do begin
    a := 2 + abs(sin(3*i)); d := d *a
    end;
    writeln('добуток =', d:10:2); readln end.
```

Завдання 2. Обчисліть добуток елементів більших, ніж 2,5.

3. Обчислення кількості. Під час обчислення кількості початкове значення змінної, де нагромаджуватиметься кількість, наприклад к, має дорівнювати нулеві (k := 0), а в тілі циклу має бути команда k:=k+l.

З а д а ч а 3. Скільки елементів послідовності $a_i=1-\cos i$, де *i* = 1, 2,...20, задовольняють умову $0<a_i<1$?

```
program Kilkist;
var i, n, k : integer; d : real;
begin
    k := 0; n :=20;
    for i ;= 1 to n do
        begin  a := 1 - cos(i);
qif (a>0) and (a<l) then k := k + 1
end;
 writeln('кількість=', k:2); readln
end.
```

Підсумок уроку.

Домашнє завдання.

1. Обчисліть суму додатних елементів числової послідовності 3sin2i, де i=1,2,...,15.
2. Визначіть скільки серед елементів послідовності cos3i, i= 1,2,...,12, є від'ємних.
3. Обчисліть добуток елементів послідовності 2-sin2i, i=1,2,...10, значення яких є більші, ніж 1 і менші, ніж 2.
5. Обчисліть суму від'ємних значень елементів послідовності 1-2cos3i, i=1,2,…,20.
6. У якій з послідовностей більше додатних елементів: 1) 2sin3i, i=1,2,...,10 чи 2) 2-3cosi, i=1,2,...,15?
7. Обчисліть z=sin1+sin2+sin3+...+sinn, де n=30.
8. Обчисліть $y=x+x^3/3+x^5/5+...x^{21}/21$, де x=0,5.

Урок 43. Методи перебирання варіантів.

1. Аналіз чисел. Розглянемо задачу виведення на екран лише тих чисел, які володіють деякою властивістю.

Задача1 (Аналіз чотиризначних чисел). Визначити всі чотиризначні числа, сума цифр яких дорівнює їхньому добутку.

Задачу розв'язують ***методом повного*** *перебирання* усіх можливих варіантів з використанням алгоритмічної конструкції «вкладені цикли **for»**.

Позначимо чотири цифри шуканих чисел як i, j, k, m, а їхню кількість — s. Наступна програма дає розв'язок задачі:

```
program Numbers;
var i,j,k,m,s,a : integer;
 begin
s := 0;
for i := 1 to 9 do
for j := 0 to 9 do
 for k := 0 to 9 do
for m := 0 to 9 do
if i+j+k+m = i*j*k*m then b
egin
s:= s+1;
a := 1000*i + 100*j + 10*k + m;
writeln(a)
end;
writeln('усього чисел є ', s); readln
end.
```

Завдання 1. Виконайте програму і переконайтеся, що таких чисел є 12 серед 9000: 1124, 1142, ..., 4211. Розв'яжіть цю задачу для тризначних чисел.

2. **Задача про решту.** Застосуємо метод перебирання варіантів для розв'язування задачі про видачу решти.

З а д а ч а 2 (про решту). У касі є монети номіналом 3, 5, 10. Скількома способами касир може видати покупцеві решту на деяку суму Num?

Задачу розв'язуємо шляхом перебирання можливих варіантів і вибирання з-поміж них потрібних способів видачі решти. Тут, щоб обмежити перебирання варіантів, уведемо величини Xmax, Ymax, Zmax — максимальні кількості монет для видачі решти монетами одного номіналу. Введемо також величину p — ознаку того, що задача має розв'язок. Програма матиме вигляд

```
program Change; uses Crt;
var Num.Xmax, Ymax, Zmax, x,y,z, p:integer;
begin
clrscr;
write(' У ведіть суму: ');
readln(Num);
P :=0;
```

```
Xmax := Num div 3;
Ymax := Num div 5;
Zmax := Num div 10;
writeln('  3  5  10 Усього монет');
writeln('.....');
for x := 0 to Xmax do
for y := 0 to Ymax do
for z := 0 to Zmax  do
if 3*x+5*y+10*z = Num then
begin
 writeln(x:3, y:3, z:3, x+y+z:8);
p :=p+1
end;
writeln;
if p = 0 then writeln('Немає варіантів')
else writeln('Усього способів є ', p); readln
end.
```

Виконаємо програму для решти 23. Результати будуть такі:

```
Уведіть число? 23
3   5 10 Усього монет
1   0   2         3
1   2   1         4
1   4   0         5
6   1   0         7
Усього способів є 4
```

Завдання 2. Поекспериментуйте з програмою для різних значень решти. Модифікуйте програму для номіналів монет: 1**,** 2, 5, 10.

3. Задача про високосні роки. Високосний рік має 366 днів, а звичайний — 365. Високосним є рік, значення якого ділиться на 4, не ділиться на 100, але ділиться на 400.

З а д ач а 3. Визначити, скільки днів було в двадцятому столітті.

```
program AboutDays;
var i: 1901. .2000;
Vysokosnyi : boolean; KilDniv : longint;
begin
KilDniv := 0; for i :- 1901 to 2000 do
begin
Vysokosnyi := (i mod 4 = 0) and (i mod 100<> 0)
or (i mod 400 = 0);
if Vysokosnyi then KilDniv :=KilDniv + 366
else KilDniv := KilDniv + 365
end;
 writeln('Було ', KilDniv, ' днів');
writeln ('Бажаємо успіхів у XXI столітті')
end.
```

Завдання 3. Виконайте програму і поекспериментуйте з нею. Скільки днів було у XVI столітті, скільки днів буде у XXI?

Підсумок уроку.

Домашнє завдання.

1. Обчисліть s: s:=0; **for** i:=1 to 2 do **for** j:=1 to 3 do s:=s+i*j.
2. Уведіть чотиризначне число, наприклад, 1998. Обчисліть суму його цифр. Підказка: використайте операції **div** і **mod.**
3. Виведіть на екран усі двозначні числа, які діляться на 3 або на 5.
4. Визначіть, скільки є «щасливих» автомобільних номерів в одній серії з комарами від 0000 до 9999. Номер є щасливим, якщо сума першої половини цифр дорівнює сумі другої.
5. Визначіть, скільки є «щасливих» автобусних квитків в одній серії з номерами від 000000 до 999999.

6. Визначіть цифри *a, b,* c, які задовольняють рівняння *abb + cab = bac.* Дослідіть можливість скорочення повного перебору.
7. Визначіть кількість тризначних чисел, сума цифр у яких дорівнює деякому заданому числу *n.* Виведіть їх на екран. Використовуйте лише два вкладені цикли.
8. Скільки днів пройшло від початку року до деякої заданої дати?
9. Скільки днів пройшло від вашого дня народження?

Урок 44. Цикли з передумовою (1).

Мета: формування навичок використання циклу з передумовою для розв'язування типових задач; виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

І. Актуалізація опорних знань учнів.

На початку уроку рекомендується провести письмове опитування (можна у вигляді диктанту) по матеріалах попереднього уроку. Далі можна розглянути деякі типові задачі з використанням циклу з передумовою. Нагадуємо, що в усіх цих задачах кількість повторень буде визначатись залежно від початкових та кінцевих умов.

Приклад1. Вивести у вигляді таблиці числа від 20 до 30 та їхні квадрати і куби за допомогою команди **while** можна так:

```
i := 20;
while i <= 30 do
begin
writeln(i:4, i*i:6, i*i*i:8);
i :=i + 1
end;
```

Особливість команди **while.** У циклі «доки» на відміну від циклу «для» початкове значення параметра треба задавати "вручну" перед командою циклу (команда i:=20). Крім цього в тілі циклу треба записати команду зміни значення параметра (i := i + 1).

Зауваження. У будь-якій програмі цикл **for** можна замінити рівносильним циклом **while,** але не навпаки. Тому цикл **while** вважається більш універсальним, ніж цикл for.

З а д а ч а 1. Обчислити суму перших ста натуральних чисел, використовуючи цикл **while.**

Розгляньте схему алгоритму на рис. 13 і програму Suma2, в якій вперше замість команди присвоєння типу number := number + 1 застосовано рівносильну команду inc(number), що є в наборі команд середовища Turbo Pascal for DOS.

Програма Suma2 має такий вигляд:

```
program Suma2;
var number,s : integer;
begin
s := 0;
number := 1;
while number <= 100 do
begin
s := s + number; inc (number) end;
writeln (s);
end.
```

2. Табулювання функції. Табулювання функції — це побудова таблиці значень функції ***y*** = *f(x)* для різних значень аргументу *x,* де *x* змінюється на деякому проміжку *[a; b]* з кроком *h.*

З а д а ч а 2. Протабулювати функцію ***y*** = xsinx на проміжку [1; 2] з кроком *h* = 0,1.

```
program Tabul;
var x, a, b, h, y : real;
begin
write('Введіть a, b, h (тут 1  2  0.1):  ');
```

```
readln(a, b, h);
writeln('   x    y=xsinx  ');
x := a;
while x <= b + h/2 do begin
y := x * sin(x);
writeln(x:6:l, y:9:2);
x := x + h
end
end.
```

Умова x <=b не завжди забезпечує попадання в останню точку x=*b* через особливості машинної арифметики. Тому тут використано умову x<= b + h/2. Виконавши алгоритм, отримаємо таблицю значень заданої функції:

x	y=xsinx
1.0	0.84
1.1	0.98
1.2	1.12
1.3	1.25
1.4	1.38
1.5	1.50
1.6	1.60
1.7	1.69
1.8	1.75
1.9	1.80
2.0	1.82

3. Цикли з наперед невідомою кількістю повторень. Цикли із заздалегідь невідомою кількістю повторень не можна записати за допомогою команди **для.** їх треба записувати за допомогою циклу «доки». Ось чому ***цикл «доки» є більш універсальний, ніж цикл «для».***

З а д а ч а 3. Визначити дійсне додатне число *а,* для якого виконується співвідношення a/2 = 0 у комп'ютерній арифметиці дійсних чисел. Таке число характеризує нижню додатню межу типу даних **real.**

```
program MinRealNumber;
var a : real;
begin
a:=1;
while a/2 > 0 do a := a / 2;
writeln('a =', a)
end.
```

Відповідь: a - 2.9E-39.

Виконайте вправи :

1.Якого значення набуде змінна після виконання команд:
а) p:=4; while p<10 do p:=2*p+1; p:=p+1 (відповідь: p = 20);
б) p:=4; while p<10 do begin p:=2*p+1; p:=p+1 end;
в) p:=5; while p>2 do p:=(p-3)*2; p:=p-3;
2.Якого значення набуде змінна після виконання команд:
а) p:=7; while p>=5 do p:=(p+3)/2; p:=p-1;
б) p:=7; while p>=5 do begin p:=(p+3)/2; p:=p-1 end;

Підсумок уроку.

Домашнє завдання.

Дайте відповідь на запитання:

1. Яке призначення циклу «доки»?
2. Який загальний вигляд має команда циклу **while?**
3. Чому цикл «доки» вважають більш універсальним, ніж цикл «для»?

1. Визначте суму парних чисел від 1 до 100.

2. Обчисліть добуток непарних чисел від 1 до 13. ,
3. Протабулюйте функцію cos2x: на проміжку [-2;2] з кроком 0,25 і обчисліть суму додатних значень функції.
4. Протабулюйте функцію sin3x на проміжку [-1;1] з кроком 0,2 і обчисліть кількість від'ємних значень.

Урок 45. Цикли з передумовою (2).

Мета: формування навичок використання циклу з передумовою для розв'язування типових задач; виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

I. Актуалізація опорних знань учнів.

II. Формування навичок.

1. Пошук значень функції, які володіють деякою властивістю

Розглянемо задачу, де потрібно протабулювати функцію і проаналізувати її значення.

З а д а ч а 1 . Протабулювати функцію ***y*** = sinx на проміжку [0; π], з кроком *h* = 0,1 і обчислити середнє арифметичне значень функції більших, ніж 0,1 і менших, ніж 0,6.

Оскільки крок зміни параметра циклу дробове число, використаємо цикл **while.**

```
program FindSerednie;
var x, y, s, si, h, xk: real; n : integer;
begin
x := 0; xk := pi; h := 0.1; s :=0; n:=0;
while x <= xk + h/2 do begin
y := sin(x) ;
writeln(x:3:1, y:6:2);
if (y > 0.1) and (y<0.6) then
begin s := s + y; n := n + 1 end;
x := x + h ;
end;
if n > 0 then
begin si := s / n; writeln ('Середнє значення =', s1) end
else writeln('Таких значень немає n=0')
end.
```

Завдання. Визначіть кількість від'ємних і кількість додатних значень функції, якщо крок *h* = 0,05.

2. Пошук максимального чи мінімального значення. Щоб серед багатьох даних визначити максимальне (чи мінімальне), потрібно спочатку перше дане прийняти за шукане і порівняти його з наступним Якщо наступне дане більше (менше), то взяти його за шукано порівняти з тими даними, що залишилися і т.і.

З *а д а ч а* 2. На Олімпійських іграх у фіналі 5 спортсменів (men) змагаються у стрибках. Скласти програму, яка потребуватиме ввести номер спортсмена (n), результат чергового стрибка (rez) і виведе найкращий (максимальний) результат (rezmax) і номер спортсмена- переможця (nchemp).

```
program Compet;
const men = 5;
var n, nchemp, sportsman : integer;
rez, rezmax: real;
begin
rezmax := 0; nchemp :=0;
for sportsman := 1 to men do
begin
write('Введіть номер спортсмена і його результат:'); readln(n, rez);
if rez>rezmax then begin rezmax := rez; nchemp := n end; end;
writeln('Переможцем став', nchemp);
```

```
writeln('Найкращий результат ', rezmax); readln
end.
```

З а д а ч а 3. Протабулювати функцію *y* = sinx на проміжку [0; тс], кроком *h* = 0,1 і визначити її мінімальне значення (ymin) і точку чиїм), в якій воно досягається.

```
program FindMinimum;
var x, y, ymin, xmin, h, xk: real; n : integer;
begin
x := 0; xk := pi; h := 0.1;
ymim := sin(x);
xmin := x;
while x <= xk + h/2 do
begin
y := sin(x) ;
writeln(x:3:1, y:9:2);
if y < ymin then
begin ymin := y;   xmin := x end;
x := x + h ;
end;
writeln('ymin= ', ymin:5:3, ' в точці xmin= ', xmin:3:l)
end.
```

Завдання. Визначіть максимальне значення функції.

3. Задача про найбільший спільний дільник. *Найбільший спільний дільник (НСД)* двох чисел *a* і *b* визначають за допомогою алгоритму Евкліда. Алгоритм ґрунтується на таких властивостях цілих чисел: якщо *a* > *b,* то НСД(a, *b)* = НСД(a-b, *b);* якщо a < *b,* то НСД(a, *b)*=НСД(a, *b-a);* якщо a = *b,* то НСД(a, *b)* = *a.* Застосуємо цю властивість у циклі. Розглянемо програму Evklid.

```
program Evklid;
var a,b : integer;
 begin
Write('Введіть два числа:');
readln(a.b);
while not (a=b) do
if a > b then a :- a - b else b := b - a;
writeln (a); readln
end.
```

Виконаємо трасування програми для чисел 18 і 24: НСД(18, 24) =НСД(18, 6) = НСД(12, 6) = НСД(6, 6) = 6.

4. Задача про прості числа. Розглянемо задачу, для розв'язування якої застосуємо додаткову змінну-прапорець, значення якої сигналізує про досягнення мети.

З а д а ч а 4. Визначити, чи задане натуральне число *а* є простим.

Розглянемо програму SimpleNumber. Тут спочатку перевіряється, чи число парне, а далі — чи воно ділиться без остачі на менші від нього непарні числа.Якщо так, то число не є простим, тому аналіз закінчують достроковим виходом з циклу «поки». Якщо число не ділиться на жодне менше непарне число, то робимо висновок: число просте.

```
program SimpleNumber;
var a, i : integer; flag : boolean;
begin
 write('Уведіть число '); readln(a);
flag := true;   { це ознака простого числа }
if a mod 2 = 0 then flag := false;
i := 3;    {тепер аналізуємо ділення на непарні числа}
while (i < a) and flag do
```

```
if a mod i = 0 then flag :=false
else i := i+2;
if flag then writeln('Число просте')
else writeln('Число не просте')
end.
```

Завдання. Переконайтеся, що в заданому алгоритмі перебір варіантів можна скоротити, замінивши у циклі **while** логічний вираз i < a виразом i <= sqrt(a).

Підсумок уроку.

Домашнє завдання

1. Протабулюйте функцію y=cos2x на проміжку [-2;2] з кроком 0,25 і обчисліть середнє арифметичне від'ємних значень.
2. Протабулюйте функцію y=2sin2x на проміжку [-2;2] з кроком 0,25 і визначіть: а) максимальне значення; б) мінімальне значення.
3. Протабулюйте функцію y=xcos2x: на проміжку [-2;2] з кроком 0,2 і визначіть максимальне значення функції та значення аргументу, для якого воно досягається.
4. Протабулюйте функцію y=sinxcos2x на проміжку [-4;4] з кроком 0,5 і обчисліть суму квадратів максимального і мінімального значень функції.
5. Числа *a, b, c* називаються «числами Піфагора», якщо $a^2+b^2=c^2$. Визначіть *n* наборів «чисел Піфагора».

Урок 46. Цикли з післяумовою.

Мета: формування навичок використання циклів з післяумовою для розв'язування типових задач; виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

I. Актуалізація опорних знань учнів.

Фронтальне опитування.

1. Які типи циклів вам відомі?
2. В чому полягає особливість використання того чи іншого типу циклів?
3. Як уникнути «нескінченного» циклу?

II. Формування навичок.

Задача 179

На дверях ліфта висіло загрозливе попередження про те, що двері самі зачиняються в той самий момент, коли зайвий за вагою пасажир переступить поріг ліфта. Котрий пасажир постраждає, якщо ліфт витримує вагу не більше S кг, а вага пасажирів, що стоять у черзі до ліфта, дорівнює відповідно $a_p a_y a_y$ *... а ?*

Розв'язання: У цій задачі зручніше використовувати цикл з післяумовою, тому що спочатку необхідно дати можливість «увійти» пасажиру в ліфт, а потім перевіряти, чи витримає його ліфт. Умовою виходу з циклу буде перевищення сумарної ваги пасажирів, що увійшли в ліфт, деякого заданого критичного значення. Для зберігання ваги чергового пасажира в цій задачі ми будемо використовувати одну й ту саму змінну *(A),* оскільки після перевірки вага пасажира нас уже не цікавить. Програма має вигляд:

```
Program Example_179;
Uses crt;
Var N:word;   {N - номер пасажира,   що увійшов у ліфт}
Sum,A,S:real;   {Sum - сумарна вага пасажирів, що знаходяться в ліфті, A - вага чергового пасажира, що увійшов до ліфта, S - критична вага,  що може бути піднята ліфтом}
Begin Clrscr;
Sum:=0;
N:=0;  {На початку роботи програми в ліфті немає пасажирів}
Write('Введіть  критичну вагу,   що піднімає ліфт:   '); Readln(S) ;
Repeat
Write(*Введіть вагу чергового пасажира:   '); Readln(A);
Sum:=Sum+A; N:=N+1;
Until  Sum>S;
Writeln('Постраждає ' ,N,'-й пасажир.');
Readkey;
```

```
End.
```

Задача 181

Умова: Капосний папуга навчився висмикувати у дідуся Василя волосся, яке ще залишилося у того на голові. Почавши з однієї волосини, він щодня збільшував порцію вдвічі. Через скільки днів дідусеві не знадобиться гребінець, якщо спочатку в нього на голові було аж N волосин?

Розв'язання: Аналогічно до попередньої задачі, аналізувати наявність волосся на голові слід після того, як папуга вже висмикнув чергову порцію волосся. А «знущання» над дідусем скінчиться тоді, коли гребінець йому стане непотрібним, тобто кількість волосся на голові дорівнюватиме нулю.

Зверніть увагу, що в цій задачі змінна *S* використовується для підрахунку чергової порції волосся, що підлягає висмикуванню капосним папугою.

```
Program Example_181;
Uses crt;
Var S,N,Sum:longint;  {S - кількість волосся, що буде висмикнутим,  Sum - кількість волосся,
що залишилося в дідуся на голові,  N - початкова кількість  волосся}
 Day:word; {Day -  номер дня,  який папуга знущається над дідусем}
Begin
Clrscr;
Write('Початкова кількість волосся а дідуся на голові:  ') ;
Readln(N) ;
If N=0 Then writeln('Дідусь  уже лисий,  папузі  нічого робити!')
Else
begin
Day:=0;   Sum:=N;   S:=1;
{Початкова кількість волосся, що буде висмикнуте папугою}
Repeat
Sum:=Sum-S;  {Зменшення дідусевого волосся}
S:=S*2;
Day:=Day+1;  {Підрахунок номеру дня)
Until  Sum<=0;
Writeln('Папуга  знущався над  дідусем  ',Day,'  днів. ' );
End;
Readkey;
End.
```

Задача 209

Умова: На скільки років необхідно покласти в банк суму X грошових одиниць, щоб одержати суму N грошових одиниць (N > X), якщо банк нараховує 200 % річних?

Розв'язання. Очевидно, що умовою виходу з цього циклу буде отримання заданої суми грошей. Якщо за умовою задачі *N > X, то* кожну перевірку ми будемо виконувати після того, як до вкладеної суми додамо щорічний банківський процент. Отже, програма має вигляд:

```
Program Example_209;
Uses crt;
Var X,N:real;  {X - початковий внесок,  N - бажана сума)
Rez:real;  {Rez  - результуюча сума на рахунку)
Years : longint;  {Years - термін перебування грошей в банку)
Begin
Clrscr;
Write(‘Введіть початкову суму внеску:  ') ;
Readln(X);
Write('Введіть бажану суму внеску:  ');
Readln(N);
If N<=X Then writeln(‘Ви вже маєте бажану суму!')
Else
Begin Rez:=X; Years:=0;
Repeat
Rez:=3*Rez; {200% річних збільшують  за рік внесок втричі}
```

```
 Years:=Years+I;
 Until  Rez>=N;
 Writeln ('Ви отримаєте бажану суму через ' , years,' років') ;
 End;
 Readkey;
 End.
```

Задача 231

Скласти програму, яка б допомогла працівникам ДАІ визначати кількість порушників перевищення швидкості на трасі, якщо відомо, що на даному проміжку траси встановлено обмеження на швидкість V_{max} а прилад фіксує швидкість автомобілів V_1 V_2 ..., V_n.

Розв'язання: В даній задачі ніяким чином не обумовлена умова виходу з циклу, тому є пропозиція: процес підрахування порушників необхідно закінчити тоді, коли чергове введене число буде недодатнім (дійсно, з від'ємною або нульовою швидкістю автомобіль рухатися не може). Для тимчасового зберігання значення швидкості чергового автомобіля ми будемо знову використовувати одну змінну. Програма, що виконує задані обчислення, має наступний вигляд:

```
 Program Example_231;
 Uses crt;
 Var V,Vmax:real;
 {V - швидкість автомобіля, Vmax - макс, дозволена швидкість}
  Count:longint;   {Count - кількість порушників)
 Begin
 Clrscr;
 Count:-0;   {На початку роботи порушники відсутні)
 Write ('Значення максимально дозволеної швидкості:   ') ;
 Readln(Vmax);
 Vmax:=abs (Vmax) ; {Знаходження модуля для виключення помилки введення від'ємної,
максимальної швидкості}
 Repeat
 Write('Значення швидкості чергового автомобіля: ') ;
 Readln(V);
 If V>Vmax then Count:=Count+I;
 Until V<=0;
 Writeln('Кількість порушників ',Count);
 Readkey;
 End.
```

Задача 251

Дано натуральне число n і дійсні числа a_1, a_2, ..., a_n. Відомо, що в заданій послідовності є хоча б одне нульове значення. Розглядаючи члени послідовності, що розташовані до першого нульового члена, визначити середнє арифметичне членів.

Розв'язання: Для розв'язання цієї задачі значення *n* є зайвим, якщо серед членів послідовності буде хоча б один нульовий елемент, тому ми враховувати цю змінну не будемо (хоча дітям можна пояснити, що у випадку відсутності нульового члена послідовності змінна *n* може використовуватись як додаткова для виходу з циклу, щоб виключити зациклення програми).

Отже, оскільки ми не знаємо, коли зустрінеться нульовий елемент, у програмі знаходиться сума всіх чисел послідовності (змінна *sum)* та кількість введених чисел (змінна *count),* а після виходу з циклу вже знаходиться безпосередньо середнє арифметичне членів послідовності як результат ділення суми на кількість чисел, зменшену на одиницю. Зменшення на одиницю відбувається тому, що фактично в тілі циклу буде підраховано один зайвий нульовий елемент (останній).

Програма має вигляд:

```
 Program Example_251_5;
 Uses crt;
 Var count:word;  {count - кількість членів послідовності до першого нульового елемента}
 a, Sum : real;   {a - черговий член послідовності,  Sum - сума членів послідовності до
першого «0»)
  SA:real;   {SA - середнє арифметичне}
 Begin   Clrscr;
 Sum:=0;  count:=0;   {Початкові значення дорівнюють «0»}
```

```
repeat
write('Введіть черговий член послідовності:  ') ;
readln(a);
Sum:=Sum+a; count:=count+1;
until a=0;
SA:=Sum/(count-1);
Writeln('Середнє арифметичне = ',SA:8:2);
Readkey;
End.
```

Запитання для перевірки засвоєння знань

1. *Що таке цикл з передумовою?*
2. *Що таке цикл з післяумовою?*
3. *Як в програмі уникнути «зациклення» ?*
4. *Які операції називаються бінарними?*
5. *З яких розділів складається програмний блок?*

Домашнє завдання:

1. Складіть програму обчислення добутку парних чисел, менших 15.
2. Складіть таблицю значень функції y = 5 x-2 на відрізку [1; 20] із кроком h = 2.3.
3. Складіть програму обчислення суми 80 перших членів арифметичної прогресії, якщо a_1 = 10; d = 3.

Урок 47. Практична робота 5 «Програми з повтореннями».

За наведеним сценарієм виконайте завдання по створенню та налагодженню алгоритмів та програм з повтореннями.

1) *Визначити кількість і типи вхідних даних відповідно до умови задачі.*
2) *Визначити кількість і типи вихідних даних відповідно до умови задачі.*
3) *Визначити тип алгоритму або його фрагментів, побудувати математичну модель.*
4) *Визначити необхідність введення проміжних даних, їх кількість та типи.*
5) *Розробити словесний опис алгоритму та побудувати його схему.*
6) *Визначити тип вказівок повторення, необхідних для створення алгоритму відповідно до умови задачі.*
7) *Визначити послідовність введення початкових та виведення результуючих даних з використанням відповідних коментарів і форматування.*
8) *Визначити коректність використання послідовних та вкладених вказівок повторення, наявність розгалужень.*
9) *Записати алгоритм мовою програмування.*
10) *Набрати текст програми, використовуючи середовище програмування та виконати налагодження програми, виправивши синтаксичні помилки.*
11) *Виконати програму, підготувавши систему тестів.*
12) *Реалізувати інші варіанти використання повторень для виконання сформульованої задачі та. проаналізувати їх щодо кількості виконуваних дій для різних початкових даних.*

Завдання:

1. Протабулюйте функцію y=cos2x на проміжку [0;pi] з кроком 0,25 і обчисліть добуток значень, що задовольняють умові *0,5<y<1.*
2. Протабулюйте свою функцію $y=x^2+\sin 3x$ на проміжку [1-;0] з кроком 0,2 і обчисліть кількість і суму значень більших, ніж -2 і менших, ніж 2.
3. Протабулюйте свою функцію $y=\sin 3x\cos 3x+2{,}4x^2-1/2x$ [0;4] з кроком 0,4 і визначіть кількість і добуток значень більших, ніж 3 і менших, ніж 6.
4. Протабулюйте функцію *z=cos(2x-3y),* змінюючи *x* на проміжку [0;2] з кроком 0,2, а *y* на проміжку [0;1] з кроком 0,1.
5. Протабулюйте функцію *z=sin(x-y),* змінюючи *x* на проміжку [0;1] з кроком 0,25, а *y* на проміжку [0;1] з кроком 0,2. Обчисліть середнє арифметичне додатних значень функції.
6. Складіть блок-схему алгоритму й програму обчислення добутку непарних чисел, менших 16.

7. Складіть таблицю значень функції y = 4 x-5 на відрізку [1; 30] із кроком h = 3.3.
8. Складіть програму обчислення суми 100 перших членів арифметичної прогресії, якщо a_1 = 5; d = 4.
9. Складіть блок-схему алгоритму й програму обчислення добутку цілих чисел із проміжку [-6; 5).
10. Складіть таблицю значень функції y = 5x² - 2x +1 на відрізку [-5; 5] із кроком h = 2.3.
11. Складіть програму обчислення суми 150 перших членів арифметичної прогресії, якщо a_1 = –200; d = 0,2.
12. Складіть блок-схему алгоритму й програму обчислення добутку цілих чисел із проміжку [-8; 4).
13. Складіть таблицю значень функції y = 4x² + 5x - 10 на відрізку [-9; 9] із кроком h = 3.3.
14. Складіть програму обчислення суми 180 перших членів арифметичної прогресії, якщо a_1 = –100; d = 0,4.

Дайте відповіді на запитання:

1. Що називається циклом?
2. Які типи циклів ви знаєте?
3. Які особливості запису параметру циклу?
4. Коли застосовують оператор **for?** Чому?
5. До якого типу даних належить змінна параметру циклу?
6. Яке призначення циклу **repeat?**
7. Який загальний вигляд має команда циклу **repeat?**
8. Яка різниця між командами **while** і **repeat?**

Урок 48. Обробка рекурентних послідовностей.

Мета: дати поняття про рекурентні формули, метод ітерацій, числа Фібоначчі, навчити розв'язувати задачі з обробки рекурентних послідовностей.

Тип уроку: вивчення нового навчального матеріалу.

Хід уроку.

I. Організаційний момент.

II. Викладання нового матеріалу.

1. Поняття про рекурентні формули. Прикладами рекурентних формул є формули для обчислення *наступного елемента* арифметичної чи геометричної прогресії, якщо відомо *попередній елемент* і різниця чи знаменник прогресії:

$a_{n+1}=a_n+d$, $n=$ 0,1,2,... $b_{n+1}=b_n*q$, n=0,1,2,...

У задачах про прогресії зазвичай відомо значення першого елемента, різниці чи знаменника, що достатньо для їхнього розв'язування.

2. Поняття про метод ітерацій. ***Ітераціями*** називаються послідовні уточнення значення деякої величини, отримані в результаті застосування рекурентних формул.

З а д а ч а 1. Обчислити із заданою точністю квадратний корінь від числа *n,* де *n* — це номер варіанта. Якщо n =1, 4, 9, 16, 25, 36, то число *n* збільшити на 50. Обчислення виконати з точністю 0,01. Визначити скільки було ітерацій. Виконати обчислення ще два рази, задавши точність **0,001** та **0,0001.**

Результати трьох експериментів записати у таблицю спостережень:

Число		Корінь	Точність	Кількість ітерацій
1)	*n* ?	**0,01**	?	
2)	*n* ?	**0,001**	?	
3)	*n*	?	**0,0001**	?

Теоретичні відомості. Квадратний корінь будь-якого додатного числа *n* можна визначити за такою рекурентною формулою:

$b_{i+1} := (b_i + n/b_i)/2,$ i=1,2,..., де $b_1 = n.$

Алгоритм:

1. Позначити через b_1 будь-яке початкове наближення до шуканої величини, ввести *n* і присвоїти *b1:= n;*
2. Позначити через *e* задану точність і присвоїти
e := **0,01,** i := 1,
3. Обчислити *b2 := (b1+ n/b1)/2.*
4. Доки |b*2 – b1*|> *e,* виконати обчислення
b1 := *b2,*
b2 := *(b1+ n/b1)/2,*
i := i + 1.

5. Після виходу з циклу «доки» останнє значення (b2) прийняти за відповідь.

Завдання. Складіть відповідну програму самостійно.

3. Числа Фібоначчі. Цю задачу сформулював і розв'язав італієць Фібоначчі у 1228 році.

Задача 2. Є пара кроликів. Вона дає приплід — нову пару кроликів — на третій місяць, а пізніше — щомісяця. Скільки пар кроликів буде наприкінці року?

Розв'яжемо задачу. Є послідовність чисел, де перші два числа — це 1 та 1, треба знайти інші. Третім числом буде 2. Розв'язок дають числа, які отримують за таким правилом: кожне наступне число (c) в послідовності є сумою двох попередніх (a та b). Ці числа називаються числами Фібоначчі. Визначимо десять чисел послідовності.

```
program Rabbits;
var a, b, c, i : integer;
begin
write('Числа Фобіначчі: 1  1');
a:= 1; b := 1; for i := 1 to 10 do
begin
c := a + b; a := b; b := c;
write(c:3)
end
end.
```

Отримаємо числа Фібоначчі:

1 1 2 3 5 8 13 21 34 55 89 144.

Підсумок уроку.

Домашнє завдання.

1. Перший член і різниця арифметичної прогресії дорівнюють відповідно a_0 і d (задайте їх довільними). Виведіть десять перших членів прогресії.

2. Перший член і знаменник спадної геометричної прогресії дорівнюють відповідно b_0 і q. Виведіть десять перших членів прогресії.

3. . Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2}=a_{n+1}-a_n$ $a_0=1$, $a_1=2$.

4. Задача Фібоначчі № 2. Є одна пара кролів. Пара починає щомісяця народжувати нову пару на третій місяць життя. Одна пара кролів живе 6 місяців. Скільки пар кролів буде через 12 місяців?

Урок 49. Самостійна робота №4.

Мета: перевірити вміння учнів розв'язувати задачі на використання різних типів циклів.

Тип уроку: контролю знань

Хід уроку.

I. Організаційний момент.

II. Самостійна робота.

Виконайте завдання

1. Нехай M = 128. Чому дорівнюватиме значення змінної M в результаті виконання послідовності операторів:

```
i: = -2;
while i <= 5 do begin
   M := M/2; i:=i + 2 end?
```

2. Якого значення набуде змінна x в результаті виконання послідовності операторів:

```
for i := 1 to 5 do begin read (x);
if x>=0 then x :=2*i
end;
```

де змінна x послідовно набуватиме значення:

1)1, 4, -2, 5, 0; 2)1, 3, 5, 2, 4?

3. Якого значення набуде змінна x в результаті виконання такої послідовності операторів:

```
x:=0.5;
```

for i := 1 to 5 do begin x:=x*2; x:=x+ 1 end?

4. Якого значення набуде змінна а в результаті виконання послідовності дій:

```
l)f:= false;                          2)f:=false;
  for i.:= 1 to 5 do                  for i := 1 to 5 do
    begin                               begin
      if a < 0 then f := true;if a < 0 then f := true;
      a :=a- 1                          a :=-a
    end;                                end;
```

для таких початкових значень змінної а:

а) 10; б)5; в) 1; г) - 10; д) 4.

5. Якого значення набуде змінна *t* в результаті виконання таких дій:

```
1) t := 10;          2)t:=1;          3) t := 1;
   while t > 1 do       while t > 1 do    repeat
     t:=t/2;              t:=t/2;           t:=t/2;
                                          until t<= 1.
```

6. Якого значення набуде змінна *s* після виконання такої послідовності дій:

```
l)s:=0;                          3)s:=0;
   i:=0;                            i:=0;
   while i < 5 do                   while i < 5 do
     i := i + 1;                      begin
     s:=s + 1/i;                        i:=i + 1;
  s:=s+ 1/i
end;
2)s:=0;                          4)s:=1;
   i:=1;                            n:=1;
   repeat                           for i := 2 to n do
     s:=s+1/i;                        s:=s + 1/i.
     i := i -1 until i <= 1;
```

7. Проаналізувати фрагменти алгоритмів і визначити, які значення будуть надруковані в результаті виконання таких дій:

```
1) s := 1;                       2)s:=1;
   For i := 1 to 10 do              for i := 1 to 10 do
     s := s + 1;                      begin
writeln (s);                          s :=s+ 1;
                                      writeln (s) end.
```

8. Скільки разів будуть надруковані числа 2 і 3 в результаті виконання таких дій:

```
1) for i := 1 to 3 do            2) for i := 1 to 3 do
     writeln (2);                     begin
   forj:=1to2do                       writeln (2);
     writeln (3);                     forj:=1to2do
                                        writeln (3); end.
```

9. Складіть програму, за допомогою якої визначить всі трьохзначні числа-паліндроми. (Число-паліндром однаково читається справа наліво і зліва направо: 88, 212, 767767 та ін.

Урок 50. Складання алгоритму з одним циклом.

Мета: формування навичок використання циклів для розв'язування типових задач, виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

І. Актуалізація опорних знань учнів.

Фронтальне опитування.

1. Чому не потрібно брати в операторні дужки групу операторів між словами repeat- until?
2. Якого типу може бути параметр циклу і?
3. Записати блок-схеми типів циклічних алгоритмів.
4. В якому випадку виникає ситуація зациклення програми?
5. Чи можна змінювати параметр цикле for в тілі циклу?

ІІ. Формування навичок.

1. Паскаль-Рулетка. У наступному прикладі число повторень циклу заздалегідь невідомо, тому замість циклу з лічильником краще використовувати одну з різновидів циклу з перевіркою умови. Пропонуємо пограти в просту, але азартну гру на вгадування цілого числа від 1 до 10. Нехай програма «загадає» таке число, а користувач уведе передбачуване значення. Якщо число вгадане, програма привітає переможця, а якщо ні — попросить його повторити спробу ще раз. Кожна безуспішна спроба знижує призові бали. На самому початку гравцеві призначається 10 призових балів. Опис алгоритму:

+ вибрати випадкове ціле число від 1 до 10;
+ вивести запрошення на уведення цілого значення;
+ якщо уведене число менше задуманого, сповістити про це гравцеві, інакше повідомити його про те, що уведене число більше задуманого;
+ повторювати уведення цілого значення доти, поки число не буде вгадано;
+ вивести привітання переможцеві й повідомити його про набране число балів;
+ завершити роботу.

Не виключена можливість того, що число буде вгадано відразу. У цьому випадку вже не треба виводити підказку гравцеві, тому варто використовувати цикл із передумовою while. . .do:

```
program roulette;
var number, guess, bonus : Byte;
begin
 bonus := 10;
Randomize;
number := Random(11);
WriteLn('Задумане ціле число від 0 до 10. Вгадайте!');
Wri teLn;
WriteLn('Уведіть ціле число від 0 до 10');
ReadLn(guess);
while guess <> number do
begin
Dec(bonus);
WriteLn( 'Ви не вгадали.');
Wri teLn;
if guess < number then
WriteLn('Ваше число менше задуманого')
else
WriteLn('Ваше число більше задуманого') ;
WriteLn('Спробуйте ще раз!' ) ;
R e a d L n ( g u e s s ) ;
e n d ;
WriteLn('Поздоровляю! Ви вгадали й набрали ', bonus, ' балів');
WriteLn('Натисніть <Enter>');
 ReadLn;
 end.
```

У цій програмі використовуються нові оператори. Це Randomize - початкова установка спеціальної процедури - «генератора» випадкових чисел Random(n), що видає випадкові цілі числа від 0 до n - 1, а також Dec (bonus) - виклик процедури, що зменшує на одиницю значення змінної bonus.

2. Спробуємо розбагатіти. Розглянемо приклад використання циклу з післяумовою. Нехай хтось, маючи певну грошову суму, відкрив рахунок у банку. Банк щорічно нараховує певний відсоток від внеску (це називається «дисконтною ставкою відсотка»), відповідно збільшується й сума внеску. Уважається, що цей відсоток не залежить від часу й від величини внеску. Така схема називається «правилом складних відсотків». Необхідно написати програму, що розраховує величину внеску й виводить цю величину для кожного року доти, поки величина внеску не подвоїться. Розглянемо алгоритм розв'язання даної задачі:

1. Увести первісну величину внеску, дисконтну ставку відсотка й рік, коли гроші поклали в банк.
2. Розрахувати нову величину внеску.
3. Вивести рік і величину внеску цього року.

4. Повторювати кроки 2 і 3 дoти, поки величин внеску не подвоїться.

Текст програми:

```
program rockafeller;
var balance, balance_initial, rate, interest : Real; year : Word;
begin
WriteLn('Уведіть рік внесення грошей у банк '); ReadLn(year) ;
WriteLn('Уведіть величину внеску');
ReadLn(balance);
WriteLn('Уведіть ставку відсотка (0.0-1.0)'); ReadLn(rate);
balance_initial := balance;
WriteLn('Рік Внесок');
WriteLn('========== );
repeat
interest := rate * balance;
balance := balance + interest;
Inc(year) ;
WriteLn(year :6, ' ', balance)
until balance > 2 * balance_initial;
WriteLn('Натисніть <Enter>');
ReadLn;
end.
```

В даному випадку тіло циклу виконується хоча б один раз, тому використовується цикл із післяумовою. Процедура Inc(year) збільшує на одиницю значення змінної year.

3. Гра Баше на 15 предметах. Гра Баше відома у Франції. Правила гри Баше такі. Є 15 однакових предметів (за звичай це дерев'яні палички). У грі беруть участь двоє. Суперники ходять по черзі, за кожний хід граючий може взяти 1, 2 або 3 предмети. Програє той, хто вимушений взяти останній предмет. Пропускати хід неможна. Припускаючи, що нашим суперником по грі Баше буде комп'ютер, напишемо для нього програму.

Насамперед придумаємо алгоритм виграшної стратегії, при якій перший суперник починає й виграє. Очевидно, що дані 15 предметів можна розбити на 5 груп, що містять не більше ніж по 4 предмети.

При такому розподілі починаючий гравець бере перші 2 предмети, і далі, скільки б не взяв другий гравець (1,2 або 3 предмети), перший буде добирати до 4 предметів так, щоб разом вони за два напівходи вибрали одну групу. Після чотирьох ходів перший гравець залишає суперникові 1 предмет і виграє.

```
program bashe;
var a, b, m : Integer;
begin m := 15; a := 2; m := m - a;
while m > 1 do
begin
Write(' я взяв ' , a);
if a = 1 then
Write(' предмет')
else
Write(' предмета');
WriteLn(' залишилося ', m) ;
WriteLn('Суперник, ваш хід:');
ReadLn(b);
 a := 4 - b;
m := m - (a + b) ;
end;
Wri tel_n( 'Залишився ' , m, ' предмет, Ви програли');
end.
```

Змініть цю програму так, щоб можна було й грати удвох з людиною.

4. У пошуках досконалості. Число називається *досконалим*, якщо воно дорівнює сумі всіх своїх дільників, включаючи 1, наприклад 6=1+2 + 3, 28 =1+2 + + 4 + 7 + 14. Поставимо задачу визначити, чи є задане число досконалим. Алгоритм розв'язання цієї задачі досить простий. Необхідно перебрати всі натуральні числа від 1 до половинки (або і її цілої частини) від числа, що перевіряється, і, якщо остача від ділення розглянутого числа на дане дорівнює нулю, додати чергове значення до суми. Значення, більші половинки числа, що перевіряється, не має змісту розглядати, тому що вони не будуть його дільниками.

```
program sover; var
a, i , s : Integer;
begin
Write('Уведіть ціле число a:');
ReadLn(a);
s := 0;
for i := 1 to a div 2 do {Підрахунок суми дільників}
if a mod i = 0 then
begin
s : = s + i ; Write(' + ', i)
end;
if s = a then
Writeln('Число ', a, ' досконале')
 else
Wri teln(' Число ', a, ' не досконале ') ;
end.
```

Підсумок уроку.

Домашнє завдання.

Урок 51. Розв'язування задач на використання вказівки циклу.

Мета: формування навичок використання циклів для розв'язування типових задач, виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

І. Актуалізація опорних знань учнів.

Фронтальне опитування.

1. Розкажіть про організацію циклів: з передумовою, з післяумовою, з параметром;
2. Що має містити алгоритм циклічної структури?
3. Порівняйте можливості трьох типів циклів.
4. Які правила слід виконувати при використанні циклу з параметром?

ІІ. Формування навичок.

Приклад 1. Складіть програму знаходження кількості всіх тризначних чисел, які діляться на кожну зі своїх цифр.

```
Program Variant_1a;
Var i,  j,  L:  byte; c,  k:  word;
Begin k:=0;
For i:  = 1  to  9 do
For j:  s 1 to 9 do
For L:  = 1  to  9 do
Begin
c:  =  100*i+10*j+L;
if (c mod i=0) and (c mod j=0) and (c mod L=0)
then begin
```

```
k:=k+1;
write ('c, ' ') ;
end
end
writeln;
wrieln ('к =', к) ;
 end.
```

У другому варіанті розв'язання ми замість генерації числа будемо перебирати всі трицифрові числа від 111 (всі попередні обов'язково мали хоча б один 0 у своєму складі) до 999.

```
Program Variant_lb;
Var k, i: word; c, d, c: byte;
Begin
k:=0; i: =111;
while 1 < 999 do
Begin
c: = i mod 10; {цифра одиниць}
d: =i div 10 mod 10; {цифра десятків}
e: = i div 10 mod 10; {цифра сотень}
if (i<>0) and (d<>0) and (e<>0)
then if (i mod c=0) and (i mod d=0) and (i mod e=0) then begin
k:==k+1;
write   ('i,    ' ');
end
i: =i+1 ;
end;
writeln;
wrieln   ('k =',   k) ;
End.
```

Наведемо ще один варіант розв'язання з використанням циклу повторювати.

```
Program Variant_lv;
Var
k,  i:  word;  c,  d,  c:  byte;
Begin
k:=0;
i: = 111;
repeat
c:=i div 100; {цифра сотень}
d:=i div 10 - c *10; {цифра десятків}
e:=i - c*100 - d*10; {цифра одиниць }
if (c<>0) and (d<>0) and (e<>0)
and (i mod c=0) and (i mod d=0) and (i mod e=0)
then begin
k:=k+1;
write ('i, ' ');
end
i:=i+1; until i > 999;  writeln;
wrieln ('k=',  k) ;
End.
```

При визначенні подільності чисел націло ми користувалися функціями int (x) , div (a, b) *і операцією залишок від ділення* **mod.** *Значенням функції* **int** (x), *є найближче ціле число, що не перевищує задане. Функція* **div** (a, b) *видає цілу частину від ділення двох цілих чисел.* Функцію **div** (a, b) *надалі запишемо як операцію за аналогією з* ***функцією mod.***

Підсумок уроку.

Домашнє завдання.

1. Складіть алгоритм знаходження суми цифр усіх тризначних чисел, використовуючи розглянуті команди організації циклу.
2. Складіть алгоритм, що визначає, у скільки разів сума тризначних чисел менша від суми двозначних чисел.
3. Складіть алгоритм, що визначає суму парних цифр усіх двозначних чисел.
4. Складіть алгоритм, що визначає кількість «щасливих» квитків серед шестизначних чисел. (Число називають «щасливим», якщо сума перших трьох цифр дорівнює сумі наступних трьох цифр.)

Урок 52. Вкладені цикли.

Мета: формування навичок використання циклів для розв'язування типових задач, виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

I. Актуалізація опорних знань учнів.

II. Формування навичок.

У деяких випадках важливо повторити підзадачу кілька разів усередині більш загальної задачі. Один зі способів написання такої програми - включити цикл у набір інструкцій, що повторюються всередині іншого циклу. Така структура, що складається з циклу в циклі, називається вкладеними циклами.

Якщо в програмному коді використовуються вкладені цикли, то для підвищення наочності коду прийнято кожний наступний рівень вкладення зміщувати відносно попереднього.

Правило вкладення циклів: внутрішній цикл цілком укладається в тіло зовнішнього циклу

Приклад 5.19. Обчислення значення змінної *Y = 2*K+N* при всіх значеннях змінних *N*=1,2,3 і *K= 2,*4, 6, 8.

Якщо перебирати всі значення *N* і *K,* ми повинні отримати 12 значень змінної *Y.*

Скласти програму можна в такий спосіб: для кожного значення *N* перебрати всі значення *K* від 2 до 8, тобто *N* використати як параметр зовнішнього циклу, *K* - як параметр внутрішнього циклу. Фрагмент обчислення У має вигляд:

```
Program Prim;
 var N, к, Y : integer;
 Begin
        For N:=1 то 3 Do  Begin
                K:=2;
                While K<=8 Do Begin
                Y:=2*K+N;
                WriteLn (N:3, K:3, Y:3);
                K:=K+2
        End;
    End;
 End.
```

Параметр *N* змінюється з кроком 1, тому зовнішній цикл організований з використанням оператора For; параметр *K* змінюється з кроком 2, тому внутрішній цикл є циклом While.

Створення та реалізація програми з вкладеним циклом

(Самостійна робота)

Задача. Вивести на екран всі прості числа на інтервалі *[A; B]*.

Простими є числа (крім 1), які діляться без остачі тільки на 1 і на самого себе. Розглянемо алгоритм перевірки, чи є число С простим:

- Вводиться змінна Prap, яка відіграє роль прапорця (за її значенням можна визначити, чи відбулась деяка подія). Змінній Prap присвоюється початкове значення Fal se.
- Перебираються всі числа від 2 до (С div 2). Якщо число і є дільником числа С (тобто

С ділиться на і без остачі), то прапорцевій змінній Prap присвоюється значення True.

- Після завершення перебору можливих дільників перевіряється значення змінної Prap. Якщо значення Prap в процесі повторень циклу змінилось на True, то це означає, що С має хоча б один дільник і не є простим.

2. Запишемо програму для визначення, чи є число С простим. Зверніть увагу: задання значення змінної С навмисно пропущене.

```
var A, B, C, I,: integer;   Prap : Boolean;
Begin { задання значення змінної С }
 Prap := False; For I := 2 To C div 2 Do
if C Mod I = 0 Then Prap := True;
 If Not Prap    Then WriteLn (c,  ' - просте число')
Else WriteLn (C, ' - складене число');
End.
```

3. За умовою задачі потрібно перебрати всі числа в інтервалі *[A; B]*, тобто С послідовно набуває значень з діапазону *A..B*. Для кожного *С* від *А* до *В* потрібно виконати алгоритм перевірки, чи є число С простим.

4. Внесемо зміни до програми. Значення *А* і *В* вводяться з клавіатури:

```
var A, в, с, і : Integer; Prap : Boolean;
Begin
 write ('A, B =>'); ReadLn (A, B);
```

5. Додаємо керуючий рядок циклу *For*, тілом циклу якого є програмний код перевірки, чи є число С простим:

```
For C := A to B Do Begin
Prap := False;
For I := 2 To C div 2 Do
If C Mod I = 0 Then Prap := True;
if Not Prap
Then WriteLn (C, ' - просте число');
End;
```

6. Збережіть файл. Виконайте програму для різних значень *А* і *В*.

Підсумок уроку.

Домашнє завдання.

Дайте відповіді на запитання:

1. *Чи може оператор, повторюваний у циклі, сам бути циклом?*
2. *У чому полягає правило вкладення циклів?*
3. *Наведіть приклади задач, при розв'язуванні яких виникає необхідність використання вкладених циклів.*

Задача. *Скласти програму, яка друкує всі 3-цифрові натуральні числа, сума цифр яких дорівнює їхньому добутку і визначає кількість таких чисел. Для розв'язання поставленої задачі зовсім не обов'язково перебирати всі натуральні числа від 100 до 999, тому що нас цікавить не саме число, а цифри в його десятковому записі. Тому можна перебирати всі можливі сполучення цифр, з яких складається десятковий запис трицифрового числа і перевіряти для кожного сполучення умову задачі. Створіть новий файл. Запишіть програму і виконайте її. Проаналізуйте результат роботи програми.*

```
var A, B, C, K : integer;
Begin
K := 0;
For A := 1 To 9 Do
For B := 0 то 9 Do
For C := 0 To 9 Do
Lf A + B + C = A* B * C  Then
  Begin
WriteLn (100*A+10*B+ C);
K := к + 1;
 End;
WriteLn ('K=', K);
End.
```

3) Внесіть зміни у програму з тим, щоб знайти кількість усіх трицифрових натуральних чисел, сума цифр яких дорівнює даному цілому числу. Виконайте програму, проаналізуйте результат.

4) Сума цифр двоцифрового числа дорівнює 11. Якщо до цього числа додати 27, то вийде число, записане тими ж цифрами, але в зворотному порядку. Знайдіть це число.

Ідея розв'язування та ж сама, що й у п.1: перебираємо всі можливі сполучення цифр A і B в десятковому записі двоцифрового числа AB, і для кожного сполучення перевіряємо умову задачі. Умову задачі можна записати у вигляді логічного виразу:

(10*A+B)+27 = 10*B+A.

Після внесення змін програма має такий вигляд:

```
Var A, B : Integer;
 Begin
For A := 1 To 9 Do
For B := 0 To 9 Do
If (10*A+B)+27 = 10*B+A Then WriteLn (10*A+B);
End.
```

5) Сума квадратів цифр двоцифрового числа на 1 більша від потроєного добутку цих цифр. Після ділення цього двоцифрового числа на суму його цифр у частці виходить 7 і в остачі 6. Знайдіть це число.

Умову задачі можна записати у вигляді логічного виразу: (A*A+B*B=3*A*B+1) And ((10*A+B) div (A+B)=7) And ((10*A+B) mod (A+B)=6) *або рівносильного виразу:*

(A*A+B*B=3*A*B+1) And (10*A+B = (A+B)*7+6).

6) Внесіть зміни у програму, виконайте її, перевірте, чи отримали ви правильний результат.

Урок 53. Вкладені цикли.

Мета: формування навичок використання вкладених циклів для розв'язування типових задач, виховання інформаційної культури.

Тип уроку: формування навичок.

Хід уроку.

I. Актуалізація опорних знань учнів.

Дайте характеристику вказівкам циклу трьох типів.

II. Формування навичок.

Розглянемо приклад, де доводиться використовувати два цикли, один усередині іншого.

***Приклад** 1.* Скласти програму виведення всіх натуральних чисел, менших за *n,* квадрат суми цифр яких дорівнює заданому числу m.

Ідея розв'язання. Вводиться натуральне число, до якого треба виводити всі натуральні числа, що задовольняють задану умову. Нехай, наприклад, користувач уведе число — 21.

Друге число, яке треба ввести користувачу, — це число, якому дорівнює квадрат суми цифр натуральних чисел.

Зрозуміло, що це число має бути точним квадратом, воно може дорівнювати: 4, 9, 16, 25, 36 і т.д.

Припустимо, що користувач увів число 4.

Треба знайти всі натуральні числа від 1 до 21, квадрат суми цифр яких дорівнює 4. Починаємо з чисел: 1, 2, 3, 4, 5,..., 21, вибирати ті, які задовольняють задану умову.

Перше з них 2, тому що $2^2 = 4$, друге 11, тому що $(1 + 1)^2 = 2^2 = 4$, третє 20, тому що $(2 + 0)^2 = 2^2 = 4$.

Інших натуральних чисел до 21, що задовольняють задану умову, *немає.*

Усі відібрані числа треба вивести на екран: 2, 11 і 20.

Алгоритм 1. Розділ описів

Змінні: *n, m, k, a, p, s.* Тип цілий.

n — для границі значень натуральних чисел, *m* — для числа, з яким порівнюється квадрат суми цифр (точний квадрат), *k* — для організації перебору натуральних чисел від 1 до *n,* ***a*** — для тимчасового зберігання натурального числа, перед тим, як буде визначатися сума його цифр, *p* — для зберігання виділеної цифри числа, s — для зберігання суми цифр числа.

2. **Розділ операторів**

Уведення значень *n і m.* Установити початкове значення для *k* (ця змінна *«перебирає»* усі натуральні числа від 1 до *n, k* = 1).

Цикл, *поки* k <= n.

У циклі: встановити початкові значення для суми s (s:=0); запам'ятати число в змінну *а* (а:= к), оскільки після знаходження суми його цифр саме число буде зіпсовано.

Цикл для підрахунку суми цифр, *поки* k <> 0.

У циклі: виділяти по одній цифрі числа відомим способом; накопичувати суму цифр; зменшувати число на останню цифру (відкидати її цілочисельним діленням на 10).

Закінчити цикл для підрахунку суми цифр.

Перевірка виконання умови, тобто: ***якщо*** квадрат суми цифр дорівнює заданому числу, ***тоді*** вивести це натуральне число на екран.

Перейти до перевірки наступного числа.

Закінчити основний цикл перевірки чисел.

4. Закінчити програму.

Отже, маємо програму:

```
Program Problem; uses WinCrt; var
n,m,k,a,p,s: integer;
Flag: boolean;
 begin
write ('Уведіть натуральне число, до якого ');
write('виводити шукані числа ');
readln(n);
writeln('Уведіть число, з яким порівнюєте квадрат');
write('його суми цифр. Воно має бути точн. квадрат. ');
readln(m);
write('Шукані числа: ') ;
k:= 1;  Flag:=False;
while  к  <= n  do
begin
s: = 0;  a: = к;
while к <> 0 do
begin
p:= к mod 10;  s: = s  + p; k:= к div 10 end;
if  sqr(s)  = m then begin
write(a,   '  ') ;
Flag:=True;    end;
к  := a +1
end
if NOT Flag
then writeln('Таких чисел для заданих вхідних даних не  існує')
end.
```

У програмі два цикли. Один — зовнішній, для натуральних чисел, другий — внутрішній, для підрахунку суми цифр числа.

Зверніть увагу, що у випадку некоректних вхідних даних *(**n*** — дуже мале, або ***m*** — не є точним квадратом числа) програма не може видати результат. Щоб запобігти цьому, пропоную використання змінної логічного типу **Flag**. Для цієї задачі міркування можуть бути такими: на початку програми змінній **Flag** надається значення **Fals e** (числа ще не знайдені). Всередині циклу якщо знайдеться хоч одне число, що задовольняє умову, разом з виведенням його на екран змінній **Flag** присвоюється значення

True. Таким чином, якщо змінна **Flag** після виконання програми не зміниться, це свідчить про те, що необхідні числа не знайдені. В цьому випадку слід видати про це повідомлення.

5. Практичне завдання

Скласти програму пошуку всіх натуральних чисел *n* ≤ 100 000, сума цифр яких дорівнює заданому натуральному числу ***a***.

```
Program Task;
uses WinCrt;
var a,p,s : integer; n,b: longint;
begin
write('Уведіть натуральне число ') ;
readln(a);
if a>45 then writeln ('Таких чисел не існує')
else begin
b:= 1;
writeln('Натуральні числа, сума цифр ');
write('яких дорівнює числу ', a, ' наступні: ');
while b < 100000 do
begin
s := 0; n := b;
while n <> 0 do
begin
p:= n mod 10; s:= s +p; n:= n div 10 end;
if s = a then write(b, ', ') ;
b := b +1
end;
end;
end.
```

Налагодити програму і написати алгоритм етапів розв'язання.

Вправи для роботи на закріплення

1. Знайти всі тризначні числа, при цілочисельному діленні кожного з яких на 11 отримуємо частку, що дорівнює сумі квадратів значень окремих цифр числа.
2. Тризначне десяткове число закінчується цифрою 3. Якщо цю цифру перемістити через два знаки ліворуч, тобто з цієї цифри буде починатися запис нового числа, то це нове число буде на одиницю більше від потроєного вихідного числа. Знайдіть це число.
3. Знайдіть усі тризначні числа, що дорівнюють сумі кубів своїх цифр.
4. Шестизначне десяткове число починається ліворуч цифрою 1. Якщо цю цифру перенести зі свого місця на останнє місце праворуч, то значення утвореного числа буде втроє більшим від вихідного. Знайдіть вихідне число.
5. Дано ціле число *n* > 10 Написати програму одержання ***m*** останніх цифр десяткового запису числа *n.*
6. Знайти чотиризначне число, що дорівнює квадрату числа, вираженого двома останніми цифрами цього чотиризначного числа.

Підсумок уроку.

Домашнє завдання.

1. Знайти чотиризначні числа, кожне з яких ділиться націло на 11 і сума цифр яких дорівнює 11.
2. Знайти чотиризначні числа, які, якщо приписати праворуч до числа 400, дають повний квадрат.

Урок 54. Випадки застосування циклів з післяумовою.

Мета: визначити відмінності циклів з післяумовою та передумовою в мові Pascal. Визначити застосування цієї конструкції. Навчитися застосовувати правильний різновид циклу.

Тип уроку: формування навичок

Хід уроку.

І. Актуалізація опорних знань учнів.

Визначіть основні відмінності циклу з післяумовою від циклу **while.** (очікувана відповідь)

1. Коли умова стає ІСТИННОЮ, виконання циклу **repeat until** припиняється, а виконання циклу **while** триватиме.

2. Для циклу **while** на початку перевіряється умова, і тільки потім виконується дія. Для циклу **repeat until** виконується дія, і тільки потім перевіряється умова. Тому тіло циклу **repeat until** хоча б один раз виконається до перевірки умови.

3. Для циклу **repeat until** не потрібно операторних дужок.

Дійсно, ***відмінності між циклом while і циклом repeat:***

1. Оператори, що знаходяться в циклі **while,** повторюються доти, доки умова істинна. Послідовність операторів, що знаходяться в циклі **repeat,** повторюється доти, доки умова хибна.

Отже, у циклі **while** використовується умова продовження циклу, а в циклі **repeat** — умова закінчення циклу.

2. У циклі **while** повторюється один оператор (кілька операторів треба поєднувати в складений оператор за допомогою операторних дужок **begin... end),** а в циклі **repeat** можна використовувати кілька операторів без операторних дужок.

3. У циклі **while** спочатку перевіряється умова, а після цього залежно від значення умови (якщо істинна) оператор чи група операторів після слова do.

4. У циклі **repeat** послідовність операторів виконується один раз, а після цього перевіряється умова, тобто ця послідовність *завжди* виконується *хоча б один* раз, а в циклі **while** оператори, що складають тіло циклу, можуть узагалі не виконуватися жодного разу (у випадку, якщо умова відразу хибна).

II. Формування навичок.

Розглянемо приклади.

Приклад 1. Знайти найменше натуральне число, що дає при діленні націло на 2, 3, 4, 5,6 відповідно остачу 1, 2, 3, 4, 5.

Розв'язання. Береться найменше натуральне число — одиниця — і знаходяться остачі від ділення його на 2, 3, 4, 5 і 6; якщо залишки дорівнюватимуть 1, 2, 3, 4 і 5, тоді це число є шуканим, його треба видати на екран і закінчити програму, у протилежному випадку треба брати наступне натуральне число 2 і перевіряти його, і так далі.

Отже, маємо програму:

```
Program Problem1;
uses Crt;
var n : integer;
begin
n := 0;
repeat
n : -  n + 1 ;
until (n mod 2=1) and (n mod 3=2) and (n mod 4 = 3) and (n mod 5=4) and (n mod 6=5);
writeln('Шукане ціле число ', n)
end.
```

Ще один приклад, що демонструє роботу циклу з післяумовою.

I

Приклад 2. Числа, що однаково читаються зліва направо, і справа наліво, називаються паліндромами. Наприклад, числа 42324 чи 1331 — паліндроми. Складіть програму, що буде знаходити числа-паліндроми із заданого проміжку.

Розв'язання. Перевернути число і порівняти отримане число із заданим.

Нехай дано число *a,* уведемо ще одну змінну *b,* якій буде присвоєно значення змінної *a* (для чого це робиться, ви довідаєтеся пізніше): b:=a;

Заведемо ще одну змінну a_1 для нового числа, у якому цифри вже будуть переставлені.

Початкове значення цієї змінної — нуль: a1:=0;

Чому значення цієї змінної дорівнює нулю, стане ясно з програми.

Далі організуємо цикл repeat, у якому відбуватиметься перестановка цифр числа *b:* repeat

a1:= a1*10 +b mod 10;

b:= b div 10 until b = 0;

Отже, у циклі, також як і в циклі while... do..., виділяється остання цифра:

b mod 10; (наприклад, 343 mod 10 = 3); змінній a_1 надається значення: a1:= a1*10 +b mod 10; 0 * 10 +3 =3;

«відкидається» остання цифра заданого числа за допомогою операції цілочисельного ділення: b:= b div 10; 343 div 10 = 34;

перевіряється умова: *b* = 0, 34 = 0, умова хибна, значить цикл продовжується.

Виділяється остання цифра вже нового числа:

b mod 10 = 34; 34 mod 10 = 4;

число a_1 уже рівне 3, збільшується у 10 разів і до результату додається наступна цифра 4:

a1:= a1*10 +b mod 10;
3 * 10 +4 = 34;
«відкидається» остання цифра числа b:
b:= b div 10;
34 div 10 = 3;
перевіряється умова: b = 0,3 = 0; умова хибна, значить цикл продовжується.
Виділяється остання цифра числа: b mod 10; 3 mod 10 = 3; формується нове число: a1:= a1*10+b mod 10; 34*10+3=343;
«відкидається» остання цифра числа і виходить нове число: b:= b div 10 ; 3 div 10 = 0;
перевіряється умова: b = 0, 0 = 0; умова істинна, значить цикл закінчує свою роботу.

Тепер стає ясно, чому введена інша змінна b для заданого числа: її значення в циклі змінюється від початкового до нуля і, щоб зберегти початкове число в змінній a, і вводиться так звана, «робоча» змінна b.

Після закінчення циклу перевертання числа порівнюється початкове число, що «зберіглося у змінній a, і число, що вийшло після перестановки цифр і «накопичилося» у змінній a_1 .

Якщо a = a_1 , тоді значення a видається на екран, оскільки це число є *паліндромом.*

Далі, значення a збільшується на 1, тобто береться для розгляду наступне натуральне число і знову продовжується зовнішній цикл. Число перевертається, отримане нове число — a_1 порівнюється з початковим a і так далі.

Зовнішній цикл закінчується, коли значення o стає рівним правій границі інтервалу — *n.*

Отже, маємо програму:

```
Program Problem2;
uses Crt;
var m, n, a, b, al : longint;
begin
write('Уведіть ліву границю проміжку ');
readln(m) ;
write('Уведіть праву границю проміжку '); readln(n);
 a := m;
writeln('Числа паліндроми з [', m, ';', n, ']');
repeat
b := a; a1 := 0;
repeat a1 := a1*10 +b mod 10; b := b div 10
until b=0;
if al = a then write(a, ' ') ;
a := a + 1
until a > n;
end.
```

Програми, складені з циклом з передумовою (while... do...), легко можна переробити на програми з циклом з післяумовою (repeat... until...):

Отже, маємо програми

1. Для підрахунку суми цифр числа:

```
Program Sum; { Сума цифр числа }
uses Crt;
var n, s, a : longint;
begin
write('Уведіть ціле число ') ;
readln(n);
a := n; s := 0;
repeat
s := s +n mod 10; n := n div 10
until n = 0;
writeln('Сума цифр числа ', a, ' дорівнює ', s)
end.
```

2. Для перестановки першої й останньої цифр у числі:

```
Program Transpose;
uses Crt;
var n, n1, p, a, i: longint;
begin
write('Уведіть натуральне число'); readln(n);
a:= n; i:= 1; p:= n mod 10;
```

```
{остання цифра введеного числа}
repeat
i:= i*10;  n:= n div 10
until n<10;
n1:= a  -  n*i  - p +n  +p*i;
writeln('Число після перестановки цифр   ',   n1)
end.
```

Підсумок уроку.

Домашнє завдання

1 *Поясніть роботу вкладених циклів.*
2 *За яким принципом організовуються вкладені цикли ?*
3 *Чи можна два вкладених цикли замінити двома послідовними циклами? Що при цьому зміниться?*
4 *Чи може бути, на ваш погляд, більше, ніж два вкладених цикли?*

Знайти найменше натуральне число, кратне 131, з парною кількістю цифр. Скласти програму.

Очікувана відповідь:

```
 Program Task;
uses Crt;
var n,   a,   k:   longint;
begin
n:= 131;
repeat
n: = n + 131 ;
a:= n;   k:= 0;
repeat
k:= k +1;
a:= a div 10
until a = 0;
until k mod 2=0;
writeln('Найменше натуральне число,   кратне 131');
writeln('з парною кількістю цифр дорівнює   ',   n)
end.
```

Урок 55. ***Практична робота №6 «Програми з повтореннями»***

За наведеним сценарієм виконайте завдання по створенню та налагодженню алгоритмів та програм з повтореннями.

1) *Визначити кількість і типи вхідних даних відповідно до умови задачі.*
2) *Визначити кількість і типи вихідних даних відповідно до умови задачі.*
3) *Визначити тип алгоритму або його фрагментів, побудувати математичну модель.*
4) *Визначити необхідність введення проміжних даних, їх кількість та типи.*
5) *Розробити словесний опис алгоритму та побудувати його схему.*
6) *Визначити тип вказівок повторення, необхідних для створення алгоритму відповідно до умови задачі.*
7) *Визначити послідовність введення початкових та виведення результуючих даних з використанням відповідних коментарів і форматування.*
8) *Визначити коректність використання послідовних та вкладених вказівок повторення, наявність розгалужень.*
9) *Записати алгоритм мовою програмування.*
10) *Набрати текст програми, використовуючи середовище програмування та виконати налагодження програми, виправивши синтаксичні помилки.*
11) *Виконати програму, підготувавши систему тестів.*

Завдання для практичної роботи:

1. Складіть програму для знаходження суми цілих додатних парних чисел, менших 100.

2. Дана послідовність N чисел. Визначити середнє арифметичне непарних від'ємних чисел, добуток чисел, кратних 5 з проміжку [-20,20], кількість нульових чисел і замінити числа, які менші 3, на 100.

```
Program prim2;
Const N=20; var
Sr :real; S, kol, kolo, i, N, p, x: integer;
Begin
S:=0; p:=1; kol:=0; kolo:=0;
For i:=1 to n do
Begin
Writeln('введіть число послідовності'); Readln(x);
 If (x mod 2 <> 0) and (x<0) then Begin S:=s+x;   Kolo:=kolo+1, End;
If (x mod 5 =0) and (x >=-20) and (x <20) then p:=p*x;
If x=0 then kol:=kol+1;
If x<3 then Begin x:=100; Writeln('число змінилося' ,x); End;
End;
Sr:=s/kolo; Writeln('середнє=', sr:8:4,  'добуток =  ', p,  'кількість нульових чисел=', kol);
End.
```

3. В послідовності натуральних чисел визначити найменший і найбільший елементи та їх номера.

```
Program prim 8;
Var x,I,max,min,nmax,nmin,n:integer;
Begin
Writeln('введіть кількість членів послідовності'); Readln(n);
Writeln('введіть перше число послідовності'); Readln(x);
Max:=x; min:=x; nmax:=1; nmin:=1;
For i:=2 to n do Begin
Readln(x);   If x<min then Begin     Min:=x; nmin:=I;     End;
If x>max then    Begin      Max:=x; nmax:=I;     End;   End;
Writeln('max=',max,'nmax=',nmax, 'min=',min,'nmin=',nmin );
End.
```

4. Дано дійсне число A и натуральне число N. Обчислити:
A(A+1)(A+2)...(A+N-1)

```
Program prim4;
var A, P: real; N, i: integer;
begin
write (' ввести числа a i n'); readln(A,N);
P: = A; i:=1;
while i<N do       begin       P:-P+(A+I);  I:=I+1      end;
writeln (' результат =', P:6:2);
end.
```

Урок 56. Самостійна робота

Мета: перевірити знання учнів по темі «Організація циклів»
Тип уроку: контролю знань
Хід уроку.
I. Організаційний момент.
II. Виконання самостійної роботи.
1. Чому буде дорівнювати x після виконання інструкцій?
x:=0; for i:= -5 to 0 do n:=i*i;   x:=x+n;
2. Що буде виведено на екран у результаті виконання інструкцій?
n:=5; x:=0; for i:=1 to n do x:=i; writeln (x);
3. Що буде виведено на екран?
for i:=1 to 5 do for j:=1 to 5 do   write ('*');
4. Що буде на екрані в результаті виконання інструкцій?

n:=1; while n<=10 do begin x:=n*n; writeln (n:6,", x:6); end.

5. Обчисліть значення многочлена $x^5-9x^4+1,7x^2-9,6$ для x= 0,1, …,5
6. Обчислити значення функції $y=4x^3-2x^2+5$ для $x\in[-3;1]$ з кроком 0,1.
7. Ввести 12 випадкових чисел. Підрахувати суму додатних і добуток від'ємних чисел. Якщо серед введених чисел були 0, то надрукувати повідомлення про це.

Урок 57-58. Тематична атестація «Організація циклів»

Варіант 1

1. Обчисліть a, якщо a:=6; **for** i:=1 to 2 do a:=a*i-2; a:=a+1;
2. Обчисліть p, якщо p:=2; **while** p<=10 **do** begin p:=2*p+1; p:=p+1 **end;**
3. 1дюйм=2,54 см =1/12 фута. Складіть програму для виведення таблиці цих мір для перших дванадцяти дюймів. Нарисуйте графічну схему алгоритму.
4. Нарисуйте графічну схему і складіть програму для табулювання функції $y= x^2\sin 2x$ на проміжку [-1;1] з кроком 0,2 і обчислення суми та кількості додатних значень. Додаткове завдання: обчисліть добуток першого і максимального значень.

Варіант 2

1. Обчисліть a, якщо a:=2; **for** i:=1 to 3 do **begin** a:=a+i; a:=a-1 **end;**
2. Обчисліть p, якщо p:=1; **while** p<10 **do** p:=2*p+1; p:=p+1 ;
3. Нарисуйте графічну схему і напишіть програму для виведення на екран номерів і значень лише додатних елементів послідовності isini, i=1,2,...,20.
4. Нарисуйте графічну схему і складіть програму для табулювання функції $y=x^3\sin 3x$: на проміжку [-2; 2] з кроком 0,4 і обчислення добутку та кількості значень, що задовольняють умову -1<y<2. Додаткове завдання: обчисліть суму мінімального й останнього значень функції.

Урок 59-66. Розв'язування задач.

Мета: повторити базові структури алгоритмів, формувати навички використання набутих раніше знань для розв'язання різноманітних задач.

Тип уроку: формування навичок.

Хід уроку.

І. Актуалізація опорних знань учнів.

Повторення теоретичного матеріалу курсу.

Підсумки курсу

1. Розглянуто поняття алгоритму.
2. Визначено форми подання алгоритмів:
- *словесні;*
- *словесно-формульні;*
- *формульні;*
- *графічні (у вигляді блок-схем).*

3. Визначено типи алгоритмів:
- *лінійні;*
- *розгалужені;*
- *циклічні;*
- *змішані.*

4. Визначено властивості алгоритму:
- *дискретність;*
- *скінченність;*
- *зрозумілість;*
- *визначеність;*
- *масовість.*

5. З'ясовано поняття формального виконання алгоритму.
6. Дано визначення алгоритмічної мови:
7. Визначено поняття:
- *алфавіт мови;*
- *синтаксис мови;*
- *елементи мови;*
- *символи;*
- *службові слова;*
- *команди;*
- *об'єкти мови:*

- *константи, змінні (числові, символьні, масиви);*
- *допоміжні алгоритми (підпрограми - функції або процедури);*
- *вирази.*

8. Введено типізацію величин і пояснено її необхідність.
9. Введення поняття « відношення між величинами ».
10. Визначено форму запису словесного алгоритму й алгоритму з використанням величин.
11. Розглянуто основні конструкції алгоритмічної мови:

- *присвоювання;*
- *розгалуження;*
- *повторення.*

II. Формування навичок.

Розв'язування задач.

Скласти і виконати програму, задавши вхідні дані самостійно.

1. Квіткова клумба має форму круга. Обчислити її периметр і площу за заданим радіусом.
2. Обчислити периметр і площу прямокутного трикутника за заданим катетом та гострим кутом.
3. Обчислити довжину кола і площу круга за заданим діаметром.
4. Ділянка лісу має форму рівнобічної трапеції. Обчислити її периметр і площу за заданими сторонами.
5. Ресторан закуповує щодня масло m_1 кг по 8.50 грн. за кілограм, сметану t_2 кг по 2.40 грн., вершки t_3 кг по 4.10 грн. Визначити суми, потрібні для купівлі окремих продуктів, і загальну суму.
6. Скільки секунд мають доба, тиждень, рік?
7. Обчислити кінетичну $E=mv^2/2$ та потенціальну $P=mgh$ енергії тіла заданої маси m, яке рухається на висоті h зі швидкістю v.
8. Ціни на два види товарів зросли на p відсотків. Вивести старі та нові ціни.
9. Обчислити площу поверхні $S=4\pi r^2$ та об'єм $V=4\pi r^3/3$ сфери за заданим радіусом r.
10. Швидкість світла 299792 км/с. Яку відстань долає світло за годину, добу?
11. Увести врожайність трьох сортів пшениці (36, 40, 44 т/га) і розміри трьох відповідних полів (у га). Скільки зібрали пшениці з кожного поля і з трьох полів разом?
12. Радіус Місяця 1740 км. Обчислити площу поверхні $S=4\pi r^2$ та об'єм планети $V=(4/3)\pi r^3$.
13. Обчислити довжину гіпотенузи та площу прямокутного трикутника за заданими двома катетами.
14. Обчислити об'єм та площу бічної поверхні куба, якщо відоме ребро.
15. Увести продуктивності роботи трьох труб, які наповнюють басейн, і час їхньої роботи. Скільки води набрано в басейні?
16. Яку площу і периметр матиме квадрат, описаний навколо круга заданої площі S.
17. Тіло падає з прискоренням g. Визначити пройдений тілом шлях $h=gt^2/2$ після першої та другої секунд падіння.
18. Обчислити периметр і площу прямокутного трикутника за заданими катетами.
19. Телефонні розмови з трьома населеними пунктами коштують c_1, c_2, c_3 коп/хв. Розмови тривали t_1, t_2, t_3 хв відповідно. Яку суму нарахує комп'ютер до оплати за кожну і всі розмови?
20. Обчислити площу бічної поверхні $S=2\pi rh$ та об'єм $V=\pi r^2h$ діжки за заданою висотою h та радіусом основи r.
21. Квіткова клумба має форму квадрата. Обчислити її периметр і площу за заданою стороною.
22. Обчислити катет та площу прямокутного трикутника за заданими гіпотенузою та другим катетом.
23. Обчислити сторону та площу $S=d^2/2$ квадрата, якщо відома його діагональ d.

24.Обчислити площу бічної поверхні $S=\pi rl$ та об'єм $V=\pi r^2h/3$ конуса за заданою висотою h, твірною l та радіусом основи r.

25.Поїзд їхав t_1 год зі швидкістю v_1 км/год, t_2 год зі швидкість v_2 і t_3 год зі швидкістю v_3. Визначити пройдені шляхи з різною швидкістю і повний шлях.

Розгалуження.

1. Задано довжини сторін трикутника. Складіть програму, за допомогою якої визначте, чи є трикутник рівнобедреним, рівностороннім або різностороннім.
2. Задано три числа *a, b, c*. Складіть програму, за допомогою якої визначте, чи існує трикутник з такими довжинами сторін. Якщо так, то знайдіть його периметр.
3. Задано три числа *a, b, c*. Складіть програму, за допомогою якої визначте, чи існує трикутник з такими довжинами сторін. Якщо так, то знайдіть його площу.
4. Задано довжини суміжних сторін паралелограма й кут між ними. Складіть програму, за допомогою якої визначте, чи є цей паралелограм квадратом, ромбом, прямокутником або не є ні однієї з перерахованих фігур.
5. Задано сторону рівностороннього трикутника й радіус кола. Складіть програму, за допомогою якої визначте, що більше - площа трикутника або площа кругу.
6. Задано сторону рівностороннього трикутника й сторона квадрата. Складіть програму, за допомогою якої визначте, що більше - площа трикутника або площа квадрата.
7. Задано сторону рівностороннього трикутника й сторона квадрата. Складіть програму, за допомогою якої визначте, що більше - периметр трикутника або периметр квадрата.
8. Задано сторону куба й радіус кулі. Складіть програму, за допомогою якої визначите, що більше - об'єм куба або об'єм кулі.
9. Задано виміри прямокутного паралелепіпеда й діаметр кулі. Складіть програму, за допомогою якої визначите, що більше - об'єм паралелепіпеда або об'єм кулі.
10. Задано сторону квадрата й радіус кола. Складіть програму, за допомогою якої визначите, що більше - площа квадрата або площа кругу.

Вибір. Скласти програму для розв'язування наведеного нижче завдання двома способами, використовуючи: 1) команду case; 2) команду if. Придумати і задати вхідні дані так, щоб вибір був з 4-7 альтернатив.

1. Ввести номер учня зі списку. Вивести його прізвище.
2. Є дані про автомобілі чотирьох моделей. Як вхідне дане ввести номер моделі і отримати характеристики: рік випуску і ціну.
3. Ввести номер поїзда. Вивести назву пункту призначення.
4. Ввести першу букву назви країни. Вивести назву її столиці.
5. Ввести номер дня тижня. Вивести його назву.
6. Ввести номер трамвая. Вивести назви його кінцевих зупинок.
7. Ввести першу букву назви країни. Вивести назву континента.
8. Ввести номер місяця. Вивести назву пори року.
9. Ввести номер учня у списку. Вивести його ім'я.

Цикли.

1. Складіть програму, за допомогою якої знайдіть добуток:
   а) перших 500 натуральних чисел;
   б) тризначних чисел, кратних 11.
2. Складіть програму, за допомогою якої знайдіть суму:
   а) непарних тризначних чисел;
   б) парних п'ятизначних чисел;
   в) парних чотиризначних чисел;
   г) перших 500 натуральних чисел, кратних 3;
   д) перших 300 натуральних чисел, кратних 15;
   е) тризначних чисел, кратних 7;
   ж) чотиризначних чисел, кратних 3.
3. Складіть програму, за допомогою якої знайдіть суму 3 + 9 + 27 + 81 + ... + 19683.
4. Знайдіть суму та середнє арифметичне 10 довільних чисел, які вводяться з клавіатури.

5. З клавіатури вводяться 20 довільних чисел. Знайдіть суму тільки від'ємних доданків.

6. Складіть програму для знаходження суми чисел, кратних 7, які розташовані в інтервалі (100, 400).

7. Вводиться послідовність із N цілих чисел. Знайти:

А) найбільше число;

Б) найменше число;

В) суму всіх додатніх чисел;

Г) кількість нулів;

Д) суму всіх відємних чисел.

8. Протабулюйте функцію y=x·sin4x-cos2x на проміжку [-4;4] з кроком 0,2 і обчисліть суму квадратів максимального і мінімального значень функції.

9. Протабулюйте функцію y=sin5xcosx на проміжку [-2;2] з кроком 0,2 і обчисліть середнє арифметичне від'ємних значень.

10. Протабулюйте функцію z=cos(2x-3y), змінюючи x на проміжку [0;2] з кроком 0,25, а y на проміжку [0;1] з кроком 0,2. обчисліть середнє арифметичне додатних значень функції.

«Основи алгоритмізації та програмування мовою Паскаль»

Передмова

Посібник призначений для використання в якості робочого конспекту з курсу «Основи алгоритмізації та програмування мовою Паскаль». Будова посібника відповідає програмі курсу, психологічним та фізіологічним особливостям учнів 8 класу. Обєм кожного пункту відповідає одному уроку.

Розділи посібника містять необхідний об'єм теоретичного матеріалу, докладно розібрані приклади, завдання для самостійної роботи та домашні завдання. Що дає змогу вивчати матеріал як за допомогою вчителя, так і самостійно

Зміст

Основи алгоритмізації

Алгоритми. Властивості алгоритмів.

„Алгоритм не розкіш, а засіб досягнення мети"

Епіграф до уроку:

Коль кругом все будет мирно,
Так сидеть он будет смирно;
Но лишь чуть со стороны
Ожидать тебе войны,
Иль набега силы бранной,
Иль другой беды незваной,
Вмиг тогда мой петушок
Приподымет гребешок,
Закричит и встрепенется
И в то место обернется.

А.С.Пушкин

Багато хто вважає, що інформатика потрібна тільки для того, щоб навчитися працювати на комп'ютерах. Але цю помилкову думку ми постараємося спростувати на нашому уроці.

Кожна людина щодня зустрічається з безліччю задач від найпростіших і добре відомих до дуже складних. Для багатьох задач існують визначені правила (інструкції, команди), що пояснюють виконавцю, як розв'язувати дану проблему. Ці правила людина може вивчити чи заздалегідь сформулювати сама в процесі розв'язування задачі. Чим точніше описані правила, тим швидше людина опанує ними і буде ефективніше їх застосовувати. У нашому житті ми постійно складаємо опис деякої послідовності дій для досягнення бажаного результату, тому поняття алгоритму не є для нас чимось новим і незвичайним. Так, ранком мама перед твоїм виходом до школи, дає вказівку: "Коли прийдеш зі школи, відразу пообідай і вимий посуд. Після цього підмети підлогу, сходи в магазин і можеш трохи погуляти. Гуляти дозволяю не більше години, а потім відразу за уроки".

Ця інструкція складається з послідовності окремих вказівок, що і визначають твою поведінку після повернення зі школи. Це і є алгоритм.

Кожний з нас використовує сотні різних алгоритмів. Спробуйте згадати деякі з них (алгоритми виконання арифметичних дій, розв'язування задач, прибирання квартири, миття посуду, готування їжі - рецепти тощо). Отже, після обговорення кількох прикладів алгоритмів, давайте спробуємо сформулювати визначення, що ж таке алгоритм.

Саме слово алгоритм походить від algorithmi – латинської форми написання імені великого математика IX ст. аль-Хорезмі, який сформулював правила виконання арифметичних дій. Спочатку під алгоритмами і розуміли тільки правила виконання чотирьох арифметичних дій над багатоцифровими числами. В подальшому це поняття стали використовувати взагалі для позначення послідовності дій, які приводять до розв'язання задачі.

Алгоритмом називають зрозуміле і точне розпорядження виконавцю про виконання послідовності дій, спрямованих на досягнення зазначеної мети чи на вирішення поставленої задачі.

В цьому означенні використовується поняття "виконавець". Що це означає? Під виконавцем алгоритму ми розуміємо будь-яку істоту (живу чи неживу), яка спроможна виконати алгоритм. Все залежить від того, якої мети ми намагаємося досягнути. Наприклад: риття ями (виконавці - людина або екскаватор), покупка деяких товарів (один із членів родини), розв'язування математичної задачі (учень або комп'ютер) тощо.

Поняття алгоритму в інформатиці є фундаментальним, тобто таким, котре не визначається через інші ще більш прості поняття (для порівняння у фізиці - поняття простору і часу, у математиці - точка).

Будь-який виконавець (і комп'ютер зокрема) може виконувати тільки обмежений набір операцій (екскаватор копає яму, вчитель вчить, комп'ютер виконує арифметичні дії). Алгоритмічне мислення допомагає чітко побачити кроки, що ведуть до мети, замітити всі перешкоди і уміло їх обійти.

Тому алгоритми повинні мати певні властивості, разом з тим, не кожна інструкція або послідовність дій може називатися алгоритмом.

Отже, сформулюємо основні **властивості алгоритму**.

1. Зрозумілість. Щоб виконавець міг досягти поставленої перед ним мети, використовуючи даний алгоритм, він повинен уміти виконувати кожну його вказівку, тобто розуміти кожну з команд, що входять до алгоритму.

Наприклад: Мамі потрібно купити в магазині їжу. Виконавцем цього алгоритму може бути хтось із родини: батько, син, бабуся, маленька дочка тощо. Зрозуміло, що для тата достатньо сказати, які купити продукти, а далі деталізувати алгоритм не потрібно. Дорослому сину-підлітку необхідно детальніше пояснити в яких магазинах можна придбати потрібний товар, що можна купити замість відсутнього товару і таке інше. Маленькій дочці алгоритм необхідно деталізувати ще більше: де взяти сумку, щоб принести товар, яку решту грошей необхідно повернути з магазину, як дійти до магазину і як там поводитись (якщо дитина вперше йде за покупками).

2. Визначеність (однозначність). Зрозумілий алгоритм все ж таки не повинен містити вказівки, зміст яких може сприйматися неоднозначно. Наприклад, вказівки "почисти картоплю", "посоли за смаком", "прибери в квартирі" є неоднозначними, тому що в різних випадках можуть призвести до різних результатів. Крім того, в алго-

ритмах неприпустимі такі ситуації, коли після виконання чергового розпорядження алгоритму виконавцю не зрозуміло, що потрібно робити на наступному кроці. Наприклад: вас послали за яким-небудь товаром у магазин, та ще попередили "без (хліба, цукру і таке інше) не повертайся", а що робити, якщо товар відсутній?

Отож, *точність* - це властивість алгоритму, що полягає в тім, що алгоритм повинен бути однозначно витлумачений і на кожному кроці виконавець повинен знати, що йому робити далі.

3. Дискретність. Як було згадано вище, алгоритм задає повну послідовність дій, які необхідно виконувати для розв'язання задачі. При цьому, для виконання цих дій їх розбивають у визначеній послідовності на прості кроки. Виконати дії наступного розпорядження можна лише виконавши дії попереднього. Ця розбивка алгоритму на окремі елементарні дії (команди), що легко виконуються даним виконавцем, і називається дискретністю.

4. Масовість. Дуже важливо, щоб складений алгоритм забезпечував розв'язання не однієї окремої задачі, а міг виконувати розв'язання широкого класу задач даного типу. Наприклад, алгоритм покупки якого-небудь товару буде завжди однаковий, незалежно від товару, що купується. Або алгоритм прання не залежить від білизни, що переться, і таке інше. Отож, під масовістю алгоритму мається на увазі можливість його застосування для вирішення великої кількості однотипних завдань.

5. Результативність. Взагалі кажучи, очевидно, що виконання будь-якого алгоритму повинне завершуватися одержанням кінцевих результатів. Тобто ситуації, що в деяких випадках можуть призвести до так званого "зациклення", повинні бути виключені при написанні алгоритму. Наприклад, розглянемо таку ситуацію: роботу дано завдання залишити кімнату (замкнутий простір), не виконуючи руйнівних дій. У цьому випадку, якщо роботу не дати вказівки відкрити двері (що, можливо, закриті), то спроби залишити приміщення можуть бути безуспішними.

6. Ефективність - кожний крок алгоритму повинен бути виконаний точно за скінчений проміжок часу.

Для роботи багатьох програм необхідно задавати початкові значення. Ці значення передаються в алгоритм за допомогою аргументів.

***Аргументи* - це величини, значення яких необхідно задати для виконання алгоритму.** Правда, деколи зустрічаються алгоритми, що не вимагають ніяких початкових значень для свого виконання. Пізніше буде нагода познайомитися з такими алгоритмами. Однак, немає жодного алгоритму, що не дає ніякого результату. Дійсно, який же зміст у такому алгоритмові? Прикладом різноманітності результатів роботи програм є ігрові комп'ютерні програми. Одержувана ними під час роботи закодована інформація певним чином перетворюється у графічні та звукові образи.

***Результати* - це величини, значення яких одержуються внаслідок виконання алгоритму**.

При складанні багатьох алгоритмів виникає необхідність окрім аргументів та результатів використовувати ще додаткові величини. Введення в алгоритм таких величин залежить від самого автора алгоритму.

Проміжні величини — це величини, які додатково вводяться в ході розробки алгоритму.

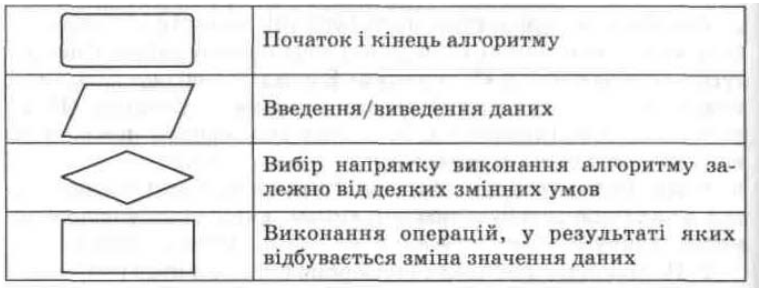
Тепер залишається з'ясувати, яким чином можна подати алгоритм виконавцю. Існує кілька методів запису алгоритмів, вибір яких залежить від виконавця та того, хто його задає.

Перший спосіб - це ***словесний опис*** алгоритму. Сьогодні на уроці розібрано вже кілька алгоритмів, і всі вони подавалися виконавцю за допомогою словесного опису.

Другий спосіб - це подача алгоритму у вигляді ***таблиць, формул, схем, малюнків*** тощо. Наприклад, всіх вас вчили в дитячому садочку правилам поведінки на дорозі. І найкраще діти, вочевидь, сприймають алгоритм, що поданий у вигляді схематичних малюнків. Дивлячись на них, дитина, а потім і доросла людина, відпрацьовує ту лінію поведінки, що їй пропонується. Аналогічно можна навести приклади алгоритмів, що записані у вигляді умовних позначок на купленому товарі, щодо його користування (заварювання чаю, прання білизни тощо). В математиці наявність формул дозволяє розв'язати задачу, навіть "не використовуючи слів".

Третій спосіб - запис алгоритмів за допомогою ***блок-схеми***. Цей метод був запропонований в інформатиці для наочності представлення алгоритму за допомогою набору спеціальних блоків. Основні з цих блоків наступні:

Четвертий спосіб - ***навчальні алгоритмічні мови*** (псевдокоди). Ці мови мають жорстко визначений синтаксис і вже максимально наближені до машинної мови (мови програмування). Але створені вони з навчальною



метою, тому мають зрозумілий для людей вигляд. Таких псевдокодів зараз існує велика кількість, починаючи з графічних середовищ "Алгоритміка", "Роботландія", "Лого-світи", "Черепашка" тощо і закінчуючи текстовими "національними" реалізаціями алгоритмічних мов, подібних до Паскаля. Ці псевдокоди мають програмну реалізацію і дуже широко застосовуються на етапі навчання основам програмування.

П'ятий спосіб максимально наближений до комп'ютера - це ***мови програмування***. Справа в тому, що найчастіше в практиці виконавцем створеного людиною алгоритму являється машина і тому він повинен бути написаний мовою, зрозумілою для комп'ютера, тобто мовою програмування

Приклади алгоритмів

3. Знайти найбільший спільний дільник двох натуральних чисел m і n (алгоритм Евкліда). Складемо алгоритм розв'язання цієї задачі, який базується на тій властивості, що якщо m>n, то найбільший спільний дільник чисел m,n такий самий, як і чисел m-n, n.

Алгоримт буде таким:

5) якщо числа рівні, то взяти любе з них за відповідь, в іншому випадку продовжити виконання алгоритму;
6) визначити більше із чисел;
7) замінити більше число різницею більшого і меншого чисел;
8) почати алгоритм спочатку.

4. Алгоритм «відгадування» задуманого числа. Нехай хто-небудь задумає довільне натуральне число. Йому пропонується провести з цим числом слідуючі дії і потім повідомити результат:

4) помножити задумане число на 5;
5) додати 8;
6) суму помножити на 2.

Необхідно за результатом «відгадати» задумане число.

Розв'язання даної задачі зводиться до розв'язання рівняння $(x\cdot5+8)\cdot2=a$, де x – невідоме задумане число, а – отриманий результат.

«Відгадування» x можна доручити виконавцю, зовсім незнайомому із змістом задачі. Для цього достатньо повідомити йому слідуючий алгоритм:

3) відняти від результата 16;
4) в отриманій різниці відкинути крайню праву цифру, отримане число і буде шуканим.

Виконуючи алгоритм, виконавець може не вникати в зміст того, що він робить, і разом з тим отримати потрібний результат. У цьому випадку говорять, що виконавець діє формально.

Домашнє завдання:

- прочитати сторінки запропонованого підручника;
- вивчити означення, що прочитані на лекції (що таке алгоритм, властивості алгоритму, способи подачі алгоритму);
- придумати будь-який алгоритм на побутову тему (кулінарний, прибирання кімнати, виконання уроків тощо);
- продумати варіанти, коли в запропонованих алгоритмах можуть не виконуватися властивості алгоритмів.
- записати алгоритм знаходження середини відрізка за допомогою циркуля лінійки.
- скільки разів буде виконуватися крок 3 алгоритма Евкліда для m=100, n=18?

Виконавці алгоритмів

1. Яке походження терміна «алгоритм»?
2. Що ми розуміємо під поняттям «алгоритм»?
3. Що таке допустимі команди виконавця?
4. Які є способи опису алгоритмів?
5. Які властивості повинен мати алгоритм?
6. Що означає скінченність (дискретність) алгоритму?
7. Що таке формальність алгоритму?
8. Що означає масовість алгоритму?

1. Виконавець алгоритму

Виконавцем алгоритму може бути людина, машина, комп'ютер, система людина-машина, верстат-автомат, робот тощо, яких «навчено» виконувати вказівки алгоритму.

Характеристики виконавця

Середовище – «місце проживання» виконавця.

Припустимі дії – обмежений набір дій, що вміє виконувати даний виконавець. Описати виконавця – значить вказати його припустимі дії. Досяжні цілі – результати, що виконавець може одержати за допомогою своїх припустимих дій.

Система команд виконавця – суворо заданий список вказівок, як виконати визначені дії. Виконавця можна представити у виді пристрою з кнопками, де кожна кнопка відповідає одній команді. Натискання кнопки означає виклик команди.

Відмова – виникає при виклику команди в неприпустимому для даної команди стані середовища.

При побудові алгоритму часто виникає необхідність пояснити виконавцю деякі складні дії, якщо їх виконання не входить в систему команд виконавця. Наприклад, перший раз даючи дитині завдання пришити ґудзик до плаття, їй треба пояснити, як необхідно підбирати нитки для шиття, як вдягати нитку в голку, як тримати голку та ґудзик при роботі, яка різниця між пришиванням ґудзика до тоненької сорочки та товстої куртки (в другому випадку ґудзик робиться на "ніжці"). В подальшому такі пояснення будуть вже зайві, бо алгоритм "пришивання ґудзика" стає вже командою в системі команд виконавця "дитина".

Взагалі кажучи кожна дія людини (якщо вона її може виконати) може вважатися командою її "системи команд", хоча колись, на етапі навчання, учитель або хтось інший ретельно пояснював, яку треба виконати послідовність дій, щоб досягти поставленої мети.

Узагальнюючи сказане, можна сказати, що кінець кінцем кожну задачу можна вважати окремою командою виконавцю, якщо його навчено виконувати поставлене завдання. Якщо ж виконавець не знає, як розв'язувати запропоновану задачу, виникає потреба розкласти її на такі підзадачі, що являються "посильними" для виконання, тобто входять до системи команд виконавця. Продовжуючи цей процес, остаточно отримують алгоритм, що складається з простих команд, зрозумілих виконавцю, або остаточно переконуються, що дана задача непосильна для вибраного виконавця, тому що в його системі команд не існує необхідних для цього команд. Наприклад, як би ми не деталізували алгоритм побудови багатоповерхової будівлі для дитини, задача кінець кінцем являється для неї непосильною.

Запропонований підхід до конструювання алгоритмів називається *методом покрокової деталізації зверху вниз*. Вочевидь, що при такому підході кожна операція остаточно буде подана у вигляді лише одного з трьох типів базових структур алгоритмів - лінійної (в літературі часто ця структура називається слідування), розгалуження або повторення (циклу). Степінь деталізації алгоритму при цьому сильно залежить від того, на якого виконавця його орієнтовано.

Досить складну конструкторську задачу неможливо розв'язати без поступового заглиблення в деталі. Подумайте, наприклад, як розробляється конструкція сучасного теплохода, автомобіля або літака.

Розглянутий принцип конструювання алгоритмів не залежить від конкретних особливостей поставленої задачі та вибору виконавця. Проте набір команд системи команд вибраного виконавця суттєво впливає на ступінь деталізації алгоритму та, кінець кінцем, на його структуру.

Візьмемо, наприклад, простий алгоритм "переходу через вулицю". Взагалі для кожної дитини можна вважати це командою, що входить до її "системи команд". Але кожен пам'ятає, як в дитинстві батьки та вихователі неодноразово повторювали: якщо в місті, де необхідно перейти вулицю, є підземний перехід, скористуйся ним, якщо немає, відшукай місце, де є світлофор, і перейди вулицю, користуючись правилами; якщо немає ні підземного переходу, ні світлофора... (далі діти самостійно продовжать це алгоритм). Однак, навіть в цьому алгоритмі є необхідність дещо деталізувати. Наприклад, що значить "перейди вулицю, користуючись правилами", при наявності світлофора? А якщо алгоритм складається для зовсім маленької дитини, то що таке світлофор і як його шукати? А якщо виконавець взагалі прибулець з інших світів? Що таке "зелений", "червоний", "жовтий"? Що таке підземний перехід? Перелік питань можна доповнити нескінченою послідовністю непорозумінь.

Алгоритми, що складаються для розв'язування окремих підзадач основної задачі, називаються допоміжними.

Вони створюються при поділі складної задачі на прості або при необхідності багаторазового використання одного ж того набору дій в одному або різних алгоритмах. Допоміжний алгоритм повинен мати тільки один вхід та один вихід, причому того, хто користується ним, зовсім не цікавить, як реалізований цей алгоритм. Головне, щоб всі команди, що входять до складу допоміжного алгоритму входили до системи команд обраного виконавця. Зверніть увагу ще на те, що в реальному житті допоміжні алгоритми можуть виконувати, навіть, зовсім інші виконавці. Наприклад, якщо батьки вдома вирішили зробити ремонт, це не значить, що вони самостійно повинні зробити собі шпалери та клей. Алгоритми виробництва матеріалів існують і їх хтось виконує, а ми тільки користуємось результатами їх роботи.

Таким чином, можна вважати допоміжний алгоритм своєрідним "чорним ящиком", на вхід якого подаються деякі вхідні дані, а на виході ми отримуємо очікуваний результат. Головне чітко домовитись про правила оформлення вхідних даних та вигляд результату. Невиконання домовленостей може привести до збою у виконанні допоміжного алгоритму або до отримання неочікуваного результату.

Описаний метод послідовної деталізації лежить в основі технології структурного програмування і широко застосовується при використанні таких мов програмування, як Паскаль, С, С++ та інших.

При описуванні програми для комп'ютера мовами високого рівня допоміжні алгоритми реалізовуються у вигляді підпрограм. Правила опису, звернення до них та повернення в точку виклику визначаються конкретною мовою програмування. Для зручності часто використовувані підпрограми можна об'єднувати в бібліотечні модулі і при необхідності підключати їх в свої програми.

Домашнє завдання:

- вивчити означення, що прочитані на лекції (що таке допоміжний алгоритм, в чому сутність метода покрокової деталізації та проектування зверху вниз);
- придумати будь-який алгоритм, в якому в залежності від виконавця необхідна різний степінь деталізації;
- продумати приклади алгоритмів, для яких будь-який степінь деталізації не дозволить виконати їх заданим виконавцем.
- записати алгоритм, за допомогою якого можна перейти вулицю у місці переходу. На яких виконавців розрахований цей алгоритм?

Загальні правила алгоритмічної мови

1. Що ми називаємо алгоритмом? Як виник термін «алгоритм»?
2. Наведіть власний приклад алгоритму.

3. Запишіть основні властивості алгоритмів.
4. Що розуміється під формальним виконанням алгоритму?
5. Хто або що може бути виконавцем алгоритму?
6. Які характеристики виконавця вам відомі?

1. Поняття про навчальну алгоритмічну мову.

Алгоритмічна мова – це система позначень і правил для однакового і точного запису алгоритмів і їх виконання. Алгоритмічна мова визначає способи запису тексту алгоритму (синтаксис алгоритму) і правила інтерпретації записаного тексту виконавцем (семантика мови). З одного боку, алгоритмічна мова близька до звичайної, з іншого – включає в себе і математичну символіку: числа, позначення величин і функцій, знаки операцій. Правила алгоритмічної мови лежать в основі мов програмування для ПК. Тому вивчення алгоритмічної мови допоможе вам швидше опанувати мову програмування.

2. Алфавіт мови

Алфавіт мови — це символи, які дозволено до використання в мові. В навчальній алгоритмічній мові використовуються:

• 33 літери українського алфавіту;
• 26 латинських літер (від *A* до *2);*
• 10 арабських цифр;
• 28 спеціальних знаків (?, !, #, $,%тощо).

Звичайно ж, у конкретних мовах програмування допускається дещо інший набір символів (наприклад, алфавіт мови Паскаль не містить літер українського алфавіту). Кожний символ алфавіту мови програмування має свій числовий код. Яким кодам відповідають символи, наведено в таблицях кодування символів.

3. **Синтаксис мови**

Синтаксис мови — це правила написання команд, службових слів,розділових знаків.

Розділовими знаками можуть бути такі символи:

1) ; —розділовий знак між командами;
2) . — розділовий знак між цілою й дробовою частинами числа;
3)(«,») —використовуються у записі виразів підпрограм;
4)[«, «] — для запису індексованих змінних;
5), — розділовий знак між елементами списку або індексами масиву;
6) : — є елементом синтаксису деяких команд, а також використовується для задання розмірів масиву. У мові програмування Паскаль для задання розмірів масиву використовується дві крапки, розміщені поруч...

4. **Елементи мови**

Елементами мови є:

Символи — основні неподільні знаки, за допомогою яких записуються тексти.

Службові слова — скорочення деяких слів, за допомогою яких записуються алгоритми. Наприклад: АЛГ, АРГ, РЕЗ, ПОЧ, КІН та ін.

Команди — дії алгоритму. Вони бувають прості й складені.

5**. Об'єкти мови**

Об'єктами мови є:

3. Константи, змінні.
4. Допоміжні алгоритми (підпрограми — функції або процедури).

Константи — це постійні величини, які визначаються на початку програми та не змінюють свого значення в процесі розв'язання завдання.

Числова константа — це деяке число. Числа можуть бути цілими або дробовими й подаватися у звичайній або експонентній формі.

У звичайній формі запису дробового числа ціла частина від дробової відділяється крапкою, а не комою. Така форма запису числа називається *записом з фіксованою крапкою,* наприклад:

2, 2.4, -7.12, 0.3.

Для запису надто великих або надто малих чисел зручніше застосовувати експонентну форму запису. її називають *записом із плаваючою крапкою.* Вона складається з мантиси (можливо зі знаком), літери *E* десяткового порядку (цілого числа, можливо зі знаком), наприклад:

2.1E-5, IE6, 1E-9

Текстова константа *— це будь-яка послідовність символів, узята в лапки «"» або апострофи « '» (у мові Паскаль можливий тільки останній варіант)*

Усередині текстової константи можуть бути й лапки, але тоді їх також треба взяти в лапки. Приклади текстових констант:

"мама",
"1234",
"Корабель ""Ластівка""",
"Справа *306",
"Значення виразу =".

Тому надалі ми подаватимемо текстові константи в лапках або апострофах.

Змінні *— це величини, які можуть змінювати своє значення в процесі розв'язування завдання.* ***Змінна*** *та* ***комірка*** *в програмуванні — поняття тотожні, оскільки кожній змінній у пам'яті комп'ютера виділяється місце, яке називається* ***коміркою.***

Змінні називаються ***ідентифікаторами.*** *Ім'я змінної має обов'язково починатися на літеру, а далі може йти послідовність літер або цифр та знаків підкреслення.*

Приклади ідентифікаторів змінних:

• а, аі, **в2а, сума,** з, (у Паскалі україномовні ідентифікатори заборонені);

Імена змінним дає автор алгоритму. Ім'я змінної бажано добирати таким чином, щоб воно за змістом підкреслювало її призначення.

Не можна давати змінним імена, що збігаються з іменами службових слів, зарезервованих у цій мові програмування.

Кожна змінна повинна мати певне значення. Якщо в процесі виконання алгоритму змінна величина не одержала конкретного значення, то її значення буде непередбаченим.

Алгоритм, який використовується у складі іншого алгоритму, називають ***допоміжним алгоритмом*** *стосовно алгоритму, в якому він використовується.*

Вираз, *«алгоритм, який використовується у складі іншого алгоритму»,* необхідно розуміти так:

• допоміжний алгоритм може бути у складі основного алгоритму;

• допоміжний алгоритм може подаватися у вигляді окремого файла на диску, а в основному алгоритмі встановлюється з ним зв'язок.

У будь-якій мові програмування є *фіксований набір стандартних функцій.* Однак за необхідності його можна розширювати, створюючи власні функції користувача. За допомогою об'єктів мови створюються вирази.

Вираз *— це сукупність об'єктів мови, пов'язана деякими операціями для обчислення одного значення певного типу.*

Вирази бувають:

- арифметичні;
- алгебраїчні;
- логічні;
- літерні (рядкові, текстові).

Арифметичні вирази *— це числа, з'єднані знаками арифметичних операцій (можливо, із застосуванням дужок).*

Алгебраїчні вирази *— це числа, літери, функції, з'єднані знаками арифметичних операцій (можливо, із застосуванням дужок).*

Логічні вирази *— це умови. Умови бувають прості й складені.* ***Прості умови*** *складаються з одного відношення між величинами.*

Складені умови *складаються з кількох простих умов, з'єднаних між собою логічними* ***операціями:***

І (and) — кон'юнкція (логічне множення);

Або (or)— диз'юнкція (логічне додавання);

Не (not)— заперечення.

Приклади складених умов:

X>2 І X<4, a<3 АБО a<5, НЕa = 0.

Мовою Паскаль відповідно умови пишуться так:

(x > 2) **and** (x < 4), (a < 3) or (a<5), not(a) = 0 Значенням логічного виразу є **«Істина»** або **«Хибність».**

Літерні вирази *— це текстові константи, змінні, функції, що опрацьовують текст, з'єднані знаком плюс «+».Знак плюс означає операцію склеювання (конкатенацію). Значенням літерного виразу є* ***текстовий рядок.***

Структура алгоритму:

алг заголовок алгоритму (список параметрів із вказівкою їх типів) арг список аргументів рез список результатів поч список проміжних результатів із вказівкою їх типів серія вказівок кін

Лінійний алгоритм - алгоритм, який містить лише вказівки про безумовне виконання деякої операції, без повторень або розгалужень (просте слідування).

Приклад 1. Записати алгоритмічною мовою алгоритм знаходження середини відрізка, якщо в систему виконавця входять звичні дії із циркулем і лінійкою.

алг розподіл відрізка *АВ* навпіл

поч поставити ніжку циркуля в точку *А,* встановити розчин циркуля рівним довжині відрізка *АВ,* провести коло, поставити ніжку циркуля в точку *В,* провести коло, через точки перетину кіл провести пряму, відзначити точку перетинання цієї прямої з відрізком *АВ*

кін

Запишіть алгоритми для розвязування задач:

6) деяке задане число х збільшити у 16 разів, використовуючи лише операції множення на 2 і додавання;
7) деяке задане число х зменшити на 8, використовуючи лише операцію віднімання 1 і 3;
8) складіть алгоритм обчислення площі трапеції;
9) запишіть алгоритм обчислення шляху, який долає автомобіль зі швидкістю v за час t.
10) на інший берег річки треба перевезти човном вовка, козу та капусту, виконуючи умову: не можна залишати разом у човні чи на березі вовка і козу, або козу і капусту (за один раз можна перевозити не більше одного об'єкта).

Домашнє завдання:

a.вивчити означення, що прочитані на лекції ;
b.записати алгоритм обчислення площі прямокутного трикутника;
c.задано два кути трикутника, складіть алгоритм, який визначає величину третього кута трикутника;
d.задано діаметр кола, складіть алгоритм, який визначає довжину кола та площу круга.

Величини. Вказівка присвоювання

1. Поняття величини.

Величина - це об'єкт, який має стале або змінне значення.

Величини діляться на змінні й сталі. *Сталою* називається величина, значення якої не змінюється в процесі виконання алгоритму, а залишається тим самим, зазначеним у тексті алгоритму (наприклад, 15; 2,4; 3,14 і т.д.). *Змінною* називається величина, значення якої змінюється в процесі виконання алгоритму.

При написанні алгоритму для змінних величин вводяться позначення аналогічно позначенням змінних у курсі алгебри. Таке позначення змінної величини в алгоритмічній мові називається ***ім'ям*** величини.

При виконанні алгоритму в кожний момент часу величина звичайно має деяке значення. Воно називається ***поточним*** значенням. У процесі виконання алгоритму величина може не одержати конкретного значення. Така величина називається ***невизначеною***.

У шкільному курсі математики й фізики найчастіше зустрічаються числові величини, значеннями яких є натуральні, цілі, дійсні числа. Однак в алгоритмах настільки ж часто зустрічаються й нечислові величини: слова, таблиці, списки, тексти, графіки, геометричні фігури й т.д. У курсі математики й фізики величини найчастіше позначаються однією буквою латинського або грецького алфавіту. Оскільки при застосуванні ПК і при записі алгоритмів і програм розмаїтість величин дуже велика, в алгоритмічній мові прийнято використовувати як імена величин довільні букви, буквосполучення й будь-які слова, що пояснюють зміст і призначення величини в алгоритмі.

Величини, значеннями яких є слова або текст, називаються ***літерними***.

Для того щоб виділити текст, що є значенням літерної величини, значення літерних величин беруться в лапки наприклад „немає розв'язку".

Як ми бачимо, величини можуть мати різний тип. Вони можуть бути натуральними, цілими, дійсними, літерними, відповідно до того, що може бути їхнім значенням. Скорочено типи змінних позначаються словами **нат** (натуральний), **ціл** (цілий), **дійсн** (дійсний), **літ** (літерний).

Основні характеристики величин:

Ім'я (ідентифікатор)	Ти	В	Значення
Правила: - тільки латинські літери, цифри, - спершу літера, потім цифра, - не застосовувати службових	Ціл а. дійсн К. літ Н.	Змінна Константа	A:=A+5 Вираз - опис правила обчислення деякої величини. Лінійна форма

2. Оператор присвоювання

При програмуванні більше складних дій виникає задача обчислення чого-небудь без виведення на екран проміжних результатів. Наприклад, у виразі $(a-2b)^2 +7a - 2b$ спочатку зручно обчислити значення $a - 2b$, зберегти його, а потім, використовуючи отримане число, знайти кінцевий результат. У цьому випадку не обійтися без спеціального оператора присвоювання, записуваного за допомогою двох символів:

змінна := вираз;

Працює оператор присвоювання так — спочатку обчислюється значення арифметичного *виразу* шляхом підстановки всіх вхідних у нього змінних; результат записується в *змінну.* Ліворуч може перебувати тільки ім'я змінної, але в жодному разі не вираз. Наприклад:

a := 2 + 7;	у результаті одержимо значення a =9
c := a - 4;	c дорівнює 5
c := c + 3;	значення c збільшується на 3 і стає рівним 8
c + 1 : = 2 -	невірно, тому що ліворуч від знака присвоювання

a;	стоїть не змінна, а вираз!

Програмісти - початківці іноді плутають оператор присвоювання й математичний символ рівності, оскільки їхні позначення схожі один на одного. Це різні речі! Математик нас не зрозуміє, якщо ми напишемо *c = c* + 3, оскільки цей запис рівносильний неправильній тотожності 0 = 3. Двійка в щоденнику за такий вираз гарантована! Однак програміст порахує вираз виду c:= c + 3 нормальним, тому що, з його погляду, це не відношення рівності, а послідовність дій, що складається з обчислення виразу в правій частині оператора присвоювання і запису отриманого результату у відповідну комірку пам'яті замість старого значення до змінної с. У даному прикладі якщо до виконання оператора c:= c + 3 змінна з мала значення 5, те після його виконання вона буде мати значення 8.

змінна:=	K:=7.5; A:='слово'
змінна:= змінна	C:=B; B:=A; A:=C; (обмін значеннями)
змінна:= вираз	A:=5*B+C;

3. ЗАГОЛОВОК АЛГОРИТМУ

Запис усякого алгоритму починається із заголовка. От приклад заголовка алгоритму знаходження остачі від ділення двох натуральних чисел:

алг остача від ділення (нат *ділене,* нат *дільник,* нат *остача)*
арг *ділене, дільник*
рез *остача*

У цьому прикладі остача від ділення — назва алгоритму, *ділене, дільник,* і *остача* — імена величин. Перед кожним ім'ям величини зазначений її тип. Вихідними даними для алгоритму знаходження остачі від ділення двох натуральних чисел є величини *ділене, дільник* типу нат, а в результаті його виконання величина *остача* здобуває значення, рівне остачі ділення значень величин *ділене* й *дільник.* Величини, які є вихідними даними для алгоритму, називаються *аргументами.* Їхній список міститься після службового слова арг (аргумент). Результати алгоритму (у даному прикладі — величина *остача)* перераховуються після службового слова результат).

Загальний вигляд заголовка алгоритму такий:

алг назва алгоритму (список величин із вказівкою типів)
арг імена аргументів
рез імена результатів

Імена аргументів *і* результатів алгоритму перераховуються через кому.

В процесі розв'язування задачі спочатку необхідно визначитися з набіром даних, необхідних для розв'язання задачі, які називаються **вхідна інформація.** Саме розв'язання – це **обробка інформації –** тобто процес перетворення вхідної інформації у **результат.**

Приклад 6.1. Заголовок алгоритму Евклида:

алг знаходження найбільшого спільного дільника (НСД) (нат *M, N,* нат *НСД)*
арг *M, N*
рез *НСД*

1. Яка вхідна інформація потрібна для таких задач:
а) знайти середнє арифметичне значення трьох чисел;
б) змінити знак числа на протилежний;
в) знайти цілу частину числа;
г)знайти площу та периметр прямокутника;
д) перекласти слово на англійську мову.
2. Вкажіть тип величини, якщо її значення дорівнює:
а) 25; б) 3.5; в)'число'.
3. Визначить тип даних для величин:
а) маса людини в кг; б) кількість учнів у класі; в) назва книги
4. Чому дорівнює значення Y після виконання послідовності присвоювань:
а) x:=5;y:=1; y:=x;
б) y:=5; x:=8; y:=x;
в) y:=5; x:=8; y:=y*x;
г) x:=5; y:=5*x;
д) y:=1;a:=5-y; y:=y+2*a.

Тестові завдання

1. Вкажіть тип величини, якщо її значення дорівнює:
а) 25; б) 36.6; в) 'слово'.
2. З наведених значень виберіть припустимі для величин: а) цілого типу; б) дійсного типу; в) літерного типу:
-5; 3.7; 38; 'три'; 20.2; '23'; 14.
3. Визначите тип для величин:
а) вага людини; б) кількість учнів у класі; в) назва дня тижня.
4. Для величини КІЛЬКІСТЬ СТОРІНОК У КНИЗІ виберіть припустиме значення:
-46; 'дев'яносто'; 175; 65,1.
8. Наведіть кілька припустимих значень для величин: а) назва породи дерева; б) висота будинку; в) вага людини.

9. Визначте, дана величина є постійною чи змінною: а) кількість днів у тижні; б) назва природного супутника Землі; в) кількість днів у січні місяці; г) кількість днів у місяці.

10. Змінні x, y, z мають відповідно тип ціл, дійсн, літ. Виберіть запис із припустимими значеннями для x, y, z (з дотриманням порядку слідування):

а) 15.3; 6; 'п'ять';
б) 38; 26.04; 'сім';
в) 'так'; 18; 10.3.

10. Наведіть приблизний припустимий інтервал значень для величин: а) кількість днів у місяці; б) швидкість автомобіля.

11. Випишіть номера команд, у записі яких допущені помилки:

x:=a+b	y:='ні'	2*a:=b+1
2B:=9	a1:=7-d	c:=c+1

10. Запишіть розпорядження у видгляі команди присвоювання:

а) збільшити значення A на одиницю;

б) V привласнити значення суми величин K і N.

11. Знайдіть значення змінної A після виконання команди присвоювання: A:=(B+3)*2; якщо початкове значення змінної B дорівнює: а) 2; б) 1.5.

12. Знайдіть значення змінної c після виконання серії команд:

a:=8; b:=a*4; c:=b-28.

Домашнє завдання:

- вивчити означення, що прочитані на лекції;
- записати алгоритм одержання числа 512 із числа 2;
- записати у вигляді схеми алгоритму послідовність виконання дій при обчисленні таких виразів: $(x + y)^2$-x/y+10; x-y/(a+2); (a+в)/(a-в)/a*в.

Алгоритми з розгалуженнями

- Давайте пригадаємо, які існують форми запису алгоритмів.

-Які позначення використовуються для запису алгоритму у вигляді блок-схеми?

-Як записати у вигляді блок-схеми лінійний алгоритм?

-На вашу думку, чи достатньо знати тільки лінійну алгоритмічну структуру, щоб розв'язувати різноманітні задачі?

-Складіть алгоритм переходу вулиці, де встановлено світлофор.

-Зверніть увагу, що в цьому прикладі вам треба обрати дію в залежності від кольора світлофора, тобто в залежності від виконання певної умови, виконується одна, або інша послідовність дій.

Уявім собі…

Англія, Лондон, Бейкер-Стрит, будинок знаменитого детектива Шерлока Холмса. Увійдемо з дозволу хазяїна, присядемо біля каміна й послухаємо його діалог з доктором Ватсоном в оповіданні Артура Конан-Дойля «Строката стрічка»:

«....Так що ж ви про все це думаєте, Ватсон? - запитав Шерлок Холмс, відкидаючись на спинку крісла.

- По-моєму, це найвищою мірою темна й брудна справа.

- Досить брудна й досить темна. Але *якщо* наша гостя права, стверджуючи, що підлога й стіни в кімнаті міцні, так що через двері, вікна й коминкову трубу неможливо туди проникнути, виходить, її сестра в хвилину своєї таємничої смерті була зовсім одна...

- У такому випадку, що означають ці нічні свисти й дивні слова вмираючої?

- Уявити собі не можу.

- *Якщо* зіставити факти: нічні свисти, цигани, з якими в цього старого доктора такі близькі стосунки, натяки вмираючої на якусь стрічку й, нарешті, той факт, що місс Элен Стоунер чула металевий брязкіт, що міг видавати залізний засув від ставні... *Якщо* згадати до того ж, що доктор зацікавлений у запобіганні заміжжя своєї пасербиці, — я думаю, що ми напали на вірні сліди, які допоможуть нам розгадати цю таємничу подію.

- Але тоді при чому тут цигани?

- Гадки не маю...»

Зверніть увагу на виділене курсивом слово *якщо,* що кілька разів використовує Шерлок Холмс у своїх міркуваннях про здійснений злочин. Уживання цього слова має глибокий зміст — у ситуації невизначеності воно дозволяє направити хід міркувань по одному з декількох можливих шляхів. Слово *якщо* або пари *якщо... то* є незмінними супутниками логічних міркувань. Мабуть, мистецтво вживання цих слів і становить секрет геніальності Шерлока Холмса?

Відношення між величинами в якості умов

Як і в математиці, в алгоритмічній мові використовуються слідуючі знаки відношення між величинами:

Для числових величин

<	*Менше*	≥	*Не менше*	=	*Дорівнює*
>	*більше*	≤	*Не більше*	≠	*Не дорівнює*

Для літерних величин

= дорівнює *≠ не дорівнює*

В алгоритмах роботи з числовими та літерними величинами такі відношення між величинами і використовуються у якості простих умов.

2. Поняття розгалуження.

Розгалуження - це така форма організації дій, при якій в залежності від виконання або невиконання деякої умови здійснюється або одна, або друга послідовність дій.

алг об'їзд небезпечної ділянки
поч зменшити швидкість
якщо ремонт закінчено
то їхати 5 км по відремонтованому шосе
інакше їхати 10 км в об'їзд
все
зупинитися біля АЗС
кін

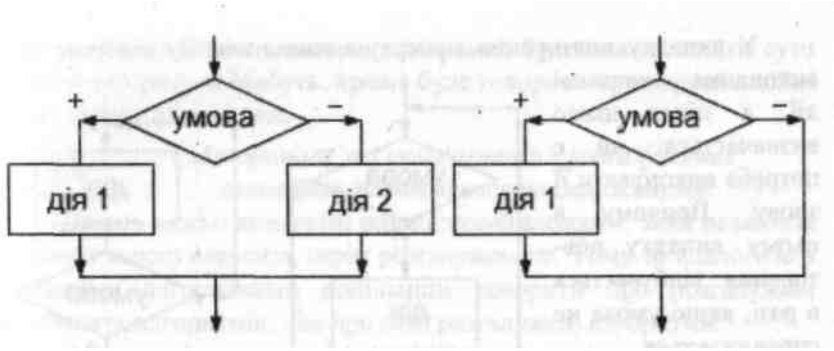
Умова - це є будь-яке твердження або запитання такого виду, що допускає лише дві можливі відповіді «так» або «ні».

Якщо відповідь на умову є позитивною, то виконується серія команд, записана після слова **то**, якщо ж відповідь негативна, - серія команд після слова інакше, а для скороченої форми - у цьому випадку нічого не виконується.

Після виконання серії команд виконавець переходить до наступної після розгалуження команди.

Якщо серія команд після слів **то** … інакше складається з більш, ніж однієї команди, то обов'язковими будуть слова **пс**, **кс** (початок та кінець серії), як-що у серії одна команда - ці слова необов'язкові. Будь-яка команда серії може знову бути командою розгалуження. У цьому випадку кажуть, що команди розгалуження вкладені одна в одну.

Повна форма розгалуження:	**Скорочена форма розгалуження**
Якщо < умова > **то серія 1** **інакше серія 2** **Все**	**Якщо < умова >** **то серія** **все**



Повна форма розгалуження:

Якщо на дворі холодно то
одягти пальто інакше одіти
куртку
Все

Скорочена форма розгалуження:

Якщо на дворі дощ
то взяти парасольку
Все

Приклад 1. *Складіть алгоритм знаходження більшого з трьох чисел.* Алгоритм має вигляд:

АЛГ Приклад 1(дійсн a,b,c,r)
АРГ a,b,c
РЕЗ r
ПОЧ
якщо a>b **то** r:=a
інакше r:=b
все
якщо r<c
то r:=c
все
ДРУКУВАТИ("Більше з трьох чисел дорівнює ",r)
КІН

Пояснения до алгоритму

Спершу відбувається порівняння двох будь-яких чисел і більше число вміщується в комірку результату (при цьому використовується повна форма команди розгалуження). Потім наступне число порівнюється із вмістом комірки результату і найбільше з них вміщується в комірку результату (при цьому використовується скорочена форма команди розгалуження).

2. Побудувати алгоритм розв'язання слідкуючої задачі: «Ракета запускається з точки на екваторі Землі зі швидкістю у (в км/с) за напрямом руху Землі по орбіті навкруги Сонця. Який буде результат цього запуску ракети в залежності від швидкості у?»

```
Алг запуск ракети (дійсн в, літ А)
    Арг у
    рез А
    поч
        якщо у < 7,8
            то А:= „ракета впаде на Землю"
                якщо у < 11,2
            то А:= «ракета стане супутником Землі»
        інакше якщо у <16,4
    то А:= «ракета стане супутником Сонця»
        інакше А : = «ракета покине Сонячну систему»
        все
    все
все
кін
```

3. Алгоритм розв'язання нерівності $ax>b$ (НЕР), де a й b — довільні дійсні числа

```
алг НЕР (дійсн a, b, з, літ у)
   арг a, b
   рез с, у
поч
   якщо a≠0
   то с.: = b/a
          якщо a>0
              то у: = „х> с"
              інакше у: = „х < с"
          все
       інакше
          якщо b < 0
              то у: = „х - будь-яке число"
              інакше у:= „розв'язків немає"
          все
   все
   кін
```

Завдання 1.

Визначте, які з тверджень істинні, які хибні, які зовсім не можуть використовуватися в якості умов, а для яких істинність залежить від значень змінних:

а) 5>6;
б) х2 +у2<3;
в) наш клас дружний;
г) число х парне;
д) три точки;
є) три точки належать одній прямій; є) колір олівця; з) тротуар мокрий; ж) А дорівнює В;

Завдання 2. Поясніть, чому неправильні такі алгоритми:

А) Якщо завтра викличуть	б) Якщо нога піднята
то вчити уроки	то опустити ногу
інакше піти на прогулянку	інакше Упадеш
Все	Все

Завдання 3.

Задано число х. Скласти алгоритм, за допомогою якого збільшіть число х на 1, якщо воно додатнє.

Завдання 4.

Задано число х. Скласти алгоритм, за допомогою якого збільшіть число х на 1, якщо воно додатнє, в іншому випадку зменшіть число х на 1.

Завдання 5.

Задано два числа x, y. Складіть алгоритм, за допомогою якого змінній z присвойте значення 1, якщо x менше y; змінній z присвойте значення нуль, якщо x = y; змінній z присвойте значення мінус одиниця, якщо x більше y.

Домашнє завдання:

вивчити означення, що прочитані на лекції;

Записати команди розгалуження для таких дій:

а) якщо число більше ніж 2, то піднести його до кубу, інакше – до четвертого степеня;

б) якщо ціна книжки не перевищує N гривень, купити цю книжку і сувенир, інакше купити тільки книжку;

в) задано два числа A і B, від більшого відняти 1, а менше помножити на 24

1. Дано довжини сторін трикутника. Складіть алгоритм, який визначає, чи є цей трикутник рівнобедреним, рівностороннім, різностороннім;

2. Дано дісне число x. Складіть алгоритм, за допомогою якого визначіть, чи є воно коренем рівняння: а) 5x+1=0; б) 15x-10,8=0

Команда повторення

Перенесемося з Англії кінця XIX століття на дві тисячі років тому - у Давню Грецію. Давня Греція - країна великих учених, поетів і легендарних героїв. Познайомимося з історією одного з них.

«...Сізіф, син бога володаря всіх вітрів Эола, був засновником міста Коринфа, що у найдавніші часи називався Эфірою.

Ніхто у всій Греції не міг рівнятися по підступництву, хитрості й спритності розуму із Сізіфом. Сізіф завдяки своїй хитрості зібрав незлічимі багатства в себе в Коринфі; далеко поширилася слава про його скарби.

Коли прийшов до нього бог смерті похмурий Танат, щоб звести його в сумне царство Аїда, то Сізіф, ще раніше відчувши наближення бога смерті, підступно обдурив бога Таната й закував його в окови. Перестали тоді на землі вмирати люди. Ніде не відбувалося велике пишне поховання; перестали приносити й жертви богам підземного царства. Порушився на землі порядок, заведений Зевсом. Тоді громовержець Зевс послав до Сізіфа могутнього бога війни Ареса. Він звільнив Таната з оковів, а Танат вивергнув душу Сізіфа й відвів її в царство тіней померлих.

Але й отут зумів допомогти собі хитрий Сізіф. Він сказав дружині своїй, щоб вона не ховала його тіла й не приносила жертви підземним богам. Послухалася чоловіка дружина Сізіфа. Аїд і Персефона довго чекали похоронних жертв. Все немає їх! Нарешті наблизився до трону Аїда Сізіф і сказав владиці царства померлих, Аїду:

- О, володар душ померлих, великий Аїд, рівний могутністю Зевсу, відпусти мене на світлу землю. Я велю дружині моєї принести тобі багаті жертви й повернуся обернено в царство тіней.

Так обдурив Сізіф владику Аїда, і той відпустив його на землю. Сізіф не повернувся, звичайно, у царство Аїда. Він залишився в пишному палаці своєму й весело банкетував, радіючи, що один із всіх смертних зумів повернутися з похмурого царства тіней.

Розгнівався Аїд, знову послав він Таната за душею Сізіфа. З'явився Танат у палац хитрішого із смертних і застав його за розкішним банкетом. Вивергнув душу Сізіфа ненависний богам і людям бог смерті; назавжди відлетіла тепер душу Сізіфа в царство тіней..

Тяжке покарання несе Сізіф у загробному житті за всі підступництва, за всі омани, які зробив він на землі. Він засуджений вкочувати на високу, круту гору величезний камінь. Напружуючи всі сили, трудиться Сізіф. Піт градом струменіє з нього від тяжкої роботи. Всі ближче вершина; ще зусилля, і скінчена буде праця Сізіфа; але вирівається з рук його камінь і з шумом котиться долілиць, піднімаючи хмари пилу. Знову приймається Сізіф за роботу.

Так вічно котить камінь Сізіф і ніколи не може досягти мети - вершини гори...»

Нічого не скажеш - сумна історія! Адже Сізіф – взагалі - непоганий хлопець, у всякому разі в розумі йому не відмовиш! Чи не можна допомогти бідоласі?

От що цікаво - дайте прочитати давній міф програмісту, і він скаже: «Нічого страшного. Мова йде про виконання нескінченного циклу. Я й сам іноді попадаю в таке ж положення й знаходжу з нього вихід!»

Цикл є однієї з найважливіших алгоритмічних структур і являє собою послідовність операторів, що повторюється неодноразово. Цикли дозволяють записати повторювані дії в компактній формі.

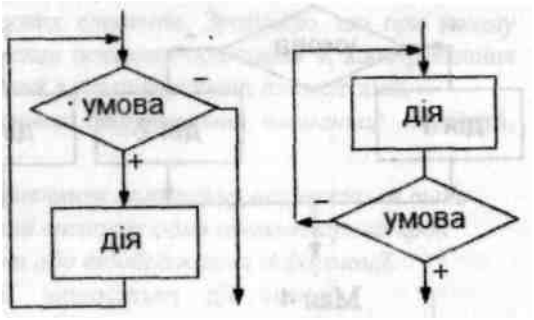
Якщо дії P1, ..., Pn треба повторяти, поки виконується деяка умова A, то використовується вказівка:

Поки A
Пц P1
.....
Pn
Кц

ЦИКЛ - це форма організації дій, при якій одна і та сама послідовність дій виконується кілька разів доти, поки виконується деяка умова.

Виконання команди приводить до того, що вказана в ній серія команд виконується декілька разів підряд. Вона виконується стільки разів, скільки потрібно для того, щоб умова перестала дотримуватися. Якщо умова не дотримується з самого початку, то серія не виконується жодного разу. Умова циклу перевіряється перед виконанням серії, але не в процесі її виконання

Оператори повторення призначені для багаторазового виконання оператора або групи операторів, їх ще



називають операторами організації циклу. Багаторазово повторювані частини обчислень називають тілом циклу, а змінні, що впливають на виконання циклу, — змінними циклу.

Алгоритм циклічної структури у загальному вигляді має містити:

5. підготовку циклу: задання початкових значень змінним циклу перед його виконанням;
6. тіло циклу: дії, повторювані в циклі для різних значень змінних циклу:
7. модифікацію (зміну) значень змінних циклу перед кожним новим його повторенням;
8. керування циклом: перевірку умови продовження (або закінчення) циклу й перехід на початок тіла циклу, якщо виконується умова продовження циклу (або вихід із циклу по його закінченні).

Для цього передбачено кілька видів команд організації циклів. Команда «**доки»** (цикл із попередньою перевіркою (передумовою)

Формат команди алгоритмічною мовою:

доки умова
пц
серія
кц

Робота команди **«Доки»**

Команда працює в такий спосіб: спочатку перевіряється умова, і якщо вона істинна, виконується зазначена послідовність команд, після чого знову перевіряється умова; якщо умова хибна, відбувається вихід з конструкції. Цикл може взагалі не виконуватися, якщо умова апріорі хибна.

Наведемо приклад:

i:=1; s:=0
доки i<10
пц s:=s+i
i:=i+1
кц

У результаті виконання цього прикладу в комірці *s* буде занесено результат обчислення: сума чисел від 1 до 9 включно. При *i* = 10 відбудеться вихід з циклу. Якщо до моменту початку роботи циклу значення *i* буде більше або дорівнюватиме 10, то цикл взагалі виконуватися не буде.

Команда «Повторювати» (цикл із постумовою (післяперевіркою). Формат команди алгоритмічною мовою:

Повторювати
пц
серія
кц
до умова

Робота команди **«Повторювати»**

Робота команди полягає в тому, що на початку виконується серія, і лише потім перевіряється умова. Якщо вона хибна, то знову виконується серія тощо. Робота циклу завершиться, якщо умова стане істинною. У будь-якому разі цикл спрацьовує хоча б один раз. Наприклад:

Формат команди алгоритмічною мовою:

```
i:=1;  s:=1
повторювати
пц
   s:=s+i
   i:=i+1 кц до  i=10
```

Цикл буде працювати доти, доки і не дорівнюватиме 10. Значення 8 при цьому дорівнюватиме 46

Команда повторення з параметром «Для».

Формат команди:

для і від j до k крок m
пц
серія
кц

Робота команди **«Для»**

де і — керуюча змінна циклу (параметр);
I — нижня межа зміни значення і;
k — верхня межа зміни значення і;
m — значення кроку. Якщо воно дорівнює 1, то його можна опускати;
***j,k,m** — У загальному випадку можуть бути виразами.*

Команда працює в такий спосіб:

5. Керуюча змінна *і (змінна циклу)* на початку набуває значення *j* і здійснюється перевірка на кінцеве значення k; якщо умова «істинна», то виконуються команди циклу.

6. Потім, щоразу до керуючої змінної і додається значення кроку m і здійснюється перевірка на кінцеве значення k; якщо умова «істинна», то виконуються команди циклу.

7. Виконання серії циклу відбувається тільки після виконання однієї з умов:
• i < k, якщо крок додатний (у цьому випадку *j* < k);
• *i* > k, якщо крок від'ємний (у цьому випадку j > k).

8. За невиконання умови відбувається вихід з циклу.

Наведемо фрагменти алгоритмів з використанням команди «**ДЛЯ».**

для і **від** 1 **до** 10 крок 1
пц
Друхувати і
кц

або

для і **від** 10 до 1 крок -1
пц
Друкувати і
кц

У результаті виконання фрагментів алгоритмів на дисплеї одержимо:
• у першому випадку 1 23456789 10;
•у другому 10 98765432 1.

Загалом, алгоритм роботи команди **для** однаковий для всіх алгоритмічних мов програмування.

Використовуючи цикл **для,** необхідно дотримуватися певних правил:

6. *Змінна, що керує циклом, не повинна змінюватися в циклі командою присвоювання (це може призвести до помилки або неочікуваного результату).*

7. *Не рекомендується виходити з циклу, не дочекавшись його завершення.*

8. *Не можна входити в цикл, оминаючи команду* **для**.

9. *У внутрішніх циклах межі зміни змінних краще обчислювати перед циклом, це заощаджує час роботи циклів.*

10.*Неприпустимо використовувати у вкладених циклах однакову змінну (це стосується й інших команд циклу).*

Команду повторення з параметром зручно використовувати, коли наперед відома кількість повторень циклу. Вона передбачає початкове присвоювання значення керуючої змінної, йоги модифікацію й перевірку умови на вхід-вихід із циклу.

Порівняно з командами **ДОКИ** або **ПОВТОРЮВАТИ,** її можливості обмежені, оскільки останні можуть мати кілька керуючих змінних циклу і кількість повторень заздалегідь може бути невідомою.

Цикл можна організувати й за допомогою команд, **ЯКЩО** й **ПЕРЕЙТИ** М, де М — мітка рядка, у яку здійснюється перехід, але з метою збереження структурності алгоритмічної мови такий варіант використовувати не рекомендується. Це вважається поганим стилем програмування.

Наведемо кілька прикладів циклічних алгоритмів.

1. Запишіть у виді команди повторення прислів'я:
а) Куй залізо, поки гаряче.
б)Скільки вовка не годуй, він у ліс дивиться.

2. Скласти алгоритм Евкліда знаходження найбільшого спільного дільника

Алгоритм знаходження найбільшого спільного дільника (НСД)

алг НСД (нат *M, N,* нат *НСД)*
арг *M, N*
рез ***НСД***
поч нат x, y x:= M; в:= N поки x≠y
пц якщо x > y то x: = x — y
інакше y: = y — x все
КЦ

НСД: = x **кін**

Задачі для самостійного розв'язання.

1. Складіть алгоритм, за допомогою якого визначте добуток:
А) перших 300 натуральних чисел;
Б) усіх двозначних парних чисел;
В) усіх двозначних непарних чисел;
Г) усіх тризначних чисел, кратних 7
2. Складіть алгоритм, за допомогою якого визначте суму:
а) усіх двозначних парних чисел;
б) усіх двозначних непарних чисел;
в) усіх тризначних непарних чисел.

Домашнє завдання:

Виконати обчислення з використанням команди повторення:
а) знайти суму перших 20 значень натурального ряду чисел;
б) знайти суму значень натурального ряду чисел від 15-го по 80-е;
в) знайти суму значень тих натуральних чисел, які кратні 7 і не перевищують 500.

Домашнє завдання:

вивчити означення, що прочитані на лекції;
1. Складіть алгоритм, за допомогою якого можна визначити суму:
А) перших 300 натуральних чисел;
Б) квадратів перших 500 натуральних чисел.
2. Складіть алгоритм, за допомогою якого можна визначити добуток:
А) всіх парних двозначних чисел;
Б) всіх чотиризначних чисел, кратних 5.

Етапи розв'язування задач

При проведенні різних експериментів, наукових і навчальних розрахунків, досліджень часто буває необхідно будувати якісь зменшені копії дійсних об'єктів, математичні графіки або використовувати інші засоби для наочного представлення поведінки різних об'єктів і процесів, тобто займатися моделюванням.

З давніх давен людина використовує моделювання для дослідження об'єктів, процесів та явищ в різних галузях своєї діяльності. Результати цих досліджень допомагають визначити та покращити характеристики реальних об'єктів та процесів, краще зрозуміти сутність явищ та пристосуватись до них або керувати ними, конструювати нові та модернізувати старі об'єкти. Моделювання допомагає людині приймати обґрунтовані рішення та передбачати наслідки своєї діяльності. Поняття комп'ютерного моделювання відображає використання в цьому процесі комп'ютера, як потужного сучасного засобу обробки інформації. Завдяки комп'ютеру суттєво розширюються галузі застосування моделювання, а також забезпечується всебічний аналіз отриманих результатів.

Що ж таке модель? Під цим словом ховаються і матеріальні моделі реально існуючих об'єктів на виставці, і телевізійна красуня, що рекламує товари та сучасний одяг, і макет Ейфелевої башти, і теорія розвитку суспільства, і всім відома формула земного тяжіння $P = mgh$, і багато чого іншого. Як же в одному слові можна об'єднати такі різні поняття?

Справа в тому, що поняття моделі об'єднує дещо спільне, а саме те, що ***модель*** - це штучно створений людиною абстрактний або матеріальний об'єкт. Аналіз та спостереження моделі дозволяє пізнати сутність реально існуючого складного об'єкта, процесу чи явища, що називаються прототипами об'єкта. Таким чином, ***модель*** - це спрощене уявлення про реальний об'єкт, процес чи явище, а ***моделювання*** - побудова моделей для дослідження та вивчення об'єктів, процесів та явищ.

Може виникнути питання, чому не можна дослідити сам оригінал, навіщо створювати моделі? Для цього може існувати багато причин:

- оригінал може не існувати в часі (гіпотеза про загиблий материк Атлантида, про побудову Єгипетських пірамід, про можливу "ядерну зиму", що може початися після атомного бомбардування);
- реально цей об'єкт не можна побачити (модель земної кулі, сонячної системи або атома);
- людина хоче побачити об'єкт, але не має можливості потрапити на місце його знаходження (модель Ейфелевої башти, єгипетської піраміди, Софіївського собору тощо);
- процес, який досліджує вчений, небезпечний для життя (ядерна реакція).

Таких прикладів можна придумати багато. І ви, якщо подумаєте, можете згадати багато моделей, що бачили у своєму житті.

Згадайте, які моделі ви бачили у своєму житті (сама тривіальна відповідь - іграшки). Навіть, коли ви зранку складаєте план своїх дій на день, це теж можна вважати моделюванням.

Для одного і того ж об'єкта можна створити велику кількість моделей. Все залежить, по-перше, від мети, що ви поставили перед собою, а по-друге, від методів та засобів, за допомогою яких ви збираєте інформацію про прототип. Наприклад, якщо ви хочете познайомитись з новим містом, то карта цього міста, фотографії, розповіді мешканців або кіноальманах дадуть вам зовсім різні уявлення про об'єкт, причому ці уявлення можуть зовсім не співпасти з тими враженнями, що ви отримаєте потім після відвідування цього міста безпосередньо. Модель цього ж самого міста для його мешканців взагалі буде іншою, тому що для них головне - це забезпечення нормальної життєдіяльності. Як ви вже переконались, кількість моделей та їх різноманітність дуже велика. Щоб не загубитися

в цьому розмаїтті, необхідно мати якусь класифікацію моделей. Розглянемо найбільш суттєві ознаки, за якими класифікуються моделі:

- галузі використання;
- урахування в моделі фактора часу;
- спосіб представлення моделей.

Розглядаючи моделі з позиції галузі використання, можна сказати, що вони бувають:

- *навчальні* - наочні посібники, тренажери, навчальні програми;
- *дослідні* - створюються для дослідження характеристик реального об'єкта (модель теплоходу перевіряється на усталеність, а модель літака - на аеродинамічні характеристики);
- *науково-технічні* - для дослідження процесів та явищ (ядерний реактор або синхрофазотрон);
- *ігрові* моделі - для вивчення можливої поведінки об'єкта в запрограмованих або непередбачених ситуаціях (військові, економічні, спортивні ігри тощо);
- *імітаційні* моделі - виконується імітація дійсної ситуації, що багато повторюється для вивчення реальних обставин (випробування лікарських препаратів на мишах або інших тваринах, політ собаки в космос).

З урахуванням фактора часу моделі можуть бути динамічні та статичні. В першому випадку над об'єктом виконуються дослідження на протязі деякого терміну, а в другому - робиться одноразовий зріз стану (наприклад, постійний нагляд сімейного лікаря та одноразове обстеження в поліклініці).

За способом представлення моделі можуть бути матеріальні та інформаційні. Матеріальні моделі - це предметне відображення об'єкта зі збереженням геометричних та фізичних властивостей. Наприклад, іграшки, чучела тварин, географічні карти, глобус і таке інше - це матеріальні моделі реально існуючих об'єктів. Матеріальною моделлю можна також назвати хімічний або фізичний дослід. Ці моделі реалізують матеріальний підхід до вивчення об'єкта чи явища.

Інформаційна модель - це сукупність інформації, що характеризує властивості та стан об'єкта, процесу чи явища, а також взаємодію із зовнішнім світом.

Інформаційні моделі можуть бути:

- *вербальними* - моделі отримані в результаті розумової діяльності людини і представлені в розумовій або словесній формі;
- *знаковими* - моделі, що виражені спеціальними знаками (малюнками, текстами, схемами, графіками, формулами тощо).

За формою представлення можна виділити наступні види інформаційних моделей:

- *геометричні* - графічні форми та об'ємні конструкції;
- *словесні* - усні та письмові описи з використанням ілюстрацій;
- *математичні* - математичні формули, що відображають зв'язок різних параметрів об'єкта;
- *структурні* - схеми, графіки, таблиці;
- *логічні* - моделі, в яких представлені різні варіанти вибору дій на основі різних умовиводів та аналізу умов;
- *спеціальні* - ноти, хімічні формули і таке інше;
- *комп'ютерні* та *некомп'ютерні*.

В сучасному світі розв'язування складних наукових та виробничих задач неможливе без використання моделей та моделювання. Серед різних видів моделей особливе місце займають математичні моделі, тому що вони дозволяють враховувати кількісні та просторові параметри явищ та використовувати точні математичні методи. Вивчення реальних явищ за допомогою математичних моделей, як правило, вимагає застосування обчислювальних методів. При цьому широко використовуються методи прикладної математики, математичної статистики та інформатики.

Підсумуємо вищевикладене.

Моделювання - це особлива форма експерименту, яка полягає в тому, що досліджується не сам об'єкт, а деяка його заміна.

Форми моделювання дуже різноманітні і залежать як від самого об'єкта, так і від мети його вивчення.

Інформаційна модель - це матеріальний або ідеальний об'єкт, що використовується замість оригіналу явища (процесу) при його дослідженні, при цьому зберігається інформація про деякі важливі для даного дослідження типові риси і властивості оригіналу, тобто його істотні сторони.

Модель необхідна для того, щоб:

- зрозуміти, як влаштований об'єкт: які його структура, основні властивості, закони розвитку і взаємодії з навколишнім світом;
- навчитися керувати об'єктом або процесом і визначати найкращі способи керування при заданій меті і критеріях (оптимізація);
- прогнозувати прямі чи непрямі наслідки впливу на об'єкт.

Інформаційні моделі можуть формулюватися на будь-яких мовах: російській, англійській й ін. В них можуть бути використані мова графічних побудов, мова хімії, біології і т.д.

Окремим типом інформаційних моделей є математична модель.

Математична модель - це заміна оригіналу явища (процесу) відповідним аналогом за допомогою математичних залежностей.

Рішення практичної задачі починається з опису вихідних даних і мети задачі. Точне формулювання умов і мети рішення - це математична постановка задачі, а математичний опис найбільш істотних властивостей реального об'єкта - це математична модель.

Комп'ютерна модель - математична модель, яка реалізована за допомогою певних програмно-апаратних засобів.

Існують задачі, які важко або неможливо розв'язати без застосування комп'ютерів. Це різного роду задачі моделювання, наприклад, моделювання фізичних процесів, біологічні, екологічні, економічні задачі моделювання, задачі оптимізації й інші.

Етапи розв'язання прикладних задач з використанням комп'ютерів

Етап	Опис етапу
Математична постановка задачі	1. Визначити умови задачі: - Що дано? - Які дані допустимі? - Які результати, в якому вигляді повинні бути отримані?
Побудова математичної моделі, вибір методу рішення	1. Розгорнутий змістовний опис задачі замінити її математичною моделлю за допомогою математичних залежностей. 2. Обгрунтовано вибрати метод розв'язування.
Складання алгоритму на основі обраного методу	Алгоритм у більшій мірі визначається обраним методом, хоча той самий метод може бути реалізований за допомогою різних алгоритмів. При складанні алгоритму слід враховувати всі його властивості.
Складання програми	Програмування (складання програми) - кодування алгоритму на одній з мов програмування.
Тестування і налагодження програми	Перевірка правильності роботи програми за допомогою тестів і виправлення виявлених помилок. Тест - це спеціально підібрані вихідні дані і результати, отримані при цих даних.
Аналіз результатів	Після остаточного виконання програми зробити аналіз результатів. Можлива зміна самого підходу до рішення задачі і повернення до першого етапу для повторного виконання всіх етапів.

Математична модель - це перелік вхідних даних, результатів, які потрібно отримати, та зв'язок між величинами, виражений у вигляді математичних співвідношень.

Для побудови математичної моделі використовується така форма:

Дано: <Перелік початкових даних>

Потрібно: <Перелік погрібних результатів>

Зв'язок: <Система рівнянь або тверджень, що зв'язують вхідні та шукані дані>

При <Умови допустимості початкових даних>

Приклади математичної моделі

Задача: Розв'язати лінійне рівняння типу ax+b=c

Дано: a, b, c Потрібно: x

Зв'язок: якщо a=0 і c – b = 0, то x - будь-яке число;

якщо a=0 і c – b $\neq$0, то коренів немає;

якщо a $\neq$ 0 і c – b = 0, то x =0;

якщо a $\neq$ 0 і c – b $\neq$ 0, то x=(c – b)/a.

III. Формування практичних навичок.

Побудувати математичну модель та алгоритм розв'язування задач:

1. У цариці Несміяни кругле обличчя, радіус якого дорівнює R. Визначте довжину сторони такого квадратного дзеркала, щоб, коли Несміяна милується собою, її відображення поміщалося в дзеркалі.

2. З тераріуму втекли x гадюк, у кобр та z гюрз. Довжина кожної гадюки 1м, кожної кобри 1м 30см, а гюрзи 1м 15см. Скільки повних метрів отруйних змій втекло з тераріуму? Яку довжину вони складають у сантиметрах?

Домашнє завдання:

- вивчити означення, що прочитані на лекції (що таке модель та моделювання, класифікація моделей, комп'ютерне моделювання);
- придумати приклади моделей, що зустрічаються нам у повсякденному житті;
- придумати набір елементів конструктора, з якого потім можна зібрати яку-небудь іграшку;
- Побудувати математичну модель та алгоритм розв'язування задач:

а) знайти корені квадратного рівняння типу ax^2+Ь=0;

б) знайти корені квадратного рівняння типу ax^2+bx+c=0;

в) прямокутник, довжини сторін якого відповідають умові a/b=b/(a-b), називається золотим. Визначити, чи є даний прямокутник золотим.

Основні поняття мови Паскаль
Мова програмування Turbo Pascal 7.0

3. Мови програмування. Паскаль.

Першими мовами програмування були FORTRAN, COBOL, ALGOL і деякі інші. У кожної з них були свої позитиви й свої недоліки. Одною з найбільш удалих довгий час уважалася ALGOL, настільки вдалою, що цю мову стали використовувати в спеціальній літературі для запису алгоритмів. Але й вона не була позбавлена недоліків, зокрема, остання версія ALGOL'a була зайво громіздкою, тому швейцарський професор наприкінці 60-х років XX ст. Никлаус Вірт зайнявся розробкою своєї власної мови, що успадкувала би від ALGOL'a краще, але була би більше лаконічною і мала би більше чітку логічну структуру. Призначалася нова мова для навчання студентів, і сам Вірт спочатку ставився до неї як до іграшки. Мова була названа на честь французького філософа й винахідника механічного калькулятора Блеза Паскаля - Паскалем. Нова мова виявилася настільки вдалою, що швидко завоювала популярність.

Поява нової мови збіглася з початком ери персональних комп'ютерів, які стали доступними майже кожній звичайній людині (це відбулося не відразу, і були часи, коли вартість «персоналки» була порівняння з вартістю легкового автомобіля!). Масла у вогонь підлила фірма Borland, яка у першій половині 80-х випустила пакет Турбо Паскаль, що містив не тільки транслятор, але й редактор, а також інші програми, які значно полегшували процес програмування. Це зробило Турбо Паскаль популярнішою системою програмування.

Турбо Паскаль. Чому не просто Паскаль, а *Тур*бо Паскаль? Слово «Турбо» в англійському лексиконі означає прискорення. Транслятор, що входить до складу Турбо Паскаля, дуже швидко переводить програму з мови програмування в машинні коди, помітно швидше, ніж транслятори в інших системах програмування. От тому - Турбо.

Скажемо ще, що Турбо Паскаль - це не окрема мова програмування, а «розширення» звичайного, стандартного Паскаля, що включає інтегроване середовище програмування. Слова «інтегроване середовище» означають, що з однієї програми є доступ до редактора текстів, транслятора, довідкової системи, налагоджувача та ін. До складу Турбо Паскаля входять додаткові набори процедур, які дозволяють не займатися щораз программуванням деяких складних дій, таких, наприклад, як виведення графіки. Крім Турбо Паскаля є й інші системи програмування на Паскалі, але ми будемо використовувати саме Турбо Паскаль.

4. Поняття про мови прогамування.

Процес роботи комп'ютера полягає у виконанні програм, тобто деякого набору команд, що надходять у визначеному порядку. Машинний код команди складається з нулів та одиниць та указує, яку саме дію треба виконати центральному процесору. Отже, щоб задати комп'ютеру послідовність дій, яку він має виконати, треба задати послідовність двійкових кодів відповідних команд. Писати такі програми дуже складна справа. Раніше для цього програміст повинен був пам'ятати не тільки всі комбінації нулів та одиниць двійкового коду кожної команди, але й двійкові коди адрес даних, що використовувалися під час виконання програми. Щоб полегшити роботу програмістів, було розроблено багато мов програмування, які в більш наочному для людини вигляді подавали послідовність дій комп'ютера.

Алгоритмічні мови, призначені для побудови описів алгоритмів, що виконуються електронними обчислювальними машинами, називаються ***мовами програмування***.

Описи алгоритмів мовою програмування називають ***програмами***.

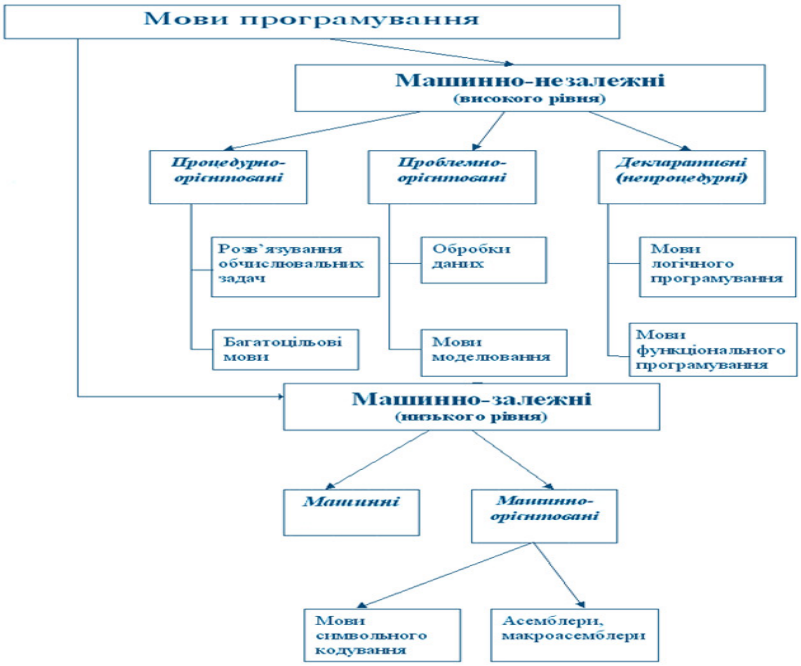
Набагато легше написати програму мовою, що наближена до людської, а перекладання цієї програми на машинні коди доручити комп'ютеру. Так з'явилися мови, що призначені спеціально для написання програм - мови програмування.

Існує багато різних мов програмування. Взагалі, для розв'язування більшості задач можна використовувати будь-яку з них. Тільки досвідчені програмісти знають, яку мову програмування краще використовувати для розв'язування складних спеціалізованих задач, щоб урахувати особливості тієї чи іншої з них.

Всі існуючи мови програмування можна поділити на дві групи:

- мови низького рівня;
- мови високого рівня.

До мов низького рівня належать мови ***асемблера*** (від англ. to assemble - складати, компонувати). У мові асемблера використовуються символьні позначення команд, які легко зрозуміти і запам'ятати. Замість послідовностей двійкових кодів команд записуються їх символьні позначення, а замість двійкових адрес даних, які використовуються під час виконання програми, - символьні імена цих даних. Іноді мову асемблера називають ***мнемокодом або автокодом***.



Більшість програмістів при складанні програм користуються деякою мовою високого рівня. Для описування алгоритмів такою мовою використовується певний набір символів - алфавіт мови. З цих символів складаються так звані ***службові слова*** мови, кожне з яких має певне призначення. Службові слова зв'язуються одне з одним в речення за певними синтаксичними правилами мови і визначають деяку послідовність дій, які мусить виконати комп'ютер.Використання мов високого рівня надає можливість описувати програми для комп'ютера, використовуючи загальноприйняті позначення операцій і функцій.

Та програми, що написані на мовах програмування високого рівня (алгоритмічних мовах програмування), комп'ютер "не розуміє". Для того, щоб він міг виконати програму, її потрібно перекласти на машинну мову. Для такого перекладу використовують спеціальні програми, що мають назву - транслятори.

Транслятор - це програма, що призначена для перекладу тексту програми з однієї мови програмування на іншу. Процес перекладання називається ***трансляцією***.

Будь-який транслятор виконує дві основні задачі.

Перша - аналіз програми, що транслюється, в результаті чого визначається її коректність. При виявленні помилок транслятор вказує на ті місця тексту програми, де порушені правила її написання.

Друга - генерація вихідної програми мовою команд комп'ютера.

Розрізняють два типи трансляторів:

- компілятори
- інтерпретатори.

Компілятор - це програма, призначена для перекладу в машинні коди програми, що написана мовою високого рівня. Процес такого перекладу називається компіляцією.Кінцевим результатом роботи компілятора є програма в машинних кодах, яка потім виконується ПК. Скомпільований варіант програми можна зберігати на диску. Для повторного виконання програми компілятор вже не потрібен. Досить завантажити з диска в пам'ять комп'ютера скомпільований перед цим варіант і виконати його.

Існує інший спосіб поєднання процесів трансляції та виконання програм. Він називається **інтерпретацією**.

Інтерпретатор - це програма, що призначена для трансляції та виконання вихідної програми по командах (на відміну від транслятора, який цей процес виконує в цілому).

У процесі трансляції відбувається перевірка програми на відповідність до правил її написання. Якщо в програмі знайдені помилки, транслятор виводить повідомлення про них на екран монітора. Інтерпретатор повідомляє про знайдені помилки після трансляції кожної команди програми, а компілятор - після завершення компіляції всієї програми. Знайти та виправити в цьому випадку помилки значно складніше, ніж при інтерпретації. Через це програми-інтерпретатори розраховані, в основному, на мови, що призначені для навчання програмуванню, і використовуються програмістами-початківцями.

Як правило, програми компілятори та інтерпретатори називаються так само, як і мови, для перекладу з яких вони призначені. Слова Паскаль, Бейсік, Сі можна сприймати і як назви мов, і як назви відповідних програм - трансляторів.

3. Опрацювання середовища ТП 7.0

3. Увійдіть в каталог системи програмування ТП 7.0.
4. Виконайте команду turbo.exe.

F10 - вихід в головне меню.

Esc - вийти з головного меню та повернутися у вікно редактора текстів.

Дія	**Доступ через меню**	Гаряча клавіша
Створити новий файл	File/New	
Відкрити файл	File/Open	F3
Зберегти файл	File/Save	F2
Виконати програму	Run/Run	Ctrl+F9
Переглянути екран	Debug/User	Alt+F3
Створити exe-файл	Compile/Compile	Alt+F9
Викликати довідкову	Help	F1
Отримати довідку про вказаний курсором		Ctrl+F1
Перейти до іншого вікна	Window/List	Alt +0
Закрити поточне вікно	Window/Close	Alt+F3
Вийти з середовища	File/Exit	Alt+X

Практичне завдання

Відкрийте існуючий в каталозі системи програмування файл Primerl.pas. В програмі Primerl використані основні елементи процесу опрацювання даних у відповідності з принципом IPO (input - processing - output), тобто введення - опрацювання - виведення.

Для повного розуміння цього принципу необхідно ознайомитися з операторами введення-виведення та присвоювання.

```
Program Primer1; {додавання двох цілих чисел}
var a,b,s: integer; begin
write (' введіть перше число'); readln(a);
write ( введіть друге число'); readln(b);
s=a+b;
writeln ( 'сума  A+B=' ,s)
end.
```

4. Виконайте програму декілька разів для різних A і B.
5. Створіть exe-файл. Для цього виконайте послідовність дій:
   - увійдіть в головне меню, виберіть меню Compile.
   - виберіть і виконайте команду Destination - Memory, яка після натискання клавіші Enter змінюється на Destination - Disk.
   - Натисніть комбінацію клавіш Alt+F9.
3. Вийдіть з середовища ТП 7.0. Виконайте файл Primerl.exe.

Домашнє завдання:

вивчити означення, що прочитані на лекції (що таке програма, класифікація мов програмування, що таке транслятор, типи трансляторів).

Основні поняття мови Turbo Pascal 7.0

Запитання для перевірки знань.

6. Для чого створено мови програмування?
7. Що можна назвати програмою?
8. Для чого програми перекладають в машинні коди?
9. Чим компілятор відрізняється від інтерпретатора?
10. Які мови програмування ви знаєте? Наведіть приклади.

Мова програмування — це один із способів подачі опису алгоритму, що розрахований на виконавця — комп'ютер. Будь-яка мова програмування характеризується трьома основними складовими: алфавітом, синтаксисом і семантикою.

Сукупність символів, які дозволяється використовувати під час побудови опису алгоритмів мовою програмування, називають **алфавітом** цієї мови.

Сукупність правил опису алгоритмів мовою програмування називають **синтаксисом** мови програмування.

Правила **семантики** пояснюють, яке смислове значення має кожний елемент мови. У будь-якій мові програмування можна виділити чотири типи елементів, що використовуються при побудові програм: символи, слова, вирази, команди (оператори). **Символи мови** — це основні знаки, якими описуються команди і дані.

Слова мови — структури, що утворені із символів алфавіту мови програмування і мають певний зміст. Слова — це імена змінних та констант, числа, службові слова та ін.

Вираз — це правило (формула) обчислення значення однієї величини. Якщо одержуване значення числове, то вираз називають арифметичним, якщо значення логічне, то вираз називають логічним або бульовим, якщо значення—текст, то вираз називають літерним.

Команда — це вказівка на виконання деякої дії. При написанні програм команди називають операторами, а величини, що використані команді — операндами.

Скінченна послідовність виконуваних почергово команд називається **серією команд**. Серія може складатися з однієї команди і навіть бути порожньою.

Алфавіт і словник мови

Програма мовою Паскаль формується за допомогою набору знаків, що утворюють алфавіт мови, і складається з літер, десяткових і шістнадцяткових цифр і спеціальних символів.

У якості літер використовують великі та малі літери латинського алфавіту: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxy z, а також _ (знак підкреслення).

У якості десяткових цифр: 1234567890. Шістнадцяткові цифри утворюються з десяткових цифр і літер від A до F (або від a до f).

При написанні програм застосовуються спеціальні символи:

+ плюс	- мінус
* зірочка (знак множення)	/ знак ділення
= дорівнює	> більше
< менше	# міжнародний комерційний номер
$ знак грошової одиниці	[] квадратні дужки
() круглі дужки	{} фігурні дужки
. крапка	, кома
: двокрапка	; крапка з комою
' апостроф	~ тильда
@ ет (або комерційне а)	пробіл.

Комбінації спеціальних символів можуть утворювати складені символи, які сприймаються комп'ютером як один символ:

:= присвоювання	<> не дорівнює
… діапазон значень	>= більше або дорівнює
(..) альтернатива [ ]	<= менше або дорівнює.
(* *) альтернатива {}	

Неподільні послідовності символів утворюють слова, що несуть певний зміст у програмі. Слова відділяються розділовими символами, такими як: пробіл, кома, символ кінця рядка, коментар. Слова поділяються на: стандартні, зарезервовані, ідентифікатори користувача. *Зарезервовані* слова є складовою частиною мови, мають фіксоване написання і назавжди визначений зміст. Наприклад: ***begin, else, function, write, end, program*** та ін.

Стандартні слова призначені для заздалегідь визначених розробником мови типів даних, констант, процедур і функцій (наприклад, ***sin, cos, Pi***). Зарезервований ідентифікатор можна перевизначити, але це може призвести до помилки, тому краще цього не робити.

Ідентифікатори користувача використовують для позначення констант, змінних, процедур і функцій, що визначені самим програмістом. Існують загальні правила написання ідентифікаторів:

6. Ідентифікатор починається тільки з літери або знака підкреслення.
7. Ідентифікатор може складатися з літер, цифр і знака підкреслення.
8. Між двома ідентифікаторами має бути хоча б один розділовий знак.
9. Максимальна довжина ідентифікатора 127 символів, але значущими є тільки перші 63 символи.
10. Ідентифікатор не може бути зарезервованим словом.

У написанні програм можна використовувати як великі, так і малі літери. Компілятор не визначає різниці між ними.

<u>Правила оформлення програм (пунктуації):</u>

6. Крапку з комою можна не ставити після begin і перед end, тому що ці слова є операторними дужками, а не операторами.
7. Крапка з комою розділяє оператори. її відсутність між операторами викликає помилку компіляції. Наявність між операторами кількох крапок з комою не є помилкою, тому що компілятор сприймає їх як ознаку наявності порожніх операторів.
8. При використанні вкладених структур може виникнути ситуація:

```
                    end;
            end;
    end
```

Крапку з комою можна ставити як після кожного, так і після останнього end. А наприкінці програми можна ставити крапку.

9. В операторах циклу крапка з комою не ставиться після while, repeat, do і перед until.
10. В умовних операторах крапка з комою не ставять після then і перед else.

Опитування за питаннями для перевірки засвоєння знань

6. Що називають алфавітом мови програмування?
7. Що називають серією команд?

8. Чи можна змінити зміст зарезервованого слова?
9. Для чого використовують ідентифікатори користувача? .
10. Чи може бути ідентифікатор користувача зарезервованим словом?

Домашнє завдання:
вивчити означення, що прочитані на лекції.

Величини. Прості типи мови Паскаль.

1. Поняття величини.

У своїй роботі програміст має справу з таким поняттям, як величина.

Що ж таке величина? З точки зору програмування *величини* — це дані, що обробляються програмами. Мова Паскаль інтерпретує дані, як *константи* або *змінні.* Як перші, так і другі визначаються ідентифікаторами (іменами), за допомогою яких можна звертатися для одержання відповідних значень. *Константами* називають такі дані, яким присвоюються значення в описовій частині програми, й у процесі виконання програми їх змінювати заборонено. Для визначення констант служить зарезервоване слово *const.*

Формат опису:

Const < ідентифікатор > = < значення константи >;

Приклад: **Const Max=1000;**

Vxod='сегмент 5';

Є константи, до значень яких можна звертатися без попереднього опису.

Ідентифікатор	Тип	Значення	Опис
True	Boolean	True	Істина
False	Boolean	False	Хибність
Maxint	integer	32767	Максимальне

Змінні, на відміну від констант, можуть змінювати свої значення в процесі виконання програми. Кожна змінна і константа належать до визначеного типу даних. Тип констант визначається компілятором автоматично. Тип змінних обов'язково вказується перед тим, як їх використати. Для опису змінних призначено зарезервоване слово *var.*

Формат опису: Var <ідентифікатор> : <тип даних>;

Приклад: Var Suml, Sum2 : real;

2. Типи даних.

Тип даних — це сукупність властивостей певного набору даних, від яких залежать: діапазон значень, якого можуть набувати ці дані, а також сукупність операцій, які можна виконувати над цими даними.

Усі типи даних у мові програмування Паскаль розділяють на дві групи: скалярні (прості), структуровані (складені). *Скалярні* типи, у свою чергу розділяють на *стандартні* та *типи користувача.* Стандартні типи користувачам пропонують розробники системи Turbo Pascal. Типи користувача — розроблюють програмісти самостійно.

До стандартних скалярних типів належать типи: цілі, дійсні, літерні, булівські. Величини *цілих* типів можна подавати як у десятковій, так і в шістнадцятковій системах числення. Якщо число подане в шістнадцят-ковій системі, перед ним без пробілу записується знак *$.* Діапазон зміни шістнадцяткових чисел від $0000 до *SFFFF.*

Цілі типи даних являють собою значення, що можна використовувати в арифметичних виразах.

Стандартні цілі типи наведено в таблиці:

Тип	Діапазон	Необхідна пам'ять (байт)
Byte	0...255	1
Shortint	-128... 127	1
Integer	-32768 ...32767	2
Word	0... 65535	2
Longint	-2147483648 ...2147483647	4

Дійсні типи даних являють собою дійсні значення, що використовуються в арифметичних виразах і займають у пам'яті від 4 до 10 байт. У програмі мовою Паскаль можна подавати дані дійсних типів у вигляді як із плаваючою, так і з фіксованою крапкою.

Дійсні *десяткові числа з фіксованою крапкою* записуються за звичайними правилами арифметики. Зверніть увагу, що ціла частина від дробової відокремлюється десятковою крапкою, а не комою. Якщо десяткова крапка відсутня, число вважається цілим. Перед числом може знаходитися знак «+» або «-». Якщо знак відсутній, то число вважається додатнім.

Дійсні *десяткові числа у форматі з плаваючою крапкою* подаються в експоненціальному вигляді: *mE+p,* де *m*- мантиса (ціле або дробове число з фіксованою десятковою крапкою), *E* - означає «десять у степені», *p* - порядок (ціле число). Мантиса має бути нормалізованою, тобто поданою у вигляді числа з діапазону від 0 до 9 (це означає, що крапка завжди знаходиться після першої значущої цифри числа). Однак можна записати мантису у вигляді будь-якого дробового числа з фіксованою крапкою. Нормалізація при цьому виконується системою автоматично. Наприклад:

Число у форматі з	Значення числа
0.4500E+02	$0.45*10^{2}=45$
-2.600E05	$-2.6*10^{5} = -260000$
+0.45670E-02	$0.4567*10^{-2}= 0.004567$

Стандартний (найчастіше використовуваний) дійсний тип даних наведений у таблиці:

Тип	Діапазон значень	Мантиса (кількість	Необхідна пам'ять
Real	±2.9*10E-39 ..	11-12	**6**

Літерний (символьний) тип може набувати значень ASCII-таблиці комп'ютера. Символьній змінній в пам'яті виділяється один байт, тому можна зберегти тільки один символ ASCII-таблиці.

Булівський тип подається двома значеннями: ***True*** *(істина)* або ***False*** *(хибність).* Цей тип застосовують у логічних виразах і виразах відношення.

Структуровані типи у своїй основі мають один або кілька скалярних типів даних. До структурованих типів даних належать рядки, масиви, файли, записи і т.д. їх ми розглянемо пізніше.

Перетворення типів:

Ціле значення можна перетворити в дійсне, присвоївши дійсній змінній ціле значення: A:=3+5.

Дійсне значення можна переворити в ціле за допомогою стандартних функцій: TRUNC(X) – ціла частина аргументу; ROUND(X) – аргумент, округлений до найближчого цілого (за правилами математики); INT(X) – аргумент, округлений до найближчого меншого цілого.

Сумісність типів:

При виконанні оператора присвоювання обчислюється вираз, що стоїть в правій частині, і його значення присвоюється змінній в лівій частині. При цьому тип виразу повинен відповідати типу змінної. Для стандартних типів це означає, що вони повинні збігатися. Допускається присвоювання змінній дійсного типу значення цілого типу. Присвоювання ж змінній цілого типу виразу дійсного типу заборонено!

Змінні і константи всіх типів використовуються у **виразах**

Вираз задає порядок виконання дій над даними і складається з операндов (констант, змінних, звертань до функцій), круглих дужок і знаків операцій. Круглі дужки ставлять, як і в математиці, для керування порядком виконання операцій. Якщо дужки відсутні, то операції виконуються залежно від їх пріоритетів, про що буде сказано далі.

У мові Паскаль є такі операції: арифметичні; відношення (порівняння); логічні. Операції можуть бути *унарними* та *бінарними.* У першому випадку операція стосується одного операнду і завжди записується перед ним, у другому операція виражає відношення між двома операндами і записується між ними. ***Арифметичні операції*** задають, арифметичні дії у виразах над значеннями операндів цілих та дійсних типів. Найчастіше використовуються арифметичні операції, що подані в наступній таблиці:

Операція	Дія	Тип операндів	Тип результату
Бінарні			
+	Додавання	Цілий	Цілий
		Дійсний	Дійсний
	Віднімання	Цілий	Цілий
		Дійсний	Дійсний
*	Множення	Цілий	Цілий
		Дійсний	Дійсний
/	Ділення	Цілий	Дійсний
		Дійсний	Дійсний
Div	Ділення націло	Цілий	Цілий
Mod	Залишок від ділення	Цілий	Цілий
Унарні			
+	Збереження знака	Цілий	Цілий
		Дійсний	Дійсний
-	Інверсія знака	Цілий	Цілий
		Дійсний	Дійсний

Операції відношення виконують порівняння двох операндів і визначають значення виразу є істинним або хибним. Результат завжди має булівський тип одного з двох значень: ***True*** *(істина)* або ***False*** *(хибність).*

Операція	Назва	Вираз	Результат
=	Дорівнює	A=B	True, якщо A дорівнює B
<>	Не дорівнює	A<>B	True, якщо A не дорівнює B
>	Більше	A>B	True, якщо A більше B
<	Менше	A<B	True, якщо A менше B
>=	Більше або дорівнює	A>=B	True, якщо A більше або дорівнює B
<=	Менше або дорівнює	A<=B	True, якщо A менше або дорівнює B

Результатом виконання *логічного (булівського)* ***виразу*** є логічне значення ***True*** або *False.*

Виконання кожної операції відбувається з урахуванням її пріоритету. Пріоритети операцій зазначені в наступній таблиці:

Операція	Приоритет	Вид операції
Not, унарні «-»і «+»	перший (вищий)	Унарна операція
*, /, div, mod, and	другий	Операції типу множення
+, -, or	третій	Операції типу додавання
=, <>, <, >, <=, >=	четвертий (нижчий)	Операції відношення

Арифметичні вирази у якості операндів можуть містити імена функцій. Про поняття *функції* мова буде йти пізніше в курсі програмування, але стандартні функції *(cos, sin,* x^2 та інші) знайомі учням зі шкільного курсу математики, і використання таких виразів у курсі інформатики відрізняється тільки правилами запису (синтаксису). Так, наприклад, у програмуванні аргумент функції обов'язково береться в круглі дужки.

3. Команда присвоювання

Команда присвоювання має вигляд

<ім'я змінної> := <вираз>;

Дія команди. Обчислюється вираз і його значення надається змінній. Вираз призначений для описування формул, за якими виконуються обчислення. Вираз може містити числа, змінні, сталі, назви функцій,з'єднані символами операцій.

Змінна і вираз мають бути одного типу або узгодженими: змінним дійсного типу можна надавати значення виразів цілого типу, а змінним рядкового типу присвоювати значення виразів символьного типу, але не навпаки.

Приклад. Розглянемо дію команд присвоєння в програмі Trykutnyk: p:=a+b+c; p:=p/2; s:=sqrt(p*(p-a)*(p-b)*(p-c)). Тут обчислюється значення периметра і воно надається змінній p, півпериметра (надається також змінній p) та площі (надається змінній s).

Основні стандартні функції та процедури

Функція	Тип аргументу	Тип результату	Математичний запис, коментар
abs(x)	integer, real	integer, real	$\|x\|$
arctan(x)	integer, real	real	arctgx
cos(x)	integer, real	real	cosx
sin(x)	integer, real	real	sinx
exp(x)	integer, real	real	ex
ln(x)	integer, real	real	lnx
sqrt(x)	integer, real	real	$\sqrt{x}$
sqr(x)	integer, real	integer, real	X2
ord(x)	char упорядкований	integer	ASCII-код симв., номер елемента
succ(x)	упорядкований	упорядкований	повертає наступне значення x
pred(x)	упорядкований	упорядкований	Повертає попереднє значення x
round(x)	real	integer	заокруглює число x до цілого
trunc(x)	real	integer	відкидає дробову частину числа x
int(x)	real	real	відкидає дробову частину числа x
frac(x)	real	real	дробова частина чис-
odd(x)	integer	boolean	true (x - непарне), false (x - парне)
random (x)	integer	integer	Генерує випадкове число з діапазону від 0

upcase(x)	char	char	замінює малу літеру латинської абетки на велику
процедури:			
inc(x.y)	integer	integer	збільшує x на y
inc(x)	integer, char	integer, char	Збільшує x на 1
dec(x.y)	integer	integer	Зменшує x на y
dec(x)	integer, char	integer, char	Зменшує x на 1

Розглянемо приклади значень функцій і виконання процедур:

round(2.1)=2, int(2.1)=2.0, x:=l; inc(x,5); (x=6),
round(6.8)=7, int(6.8)=6.0, x:=V; inc(x); (x='b'),
trunc(2.1)=2, frac(2.1)=0.1, x:=7; dec(x,3); (x=4),
trunc(6.8)=6, frac(6.8)=0.8, x:='d'; dec(x); (x=V).

$$x^{\alpha} = e^{\alpha \ln x}, \quad \log_b a = \frac{\ln a}{\ln b}$$

Це відповідає такому запису в Pascal:

exp (ln (x)* α).

Правила лінійного запису арифметичних виразів:

5. вираз повинен бути записаний у вигляді лінійного ланцюжка символів, що обумовлено конструкцією пристрою введення інформації;
6. не можна опускати знак операції множення;
7. порядок виконання операцій одного пріоритету регулюється дужками;
8. аргументи функцій записуються в круглих дужках.

Перестановка змінних. Добре іллюструє роботу із змінними наступне завдання - поміняти місцями значення двох змінних a й b (тобто, щоб в a з'явилося те, що було раніше в b, і навпаки). Здавалося б, можна записати так:

a := b; b := a;

однак тут криється помилка. Давайте перевіримо це на конкретному прикладі. Нехай a дорівнювало 10, а значення b дорівнювало 7. Після виконання оператора a := b в b залишиться 7; a дорівнюватиме 7, і тепер, якщо виконати оператор присвоювання b := a, там також виявиться 7. Ми втратили одне зі значень! Як же бути? Вихід полягає у використанні додаткової проміжної змінної t, в якій ми збережемо значення змінної a до того, як його буде треба замінити новим:

t := a; a := b; b := t;

Корисно перевірити виконання даного фрагменте програми на конкретному прикладі:

	a	b	T
Значення до	10	7	-
Після t:=a	10	7	10
Після a:=b	7	7	10
Після b:=t	7	10	10

Виконайте завдання

1. Скільки арифметичних операцій містить вираз:
   1)(x + 1/2)*(y + 7/10) -3/4;
   6) *(x + y)/(x - y)/x*y;*
   7) *(x + y)* mod *(x - y)* div 10;
   8) sqr *(x* mod *y* div *y)/x - y;*
   9) trunc (sin *((x - y)/x*y)* div 10?
2. Записати арифметичний вираз у вигляді звичайної алгебраїчної формули:
   6) *a*b*c - a/(n*m)/(a* + k/2);
   7) *a*b/(c + d) + (c- d)/b*(a* + 6);
   8) (a/(j + k/(2*x)) + c)/(j - sqr (sqrt (a)));
   9) x/(j + sqr (x)/(2 + x*sqr (x)/3));
   10)(x*sin *(y* + z*(2 + k)) + 1)/(0.7 - x/(y* + sqr (z)))*x/(a + *b).*
3. Вказати порядок виконання операцій під час обчислення значення простого арифметичного виразу:

   l)sqr(x) + sqr(sqr(i/));

6) x*sqr (y)/sqr (2);
7) *a*b/c*d;*
8) sqrt (a) + 1 - sqr *(b);*
9) -x*sqr (y)/sqr (sqr (2)).

4. Нехай *A*= 5, *B* = 4, C = 3 та P = 0.5. Обчислити значення простого арифметичного виразу:

1) *(A + B)/C*P;* 2) *(A + B)/C/P;*
3) *(A + B)/(C*P);* 4) -*A*B* + sqr (C) + 0.5.

5. Перевести формулу у вигляд , доступний для програмування:

1) $a+bc$; 2) $x+\frac{2x-10}{y+2}$; 3) $10^5\alpha(\beta-10^{-3})$;

4) $\frac{x-x_0}{x_1-x_0}-\frac{y-y_0}{y_1-y_0}$; 5) $\frac{a}{x+\frac{b}{2+\frac{c}{5+d}}}$.

6. Нехай задані такі значення дійсних змінних x = 0.5 та *h* = 0.1. Які значення матимуть ці змінні після виконання таких операторів присвоювання:

1) x:=2.5; 2) x := x + 2*h; *h:=h/2;*
3) x:=x + *h;* 4) *h* := - *h; x* := x + *h; h* := *h*21*

7. Є три цілочислові змінні з поточними значеннями A = 3, *B* = 5, *C* = 7. Які значення матимуть ці змінні в результаті виконання таких послідовностей операторів присвоювання:

4) *C* :=*A*B* + 2; *B* := *B* + 1; *A* := *C* - sqr (B);
5) P := C; C := B; B := *A; A* := P;
6) *B:=B+A; C:=C + B;*
4) A :=*A* + *1;B:=A* + *B;C:=A* + *B?*

8. Задані дійсні змінні x та *y* з поточними значеннями x = 0.8, *y* = - 2.1 та цілочислові змінні *k* та *j* із поточними значеннями *k* = 5, *j* = - 3. Знайти значення, якого набуде відповідна змінна в результаті виконання кожного з наведених нижче операторів присвоювання із зазначенням його типу:

1) k:=x; 2) k:=(k + 1)/10;
3) j:= 5 +y; 4) x :=(k + 1)/10;
5) k:=k/3; 6) j:=x + y;
7) x:=k; 8) x:=x + *y;*
9) y:=k + x; 10) j:=x + 0.9.

9. Чому дорівнюватимуть значення змінних x та *y* після виконання таких дій:

x:=2, *y:=5,* x:=*y;* *y* := x?

Домашнє завдання:

вивчити означення, що прочитані на лекції.

1. Задати за допомогою оператора присвоювання такі дії:
6) змінній z присвоїти значення, що дорівнює півсумі значень змінних x та *y;*
7) подвоїти значення змінної a;
8) значення змінної x збільшити на 0.1;
9) за нове значення змінної *a* прийняти її поточне значення, піднесене до квадрата;
10)змінити знак у значенні змінної *q.*

2. Змінній *a* присвоїти значення, що дорівнює сумі значень змінних x та *y,* а змінній *b* - подвоєному їх добутку.

6. Визначити, за допомогою якої послідовності дій можна поміняти місцями вміст змінних x та *y:*

1) x := x + y; *2)*x:=*y;* 3) *y* := x + *y;* *4)* x := x + *y;*
y:=*y*-*x;* x:=*y* - *x;* x := *y* - x; *y* := x - *y;*
x:=*y;* *y*:=x-*y;* *y* := *y* - x; x:=x-*y.*

4. Записати у вигляді оператора присвоювання обчислення виразу *y* = 2x^3 + 3x^2 + x + 5, не використовуючи при цьому операції піднесення до степеня.

5. Яке значення будуть мати змінні x та y після виконання операторів:

А) x:= 178 y:= x Div 10 e:= x Mod 10;
Б) x:= 123 y:= 1234 x:= (x Div 100)Mod 10 y:= (y Mod 1000) Div 100;
В) x:= 3456 y:=12345 x:=(x Div 100)Mod 10 y:=(y Mod 1000)Div 10;
Г) x:= 23456 y:= 12345 x:= (x Mod 10000) Div 10 y:= (x Mod 100) Div 10.

6. Обчислити значення виразів

а)trunc(6,9); б) round(6,9); в)tranc(6,2); г)round(6,2); д) trunc(-l,8); є) round(-l,8); ж) round(0,5); з)round(-0,5); і)20 div 6; к)123 div 0;
л) 3.0mod 3; м)20 mod 6; н) 2 div 5; о) 2 mod 5; п) 3 * 7div2 mod 7 / 3 - trunc(sin(1)); р) succ(round(5/2)-pred(3)).

Створення лінійних програм

1. Структура програми. Програма складається із заголовка
program <ім'я програми>;
розділів описової частини

uses label	— приєднання бібліотек та модулів; — оголошення міток (позначок);
const	— оголошення сталих;
type	— опис типів;
var	— оголошення змінних;
procedure	— оголошення процедур користувача;
function	— оголошення функцій користувача

та виконуваної частини
begin
<розділ команд>
end.

Заголовок та усі розділи, окрім останнього, є необов'язковими. Розділювачем між конструкціями (командами) програми є символ ";". У кінці програми завжди має стояти крапка.

Заголовок програмі надає програміст. В *іменах,* які користувач дає своїм програмам та змінним, великі і малі букви рівноправні: імена A та a (або MyName та myname) позначають один і той самий об'єкт.

У програму можуть входити коментарі. ***Коментар*** — фрагмент тексту програми, взятий в фігурні дужки або записаний так: (* коментар *). Коментар слугує для пояснення роботи програми і не впливає на виконання команд. Він може бути розташований у довільному місці програми.

2. Розділи оголошення сталих і змінних. Усі величини, які входять у програму, повинні бути описані у розділі сталих (констант), якщо вони не мінятимуть значення протягом виконання Програми:

const <стала 1> = значення 1>;

або у розділі оголошення змінних, якщо вони обчислюватимуться:

```
var     <список змінних 1> : <тип змінних 1>;
        <список змінних n > : <тип змінних n>;
```

Елементи списків записують через кому. Кутові дужки <...> — це засіб формалізованого описування конструкцій мови. У конкретних програмах їх не використовують.

3. Перша програма. Програма — це послідовність команд, за допомогою яких записують алгоритм розв'язування задачі. Програми (алгоритми) складають за таким принципом: ***вводять*** дані, ***визначають*** потрібне, ***виводять*** результати. Аналогічно розв'язують задачі з математики та фізики, але тут обчислення вручну виконувати не потрібно — їх виконає комп'ютер.

Розглянемо програму з назвою Trykutnyk для розв'язування задачі обчислення периметра *p* та площі s трикутника зі сторонами a = 5, b=3.6, c=4.2 за формулою Герона. Усі команди, наведені в програмі, будуть детально розглянуті нижче.

```
program Trykutnyk;
uses Crt;                       {Приєднуємо модуль Crt}
const a=5; b=3.6; c=4.2;        {Вводимо довжини сторін}
var p,s: real;                  {Оголошуємо змінні для]
begin                           {периметра та площі}
   clrscr;                      {Очищуємо екран}
   p:=a+b+c;                    {Обчислюємо периметр}
   writeln('p=', p:5:2);        {Виводимо значення периметра}
   p:=p/2;                      {Обчислюємо півпериметр}
   s:=sqrt(p*(p-a)*(p-b)*(p-c)); {Визначаємо площу}
   writeln('s=',  s:5:2);       {Виводимо значення площі}
writeln('Виконав Іванов П.');
readln
end.
```

Після виконання програми на екрані отримаємо:
p= 12.80 s= 7.43

Прості команди алгоритмів: присвоювання, введення і виведення.

4. Команди введення (read, readln) даних. Надавати значення змінним можна двома способами: за допомогою команди присвоювання, наприклад x:=5, або команд уведення даних з клавіатури. Другий спосіб робить програму більш універсальною, оскільки дає змогу розв'язувати задачі для різних значень змінних. Команда read має вигляд

Read(<змінна 1>,…<змінна n>);

Дія команди. Виконання програми зупиняється. Система переходить у режим очікування введення даних (екран темний, миготить курсор). Значення цих даних користувач набирає на клавіатурі через пропуск або натискає після кожного даного на клавішу вводу. У результаті виконання цієї команди відповідним змінним будуть присвоєні конкретні значення.

Команда readln має вигляд

readln(<змінна 1>,...,<змінна n>);

Вона діє як команда read з тою різницею, що зайві дані у рядку введення ігноруються. Наступна команда вводу читатиме дані нового рядка. Цю команду застосовують під час роботи з текстовими файлами.

Зауваження. Команду readln без параметрів часто використовують у середовищі TP для MS-DOS, щоб оглянути результати виконання програми на екрані. Щоб після цього перейти у режим редагування програми, потрібно натиснути на клавішу вводу.

Зауваження. Значення змінних логічного й перерахованого типу вводити з клавіатури *не можна.*

5. Команди виведення (write, writeln) даних. Для виведення на екран повідомлень та результатів обчислень використовують команди write та writeln:

write(<вираз 1>,< вираз 2> ..... <вираз n>);

У списку виведення можуть бути сталі, змінні або вирази.

Дія команди. Сталі, значення змінних та виразів виводяться на екран у вікно виведення, яке можна переглянути за допомогою комбінації клавіш Alt+F5. Команда

writeln(<вираз 1>,...,<вираз n>);

діє майже так само як і команда write; різниця така: наступна після неї команда write чи writeln буде виводити значення на екран у новому рядку.

Для переходу на новий рядок екрана чи для пропуску рядка використовують команду writeln без параметрів.

6. Форматний вивід. Команди write та writeln можуть здійснювати форматний вивід даних. Форматування — це подання результатів у наперед заданому користувачем вигляді. Для цього після виразу через двокрапку записують число (:n) — кількість позицій на екрані, які треба надати для виведення значення цього виразу. Формат :n застосовують для даних *цілого та рядкового* типів. Під час виведення даного *дійсного* типу зазначають загальну кількість позицій для всіх символів (n) та кількість позицій для дробової частини (m),тобто формат має вигляд :n:m.

Задача 1. Дано координати трьох вершин трикутника A(1;1), B(2;2) та C(-1;2). Обчислити медіану t_b та радіус описаного кола *r*.

```
program TrykutnykNew;
uses Crt;
var xl,yl,x2,y2,x3,y3,a,b,c,mb,r,x,y,p,s: real;
begin
    clrscr;
    writeln('введіть координати:');
    readln(xl,yl,x2,y2,x3,y3);
    a:=sqrt(sqr(x3-x2)+sqr(y3-y2)); {Обчислимо довжини}
    b:=sqrt(sqr(xl-x3)+sqr(yl-y3)); {сторін трикутника}
    c:=sqrt(sqr(xl-x2)+sqr(yl-y2));
    x:=(xl+x3)/2;                   {Обчислимо координати)
    y:=(yl+y3)/2;                   {середини сторони b)
    mb:=sqrt(sqr(x-x2)+sqr(y-y2));  {Обчислимо медіану mb}
    p:=(a+b+c)/2;                   {Обчислимо півпериметр}
    s:=sqrt(p*(p-a)*(p-b)*(p-c));   {Обчислимо площу}
    r:=a*b*c/(4*s);                 {Обчислимо радіус}
    writeln('mb=',mb:5:2);          {Виведемо результати}
    writeln('r=',r:5:2);            {Виведемо радіус}
    readln
end.
```

Задача 2. Написати програму визначення суми цифр тризначного числа.

```
program proba_1;
var x, rez: integer; begin
writeln('Задайте тризначне число:');
```

```
readin(x);
rez:=(x mod 10)+((xdiv 10) mod 10)+(x div 100);
writeln(‘ Сума цифр заданого числа: ‘, rez); readln; end.
```

7. Зворотний запис числа. Дано тризначне число *x = abc (a, b, c* - його цифри). Потрібно одержати число, записане тими ж цифрами, але у зворотному порядку. Тобто, якщо дано число 128, то одержати треба 821. Перед нами встає завдання знаходження цифр числа — *a, b* і *c.* Розмістити ці цифри у зворотному порядку легко — результат буде дорівнювати *x* = 100 c + 10 *b* + *a.* Найпростіше знайти цифру одиниць с — вона буде дорівнювати залишку від ділення числа x на 10 (наприклад, якщо *x* = 128, те *x* mod 10 буде 8, що нам і потрібно). Отже,

c := x mod 10;

Щоб знайти *b,* спочатку знайдемо *ab,* рівне *x,* діленому націло на 10 (для 128 це буде 12), а потім уже визначимо *y* — цифру одиниць числа, що вийшло після останніх дій :

b := x div 10 mod 10;

Знайти *a* просто — це результат ділення *x* націло на 100:

a :.= x div 100;

Тепер можна написати всю програму:

```
program naoborot;
var x, a, b, з : Integer;
begin
  Write('x==>');
  ReadLn(x);
  c := x mod 10;
  b := x div 10 mod 10;
  a := x div 100;
  Write(100 * з + 10 * b + a) ;
  end.
```

Задача 1. Скласти програму обчислення:

4) Середнього арифметичного чисел A, B, C;
5) Площі прямокутного трикутника з катетами A і B;
6) Суми цифр тризначного числа A.

Задача 2. Обчислити добуток трьох дійсних чисел.

Задача 3. Обчислити значення виразу $\frac{a^2 - 2ab - 3}{2}$

де *a* й *b* — дійсні числа.

Задача 4. Дано чотиризначне число *x = abcd.* Одержати число, записане тими ж цифрами у зворотному порядку *(y = dcba).*

Задача 5. Дано тризначне число *x = abc.* Знайти суму квадратів його цифр.

Домашнє завдання.

Підготувати відповіді на запитання:

5. Яку структуру повинна мати правильно написана програма на мові програмування Паскаль?
6. Які блоки в програмі обов’язкові, а які ні?
7. Що таке лінійна програма?
8. Які ви знаєте вказівки введення та виведення інформації?

Скласти програму, задавши вхідні дані самостійно.

5. Радіус Місяця 1740км. Обчислити площу поверхні $S=4\pi r^2$ та об’єм планети $V=(4/3)\pi r^3$.
6. Обчислити кінетичну $E=mv^2/2$ та потенціальну $P=mgh$ енергії тіла заданої маси *m*, яке рухається на висоті *h* зі швидкістю *v*.
7. Скільки секунд мають доба, тиждень, рік?
8. Задано двозначне число. Визначити суму цифр цього числа.

Розв’язування задач на створення лінійних програм

Розв’язання прикладів на повторення:

Приклад 1. Якщо значення цілочисельних змінних таке: *X*= 15, *Y*= 25, *Z*= 8, то чому дорівнюватимуть значення цих змінних після виконання операцій присвоювання:

X := sqrt(Y)*2;
Y := Y-2;
Z := Y div 2 mod 3;

Очікувана відповідь: **X:= sqrt(25)*2** —тут присвоєння не може бути виконаним, оскільки значення квадратного кореня — дійсне число, а *X*—ціле число

Y:= **25-2** Y = 23
Z:= 23 **div** 2 **mod** 3 Z = 2

Приклад 2. Написати команду присвоєння, що надає значення середнього арифметичного змінних *X та* У змінній *Z* (Тип змінних дійсний).
Очікувана відповідь: **Z : = (X + У) / 2**
Приклад 3. За допомогою яких операторів присвоєння можна поміняти місцями значення двох змінних *X та Y.*
Очікувана відповідь: Для цього необхідне використання третьої змінної того ж типу, що й змінні X та У, наприклад, Z: Z := X; X :=Y; Y := Z

Розв'язування задач

Задача 1.

Якщо на одну шальку терезів посадити Даринку, яка важить N кг, і Наталку, яка важить на 5кг менше, а на іншу насипати М кг цукерок, то скільки кілограмів цукерок доведеться з'їсти дівчаткам, щоб шальки терезів зрівноважилися?

Введемо змінні для зберігання результатів: *N*— вага Даринки; *M* — вага цукерок; *P*— вага цукерок, які необхідно з'їсти дівчаткам.

Тоді програма для розв'язання задачі буде такою:

```
Program Task_1;
Uses  crt;
Var M, N, P : real;
Begin
Clrscr;
  Write('Введіть вагу Даринки'); Readln(N);
  Write('Введіть вагу цукерок, що лежать на терезах'); Readln(M);
  P :=N+N-5-M; {N-5- вага Наталки)
  Writeln("Дівчаткам необхідно з'їсти ' ,P, 'кг цукерок.'); Readln;{Процедура затримує зображення на
  екрані до натискання клавіші Enter}
  End.
```

Задача 2

Визначити, яку платню одержить на фірмі сумісник за виконану роботу, якщо йому нараховано S грн., а податок становить 20%.

Необхідні змінні: *S* — сума нарахувань сумісника; *P*—реальна платня, яку він одержить у касі (за умовою вона становить 80 % від нарахувань).

Програма має наступний вигляд:

```
Program Task_2;
Uses crt;
Var  P,S   :   real;
Begin
Clrscr;
Write ('Введіть суму нарахувань робітника') ; Readln(S); P   := S*0.8;
Writeln('Платня сумісника становить:',  P:8:2);
Readkey;
End.
```

Задача 3. Від міста А до В автомобіль їхав t_1 = 5 год з середньою швидкістю v_1 = 70 км/год, від В до С — t_2 = 4 год. зі швидкістю v_2 = 75 км/год. Визначити відстань між містами.

```
program Distance;
var t1, vl, t2, v2, ab, be, ac : integer;
begin
tl := 5 ; t2 := 4;  vl := 70 ; v2 := 75;
ab := vl * tl; be := v2 * t2; ac :== ab +bc;
writeln (ab:6, bc:6, ac:6);
readln
end.
```

Виконаємо програму і на екрані отримаємо: **350 300** 650. *Завдання.* Модифікуйте програму на випадок чотирьох міст.

Задача 4. Автотранспортна фірма «Радар» купує п'ять (kl) мікроавтобусів «Пежо» по 32100 грн. (c1) за автобус і три (к2) мікроавтобуси «Івеко» по 29500 грн. (c2). Яку суму (suma) потрібно заплатити?

```
program Radar;
```

```
var kl, cl, k2, c2, suma : integer;
   begin
kl := 5 ; k2 := 3;  cl := 32100 ; c2 := 29500;
suma := kl * cl + k2 * c2;
writeln (suma:8);
readln
end.
```

Виконаємо програму, і замість результату отримаємо повідомлення про помилку. Виявляється, що значення змінної suma є 249 000, і воно вийшло за допустимі межі, визначені для типу **integer.** Ось чому правильно оголосити змінні потрібно так:

var kl, k2, cl, c2 : integer; suma : longint; .

Отже, щоб розв'язати задачу, потрібна додаткова пам'ять, оскільки змінні типу **longint** займають удвічі більше пам'яті, ніж змінні типу integer.

Завдання. Виконайте програму Radar і поекспериментуйте з різними цінами. Модифікуйте програму, якщо купують три марки автобусів.

Задача 5. Ввести з клавіатури будь-яке тризначне число. Визначити суму його цифр і вивести цифри числа у зворотному порядку.

Нехай змінна a міститиме значення заданого числа. Цифри числа позначимо так: i — кількість сотень, j — кількість десятків, k — кількість одиниць, а їхню суму — s. Для визначення цифр деякого числа використовують операції **div** та **mod.**

```
program MyNumber; var a, i, j, k, s : integer; begin
      write('Введіть число a: ');read (a);
      i := a div 100;    {Отримаємо кількість сотень}
      j := a div  10 mod 10; {Отримаємо к-сть десятків}
      k := a mod 10;  {Отримаємо кількість одиниць}
      s := i + j + k;
      writeln('Сума цифр числа a =  ', s);
  writeln(k,j,i)
 end.
```

Виконаємо програму. Введемо число 235 - отримаємо результат Суми цифр числа a = **10**

Вправи та задачі для самостійного розв'язання

6. Швидкість світла 299792 км/с. Яку відстань долає світло за хвилину, годину, добу?
7. Квіткова клумба має форму круга. Обчисліть її периметр і площу за заданим радіусом.
8. Тіло падає з прискоренням g. визначіть пройдений тілом шлях $h=gt^2/2$ після першої та другої секунди падіння.
9. Яка маса золотої кульки заданого радіуса, наприклад 0.02м, якщо густина золота 19300кг/м^3? ($M=\rho \cdot V$, $V=(4/3)\pi r^3$)
10. Яка маса зливка срібла у вигляді куба з заданою стороною, наприклад 6см, якщо густина срібла 10500кг/м^3?

Домашнє завдання.

Виконайте вправи та задачі.

3. Нехай a=0. Якого значення набуде змінна a, якщо команду a:= a+2 виконати: а) один раз; б) два рази підряд; в) три рази підряд.
4. Нехай a=1. Якого значення набуде змінна a, якщо команду a:= a*2 виконати: а) один раз; б) два рази підряд; в) три рази підряд.

3. Які значення матимуть змінні (та якого вони є типу) після виконання команд присвоювання, якщо раніше були виконані команди A:=2; B:=5; C:=0:

а) A1 := (2*A - 3*B)/(11-2*B);
б) A2 := A/2 + B+5*A/(B+C);
в) A3 := 3 + 25 div 4 + 7 mod 3;
г) A4 := 110 div 100 +110 mod 100 + 110 mod 10;
д) A5 := B div A + B/A + B mod A (відповідь: 5.5)?

7. Уведіть тризначне ціле число. Визначте суму і добуток крайніх цифр.

8. Уведіть чотиризначне число. Виведіть його цифри а) у стовпчик; б) у зворотному порядку.

9. Уведіть п'ятизначне ціле число. Обчисліть суму його цифр.

7. Задано координати вершин трикутника у площині: (x_1 ; y_1), (x_2; y_2), (x_3; y_3). Обчисліть довжини його сторін (складіть програму).

Довідка. Відстань між двома точками, які задані координатами (x_1 ; y_1) та (x_2; y_2), визначають так:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\ .$$

9. Задано координати вершин чотирикутника у площині: (x_1 ; y_1), (x_2; y_2), (x_3; y_3), (x_4; y_4). Складіть програму для обчислення довжини його сторін.

Модуль CRT. Графічний режим роботи

1. **Ініціалізація графічного режиму**

Після запуску інтегрованого середовища Турбо Паскаля включається текстовий режим і для використання графіки необхідно виконати ряд дій по переходу в графічний режим.

Насамперед необхідно підключити модуль Graph Турбо Паскаля. У цьому модулі описані процедури й функції, призначені для роботи із графічним екраном, а також деякі вбудовані константи й змінні, які можуть бути використані в графічних програмах. Для того щоб скористатися всіма можливостями модуля Graph, на початку програми (після її заголовка) необхідно розмістити оператор: uses Graph;

Основну частину модуля становлять процедури виводу основних графічних елементів, таких як точки, відрізки прямих ліній, дуги, цілі кола та ін. Такі елементи називаються *графічними примітивами.* Інша група процедур призначена для керування графічним режимом. Усього в бібліотеці модуля Graph перебуває більше 50 процедур і функцій для роботи із графікою.

Далі варто визначити тип відеоадаптера, установленого на комп'ютері. Відеоадаптером називають набір мікросхем, керуючих роботою конкретного дисплея. Можна доручити з'ясувати тип відеоадаптера програмі.

Як майже всякий фізичний пристрій комп'ютера, відеоадаптер може працювати тільки в тому випадку, коли завантажена програма, що управляє його роботою. Така програма називається *драйвером* пристрою (driver у перекладі з англійської мови — «шофер», так що драйвер є «водієм», що управляє роботою пристрою). Графічні драйвери містяться у файлах з розширенням **.BGI** (наприклад, **EGAVGA.BGI** або **IBM8514.BGI). У** Турбо Паскалі вже є необхідний набір .BGI-Файлів, тому програмістові залишається лише вказати у відповідному місці програми розташування каталогу, що містить ці файли.

Графічний режим спочатку треба задати. Це виконують так:

```
<розділи описів та оголошень конкретної програми>;
var driver, mode : integer;    {Для характеристик дисплея}
begin
   driver:= detect; {detect - стандартна стала}
   initgraph (driver, mode, ''); {Задания графічного режиму}
   if graphresult < > 0 then begin
writeln('графічний режим задати не вдалося');
halt            {Стоп}
end;
<текст конкретної програми з графічними командами>
end.
```

2. Процедури і функції для графічних побудов. Розглянемо процедури модуля **Graph,** призначені для графічних побудов.

initgraph (driver, mode, <шлях до драйвера>) — задає графічний режим. Шлях до драйвера зазначають (у лапках), якщо він не є в тому ж каталозі, що й файл turbo.exe;

detectgraph (<драйвер>, <режим>) — повертає значення характеристик дисплея;

setcolor (<колір>) — задає колір майбутнього зображення;

setbkcolor (<колір>) — задає колір тла;

putpixel (x, y, <колір>) — висвітлює точку (x,y) заданим кольором;

line (xl, yl, x2, y2) — рисує лінію між двома точками;

lineto (x, y) — рисує лінію від поточної точки до точки (x,y);

linerel (dx, dy) — рисує лінію від поточної точки з заданими приростами;

rectangle (xl, yl, x2, y2) — рисує прямокутник з заданими координатами діагонально протилежних вершин (лівої верхньої та правої нижньої);

setviewport (xl, yl, x2, y2, true) — задає координати нового графічного вікна. Логічна стала true задає режим відсікання зображення, яке виходитиме за межі вікна;

bar (xl, yl, x2, y2) — рисує зафарбований прямокутник; **bar3d** (xl, yl, x2, y2, <об'ємна глибина>, true) — рисує паралелепіпед;
circle (x, y, **R)** — рисує коло з радіусом R і центром у (x,y); **arc** (x, y, <початковий кут>, <кінцевий кут>, <радіус>) — рисує дугу; **pieslice** (x, y, <початковий кут>, <кінцевий кут>, <радіус>)
— рисує зафарбований сектор;
ellipse (x, y, <початковий кут>, <кінцевий кут>, <горизонт. радіус>, <вертик. радіус>) — рисує еліпс чи дугу еліпса;
setfillstyle (<заповнення>, <колір>) — задає спосіб заповнення замкнутої області залежно від значення параметра заповнення: 0 — заповнення кольором фону, 1 — суцільне заповнення, 2 — заповнення товстими горизонтальними лініями, 3 — заповнення нахиленими лініями, ..., 10
— заповнення точками, 11— щільне заповнення точками;
floodfill (x, y, <колір межі>) — заповнює замкнену область,
що містить точку (x,y); **closegraph** — закриває графічний режим;
outtext (<текст>) — виводить заданий текст з поточної позиції; **outtextxy** (x, y, <текст>) — виводить текст у заданому місці; **settextstyle** (<шрифт>, <напрям>, <розмір>) — задає вигляд
символів, напрямок виведення: 0 — горизонтально чи 1 — вертикально, і розміри символів: 1, 2, 3.

3. Розглянемо деякі функції модуля **Graph.**

graphresult — повертає код помилки, якщо неможливо задати графічний режим, і 0 — у разі задания;
getmaxx — повертає значення розміру екрана уздовж осі OX;
getmaxy — повертає значення розміру екрана уздовж осі 0У;
getcolor — повертає значення поточного кольору;
getcolor(x.y) — повертає значення кольору точки (x,y);
getx, gety — повертають координати поточного пікселя.

4. Кольори. Кольори задають числами або англійськими назвами:

black=0 - чорний;	darkgray=8 - темно-сірий;
blue=l - синій;	lightblue=9 - яскраво-синій;
green=2 - зелений;	lightgreen=10 - яскраво-зелений;
cуап=3 - блакитний;	lightcyan=ll - яскраво-блакитн.;
red=4 — червоний;	lightred=12 — яскраво-червоний;
magenta=5 - фіолетовий;	lightmagenta=13 - яскраво-фіол.;
brown=6 — коричневий;	yellow=14 - жовтий;
lightgray=7 - світло-сірий;	white=15 - білий.

Задача 1. Нарисувати різними кольорами десять концентричних кіл, які мають спільний центр по середині екрана, тобто в точці з графічними координатами (320; 240), і описати навколо кіл червоний прямокутник.

```
program Circle10;
uses Crt, Graph;
var driver, mode, r : integer;
begin  clrscr;
    driver := detect; initgraph( driver, mode, '');
    r := 10;       {Радіус першого кола 10 пікселів)
    while r <= 100 do
    begin
    setcolor(r div 10); circle(320, 240, r);
    r := r + 10 end; setcolor(red);
rectangle(220, 140, 420,340); readln
end.
```

Задача 2. Нарисувати емблему. У верхній лівій частині графічного екрана на чорному фоні нарисувати блакитний квадрат, а в ньому - чорне коло, зафарбоване жовтим кольором. У центрі емблеми чорними літерами написати слово "Балта".

```
program Emblema;
uses Crt, Graph;
var driver, mode: integer;
begin  clrscr;
driver:—detect;
initgraph(driver,mode,'');
setbkcolor(O);
```

```
setcolor(3);
rectangle(100,0,300,200);
setfillstyle(l,3);
floodfill(200,100,3);
setcolor(14);
circle( 200,100,100);
setfillstyle(l,14);
floodfill(200,100,14);
setcolor(O);
circle(200,100,100);
settextstyle(0,0,3); outtextxy(135,95, 'Балта'); readln end.
```

Побудова графіків функцій. Графічні команди використовують також для рисування графіків функцій.

Задача3. Зобразити на екрані графік функції y=sinx.

```
program Grafiksin;
uses crt, graph;
var driver, mode, i, x1, y1 : integer; x, y : real;
begin    clrscr; driver := detect;
initgraph(driver, mode,'');
setcolor(4); setbkcolor(11);
setlinestyle(0,1,3);
line(20, 240, 450, 240); line (60, 340, 60, 120);
x:=0;    x1:=60; y1:=240;
Moveto(x1, y1); setcolor(8);
while x<=2*pi+0.1 do  {розгл. відрізок [0;2π]}
begin
            y:= sin(x);
            y1:= - trunc(100*y) + 240;
lineto(x1, y1);
x:=x+0.1; x1:=x1+5
end;
settextstyle(0, 0, 1); outtextxy(60, 245, '0');
outtextxy(360, 245, '6.3'); settextstyle(0, 0, 2);
outtextxy(80, 100, 'Графік функції sin(x)')
 end.
```

Задача 4. Різнобарвні смуги. Намалювати 14 різнобарвних вертикальних смуг, пофарбованих 14 кольорами (крім білого й чорного), можна за допомогою процедур малювання ліній і установки кольору:

```
program stripes;
uses Graph;
 var gd,  gm,  c,  x,  y,  i  :  Integer;
begin
     gd   := Detect;
     InitGraph(gd, gm, '');
     SetBkColor(white);
     ClearDevice;
     x := 0;
     for c := 1 to 14 do
     begin
       x := x + 35;
       SetColor(c) ;
       Line(x, 0, x, 400);
       for i :=1 to 5 do
       Line(x + i, 0, x + i, 400) ;
                                        {Малюємо 6 смуг}
end;
    Readln;
    CloseGraph;
  end.
```

Задача 5. **Малюємо Королеву Краси.** Ну от, тепер можна спробувати намалювати царівну Будур або хоча б схожого на неї зображення.

```
program man; uses Graph;
var gd,  gm  :  Integer;
begin
  gd  :=  Detect;
  InitGraph(gd,  gm,  ' ');
  SetFiUStyleU,   Green);{Трава}
  Bar(0,   350,   639, 479);
  SetFillStyle(1, LightBlue);{Небо}
  FloodFill(0,   0, Green) ;
  SetColor(Red);
  Circle(320, 200, 19);{Голова}
  SetLineStyle(0, 0, 3);
  Rectangle(300, 220, 340, 300);{Тулуб}
Line(320, 300, 300, 350);{Ноги}
  Line(320, 300, 340, 350);
  Line(300, 240, 250, 250); {Руки}
  Line(340, 240, 390, 250);
  SetFillStyle(1, Red);
 FloodFill(320, 200, Red)
 FloodFill(320, 230, Red)
 SetColor(Yellow) ;
 Circle(315, 190, 2):, {Ліве око}
 Circle(325, 190, 2); {Праве око}
 Line(315, 210, 325, 210) {Рот}
 Readln;
 CloseGraph;
 end.
```

Ну як? Намальована програмою дівчина... звичайно, далеко не прекрасна дочка султана! Але в Турбо Паскалі є досить засобів і можливостей для того, щоб створити по-справжньому гарну картинку.

Задача 6. **Павутина.** Програма web («павутина») виводить на екран зображення, складене з відрізків прямих і концентричних кіл. Загальний центр точки перетину відрізків і кіл знаходиться в центрі екрана. Тут використовуються функції GetMaxX і GetMaxY. Графічні координати правого нижнього кута екрана рівні (GetMaxX, GetMaxY).

У програмі використовується процедура Delay модуля Crt. Ця процедура припиняє виконання програми на зазначену кількість мілісекунд.

Програма припиняє свою роботу при натисканні клавіші **Enter.**

```
program web;
uses CRT, Graph;
var i : Word; gd, gm : Integer;
begin
gd := Detect;
gm := 0;
InitGraph(gd, gm, '');
SetBkColor(Blue);
SetColor(LightCyan) ;
Line(0, 0, GetMaxX, GetMaxY);
Delay(1000);
SetColor(Yellow);
Line(0, GetMaxY, GetMaxX, 0);
Delay(1000);
SetColor(LightGreen) ;
Line(0, GetMaxY div 2, GetMaxX, GetMaxY div 2);
Delay(1000);
```

```
SetColor(LightGray) ;
Line(GetMaxX div 2, 0, GetMaxX div 2, GetMaxY);
Delay(1000);
SetColor(LightRed);
for i:=2 to 20 do
begin SetColor(16 - i div 2);
Circle(GetMaxX div 2, GetMaxY div 2, GetMaxY div i);
Delay(500 - 15 * i);
end;
ReadLn;
 CloseGraph;
 end.
```

Домашнє завдання.

Побудувати трикутник з вершинами в точках (100, 100), (150, 100), (80, 70). Колір фону – сірий, колір ліній – червоний.

Елементи комп'ютерної графіки.

Мета: ознайомлення з функціями роботи в графічному режимі; формування навичок роботи в графічному режимі екрану; виховання інформаційної культури.

На цьому уроці пропоную розв'язати цікаві задачі із застосуванням графічного режиму роботи монітору.

Тип уроку: формування навичок.

Хід уроку.

I. Актуалізація опорних знань учнів

- Як задати графічний режим?
- Як нарисувати пряму лінію?
- Як нарисувати коло?

II. Формування практичних навичок та вмінь.

Розв'язування задач.

Задача 1.

Умова: *Скласти програму, яка при натисканні клавіші Д (день) малює сонце, а при натисканні клавіші Н (ніч) малює місяць.*

Розв'язування: По-перше, для вибору малюнку (день чи ніч) введемо символьну змінну *Ch,* залежно від значення якої і будемо малювати сонце чи місяць. По-друге, малювання сонця складається з малювання зафарбованого кола процедурою *FillEllipse* (ця процедура малює зафарбований еліпс, але, якщо еліпс має однакові радіуси по осям, то він перетворюється на коло) та кількох прямих (променів), а місяць можна отримати, якщо накласти одне на одне два кола різних кольорів (жовтого та чорного) з деяким зміщенням. Програма має вигляд:

```
Program Example_1;
Uses graph,crt;  {Підключення бібліотек}
Var GraphDriver,GraphMode:integer;
Ch:char;
Begin  Clrscr;
Writeln('Введіть Ваш вибір:  Д - день,  Н - ніч.');
Readln(ch);
 GraphDriver:=VGA;  {Ініціалізація графічного режиму}
 GraphMode:=VGAHi;
 InitGraph(GraphDriver,GraphMode,'' );
  if  (Ch='Д')  or  (Ch='д')   then
  begin setfillstyle(I,yellow);
 setcolor(yellow);
   fillellipse(100,80,50,50);  {Малювання сонця}
   {Малювання променів}
 line(100,80,250, 80) ;   line(100,80,240,30) ;
 line(100,80,200,250);  line(100,80,230,180);
 line(100,80,150,250);  line(100,80,100,300);
 line(100,80,50,380);  line(100,80,20,280) ;
 line(100,80,0,150);  line(100,80,0,80);
 line(100,80,0,30);  line(100,80,10,0) ;
 line(100,80,50,0);  line(100,80,100,0);
```

```
 line(100,80,150,0);
  end
 else
 if  (Ch='H')  or  (Ch='H')    then
 begin
  setfillstyle(1,yellow);  setcolor(yellow);
  fillellipse(100,80,50,50);  setfillstyle(1,black);
  setcolor(black);  fillellipse(130,80,50,50);
  end
 else writeln (' Ви помилилися!') ;
 Readkey;     Closegraph;
 End.
```

Задача 2. ***Умова:*** *«Зоряне небо». Заповнити екран монітора різнокольоровими точками, кількість яких, колір та координати визначаються випадково.*

Розв'язання: Для вибору випадковим чином вказаних величин скористуємось функцією *Random,* що вибирає числа із заданого діапазону, д причому врахуємо, що, якщо в дужках після функції вказане ціле число, то будуть генеруватися цілі числа в діапазоні від 0 до вказаного числа. Зверніть увагу на те, що всього можливих кольорів 16 (від 0 до 15), але на чорному тлі чорний колір (з нульовим номером) не видимий, тому можна скористатися такою формулою для отримання ненульових цілих чисел в діапазоні від 1 до 15: random (14) +1

Аналогічно можна вибрати координати та кількість «зірок» (точок) на екрані, причому відслідкувати, щоб кількість ніколи не була нульовою. Сама «зірка» (точка) на екрані може бути отримана процедурою *Putpixel,* що задає колір та координати точки виведення. Програма має вигляд:

```
Program Example_2;
Uses  graph;
Var GraphDriver,GraphMode:integer;
      x,y,color,N:integer;   {x,y - координати точки  -   'зірки', color - колір точки,  N - кількість точок)
     i:integer;   {i - змінна циклу)
     Begin Randomize;
     GraphDriver:=VGA;
     GraphMode:=VGAHi; InitGraph(GraphDriver,GraphMode,' ') ;
      {Генерується кількість точок в діапазоні від 200 до 1200} N:=random(1000)+200;
      for i:=1  to N do
      begin x:=random(640);   y:=random(480);   color:=random(14)+1; putpixel (x,y,color) ;  {Виведення піксела
      заданого кольору
               color у задані координати екрану x та y}
                end;
      Readkey;     Closegraph;
      End.
```

Запитання для перевірки засвоєння знань

6. *Які процедури визначають тип ліній?*
7. *Як скористатися процедурою малювання еліпса для малювання кола?*
8. *Що задає процедура* ***PutPixel?***
9. *Якими процедурами визначається тип зафарбування?*
10. *Які процедури дають змогу намалювати пряму лінію?*

Домашнє завдання:

Побудувати і замалювати зеленим кольором круг радіусом 100, центр якого співпадає з центром екрана дисплея. Колір фону – малиновий.

Організація розгалужень

Логічні операції та вирази

1. Логічні *вирази. Обчислення значень логічних виразів*

Крім арифметичних виразів, у Pascal існує ще один тип виразів – логічний.

Логічним виразом називається такий вираз, внаслідок обчислення якого одержується логічне значення типу true або **false («істина» або «хиба»).**

Із стандартним типом змінних **Boolean,** які можуть набувати лише двох значень **True** або **False,** ви вже ознайомилися. Отже, саме такий тип і мають результати обчислення логічних виразів.

Логічні вирази поділяються на ***прості*** та ***складені.***

Простим логічним виразом називається вираз, який записується за допомогою знаків співвідношень <, >, <=, >=, =, та <>.

Приклади простих логічних виразів можуть здатися вам простими:

$a + b > c + d$, $n > m$, $x = y$.

Порівняйте тепер призначення символів «:=» та «=»! Зверніть увагу на те, що спочатку виконуються арифметичні дії, а вже потім порівняння отриманих результатів.

Складеним логічним виразом називається вираз, в якому використовуються логічні операції and, or, not («так», «або», «ні»).

Наведемо приклади. З математики вам відомі такі записи:

$x \in [a, b]$ та $x \notin [a, b]$.

Спробуємо записати їх у вигляді логічних виразів

(x >= a) **and** (x <= b),

(x<a)or(x>b).

Під час запису складених логічних виразів прості логічні вирази обов'язково слід брати у круглі дужки!

Чи можна записати простий логічний вираз n<> m у вигляді складеного? Виявляється, можна:

not (n = m).

Визначимо правила, за якими обчислюються значення складених логічних виразів. Для цього існують таблиці істинності, в яких цифра 0 означає **false,** а цифра 1 - **true.** Наведені таблиці можна перефразувати таким чином.

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

***Логічна операція* and *дає результат* true *тоді і тільки тоді, коли обидва операнда мають значення* true.**

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

***Логічна операція* or *дає результат* true *тоді, коли хоча б один операнд має значення* true.**

A	not A
0	1
1	0

Логічна операція* not *завжди дає результат, протилежний значенню її операнда.

Вираз	Значення	Вираз	Значення
not true	false	**not** false	true
true **and** true	true	true or true	true
true **and** false	false	true or false	true
false **and** true	false	false or true	true
false **and** false	false	false or false	false

Приклад. Нехай x=3, y= -9. Розглянемо деякі логічні вирази та їхні значення.

Прості вирази	Значення	Складені вирази	Значення

x=3	true	not(y - 50)	true
x>y	true	(1<x) and (x<5)	true
7 mod 3=1	true	(x>4) or (y<-15)	false
y div 2=4	false	(x>4) or (y>-15)	true

Подвійну нерівність 1<x<5 як складений логічний вираз записують так: (1<x) and (x<5). Сукупність нерівностей вигляду x<1; x>5 — так: (x<1) or (x>5). *Прості логічні вирази, які входять у складені, завжди беруть у дужки.*

2. Логічні змінні

Логічні змінні. Значення логічних виразів можна надавати логічним змінним. Це скорочує текст програми.

Для **роботи** з логічними змінними є тип даних **boolean.** Нагадаємо, що логічних сталих є лише дві: **true** і **false.**

Логічні змінні треба описувати у розділі оголошення змінних так:

var <список імен змінних>: **boolean;**

Наприклад, **var** z, z1, z2: **boolean.**

Нехай x = 2. Якого значення набуде змінна z2 після виконання таких трьох команд присвоювання:

z := x > 5; z1:= **not** z; z2 := z **or** z1 ?

Відповідь: тут **z=false, a zl=true,** тому змінна z2 матиме значення «істина» **(true).**

Довідка. Логічним змінним не можна надавати значення в діалоговому режимі командою **read,** однак їх можна виводити на екран командою **write.**

Задача. Нехай *a, b, c* — коефіцієнти квадратного рівняння $ax^2+bx+c=0$. Відповісти на запитання: вислів «Квадратне рівняння має два дійсні різні корені» є істинний чи хибний?

Розглянемо програму Lohika.

```
program Lohika;
var a, b, c : real; L : boolean;
begin
    write ('Введіть числа a,b,c: '); readln (a, b, c); L :== b*b-4*a*c> 0;
    writeln ('Квадратне рівняння має два дійсні різні корені — ', L)
    end.
```

Якщо під час виконання програми ввести три числа, наприклад, 2.5 8.1 -2.9, то на екрані отримаємо:

Квадратне рівняння має два дійсні різні корені — TRUE.

Висновок

Таблиця пріоритетів арифметичних і логічних операцій має такий вигляд:

6	( ) — спочатку виконуються дії в дужках
7	Функції, логічна операція **not**
8	*, /, **div, mod, and**
9	+, -, **or**
10	>, <, >=, <=, =, <>

Умовою безпомилкового виконання таких операторів є збігання типів, тобто змінні в лівій частині цих операторів повинні бути описані типом **boolean.**

По-друге, результат обчислення логічних виразів **true** та **false** можна ще трактувати як «так» та «ні». Це наводить на думку про використання логічних виразів для визначення оцінки деякої ситуації, що склалася, і прийняття рішення про те, що робити далі.

Дайте відповіді на запитання

I. Для чого використовують логічні вирази?
2. Що таке простий логічний вираз?
5. Які є символи відношень між величинами?
6. Що таке складений логічний вираз?
5. Які є логічні операції?
6. Дайте означення логічної операції **not.**
7. Дайте означення логічної операції **and.**
8. Дайте означення логічної операції **or.**
9. Який пріоритет логічних операцій?

Розвязати вправи та задачі

1. Усно. Чи істинний простий логічний вираз x > 10, якщо:

а) x=0 (відповідь: ні); б) x=2; в) x=10; г) x=5; д) x=15?

2. Усно. Чи буде хибним вираз x >= 10, якщо:

а) x=1 (відповідь: так); б) x=3; в) x=10; г) x=12; д) x=25?

3. Чи істинний складений логічний вираз (x > 1) **and** (x < 5), якщо:
а) x=0 (відповідь: ні); б) x=2; в) x=10; г) x=5; д) x=15?
4. Чи істинний складений логічний вираз (x <= 8) **and** (x > 3), якщо:
а) x=0 (відповідь: ні); б) x=2; в) x=10; г) x=5; д) x=15?
5. Якого значення **(true** чи **false)** набуде вираз (x <= 2) or (x > 5), якщо:
а) x=0 (відповідь: **true);** б) x=2; в) x=10; г) x=5; д) x=15?
6. Запишіть логічні вирази для нерівностей:
а) 0<x<10 (відповідь: (x>=0) **and** (x<10)); б) -5<x≤8; в) 2<x≤7; г) x≤1 або x>9;
д) x≤2, x>12; є) x≤*0* і *y≥0*.

Домашнє завдання.
Вивчити теоретичний матеріал уроку.

1. Запишіть логічний вираз для визначення, чи точка x належить відрізку:
а) [0; 3) (відповідь: (x>=0) **and** (x<3)); б) [-5; 5); в) [10; 20];
г) [2;14] або [20;25]; д) [4;10] і [8; 12].
2. Запишіть нерівності, які відповідають логічним виразам:
а) (x>2) **and** (x<=20) (відповідь: 2<x<20);
б) (x<-2) **or** (x>15); в) (x>=5) **and** (x<25);
г) (x>3) **and not** (x>5); д) (x>=-5) **and** (x<5) **or** (x>0).

Обчислення значень логічних виразів

10. Для чого використовують логічні вирази?
11. Що таке простий логічний вираз?
12. Які є символи відношень між величинами?
13. Що таке складений логічний вираз?
14. Які логічні операції ви знаєте?
15. Дайте означення логічної операції not.
16. Дайте означення логічної операції and.
17. Дайте означення логічної операції or.
18. Який пріоритет логічних операцій?

Розвязування вправ та задач.

1. Запишіть умову того, що число *a* є: а) парне; б) ділиться без остачі на 3; в) не ділиться без остачі на 3; г) ділиться на 3 і на 5; д) ділиться на 3 або на 5.
2. Запишіть умову того, що деякий день року припадає на 6 травня.
3. Ціну товару позначено змінною c. Яка умова того, що ціна: а) не перевищує 20 грн.; б) є більшою, ніж 10 і меншою, ніж 15 грн.?
4. Складіть логічні вирази для перевірки, чи є точка *(x;y):*
а) У другій чверті (відповідь: (x < 0) **and** (y > 0));
б) на координатних осях;
в) у другій або третій чверті;
г) у квадраті зі стороною, що дорівнює 1, побудованому на координатних осях у першій чверті;
д) у крузі одиничного радіуса з центром у початку координат (підказка: умова належності точки кругу така: $x^2+y^2<1$).
5. Поставити у відповідність розташованим ліворуч виразам вирази, що розташовані праворуч:

l) **not**(x = y),	a) $x \in [0,1]$,
2)(x<y)**or**(x = y),	*b)* $x \neq y$,
5) (x< 0)**or**(x> 1),	*c)*$x \leq y$,
6) (x >= 0) **and** (x <= 1),	d) $x \notin [0, 1]$.

6. Обчислити значення логічних виразів:
5) x < y при *x* = 2.5, *y* = **0.1;**
6) a **and not** (b = c) при *a* = **false,** ***b*** **= false,** *c* **= true;**
7) not (a and b) or b = ***a*** при ***a*** **= true,** ***b*** **= false;**
8) (not a and (x < y)) or (x < 0) при ***x*** **= -0.1,** *y* **= 0.7,** ***a*** **= true.**
7. Записати у вигляді логічних виразів висловлювання, наведені нижче:
3) значення *x* не належить інтервалу (0; 1);
4) значення *x* належить відрізку [- 1; 0] або відрізку [2; 5];
9) точка *M(x; y)* лежить у другій чверті координатної площини;
10) точка *M(x; y)* лежить усередині або на межі одиничного круга з центром у початку координат;
11) точка *M(x; y)* не лежить на одиничному колі з центром у початку координат;
12) 0<A< 1,5;
13) 3 >B>C> 0,1;
14) 5<A<6<B<7,6.

8. Математична логіка - це розділ математики, який вивчає методи встановлення істинності або хибності висловлювань. У логіці визначені такі операції: «і» ($\wedge$) - кон'юнкція, або логічне множення; «або» ($\vee$) - диз'юнкція, або логічне додавання; «не» ($\neg$) - інверсія, або заперечення. Ці логічні операції повністю відповідають логічним операціям **«and»,** «or», **«not»,** що використовуються в логічних виразах. Представити у вигляді логічних виразів такі записи:

$1) x \vee \neg y \wedge (x \vee y) \wedge y = z;$

$2) \neg x \wedge (a * b - c)b + d \le \frac{a(b+d)}{c} \vee x;$

$3) 5 - 2a \ne -2 \wedge x \wedge y \vee \neg((x \vee y) \wedge z);$

$4) 3{,}47\frac{b-c}{a} + b \ge (a - 8{,}96d)^2 \vee a^3 - c < 3b;$

$5) \neg(x \vee y \vee z) \wedge a > \sin c;$

Домашнє завдання

Розвязати задачу: Олексій, Борис і Григорій знайшли в землі амфору. Кожний з них висловив по два припущення:
А: Це амфора грецька, V століття.
Б: Це амфора фінікійська, III століття.
Г: Це амфора не грецька, IV століття.
Вчитель історії сказав хлопцям, що кожний з них висловив правильну думку тільки в одному з двох своїх припущень.
Де й у якім столітті була виготовлена амфора?

Вказівка розгалуження

Розвяжіть задачу:

Віктор, Роман, Леонід та Сергій зайняли на математичній олімпіаді чотири перших місця. Коли їх запитали про розподіл місць, вони дали три відповіді:

- *Сергій – перший, Роман – другий;*
- *Сергій – другий, Віктор – третій;*
- *Леонід – другий, Віктор – четвертий.*

Відомо, що у кожній відповіді тільки одне твердження вірне. Як розподілилися місця?

1. *Оператор умовного переходу. Повна та скорочена форми*

Уявіть собі, що ви за кермом автомобіля і перед вами стоїть вибір дальшого руху: або їхати поганою, але коротшою дорогою, або ж гарною, але довшою. Звичайно, що ваш вибір буде залежати від певних умов: по-перше, чи є у вас зайвий час, по-друге, хто господар автомобіля?

Подібну проблему завжди вирішують оператори розгалуження.

Загальний вигляд повної форми оператора умовного переходу:

if <логічний вираз> **then** P1 **else** P2,

де логічний вираз - це вираз, який може набувати одного з двох значень **true** або **false,** P1 та P2 - це оператори або процедури.

Робота оператора умовного переходу не викликає ніяких труднощів. Цей оператор використовує результат обчислення логічного виразу для вибору того чи іншого шляху наступного виконання алгоритму - виконання оператора P1 або оператора P2. Після цього робота алгоритму продовжується далі за вказаними операторами.

Після аналізу значення логічного виразу буде вибраний лише один з наступних напрямків виконання алгоритму (P1 або P2), після чого цей алгоритм буде виконуватися далі.

Загальний вигляд скороченої форми оператора умовного переходу:

if <логічний вираз> **then** P,

де значення вказаних параметрів такі самі, як і в повній формі.

Відмінність між двома формами умовного оператора: в першій - повній - незалежно від значення логічного виразу якісь дії обов'язково будуть виконані, а вже потім продовжено виконання алгоритму далі, у другій - скороченій - у випадку, коли логічний вираз набуде значення **true,** будуть виконані якісь дії, а потім продовжено виконання алгоритму, а у випадку, коли логічний вираз набуде значення **false,** алгоритм відразу буде продовжено далі.

2. Складений оператор

Розширимо поняття оператора в Pascal. Справа в тому, що ми з вами поки що мали справу лише з *простими* операторами -присвоювання і умовного переходу. А що робити, коли після службових слів **then** або **else** нам потрібно вказати не один такий оператор, а кілька? Для такого випадку в Pascal введене поняття *складеного* оператора.

Складеним оператором називають послідовність кількох операторів або викликів процедур, розділених символом «;» та взятих в операторні дужки **begin** ... **end.**

Складений оператор — це конструкція такого вигляду:

```
begin
    <команда 1>;
```

```
        …
        <команда n>;
    end;
```

Складена команда трактується як одна команда.

Тепер уже час переходити до прикладів. Розглянемо алгоритм пошуку найбільшого з двох заданих чисел A та *B.*

```
program max_A_B;
var a, b, max: real;
begin
write ('Задайте два будь-які числа: ');
readln (a, b);
if a > b then
begin
writeln ('Перше число більше за друге.');
max := a
end
 else
begin
writeln ('Друге число більше або дорівнює першому.');
max := b
end
writeln ('Це число -' , max:10:5);
readln
end.
```

Розглянемо детальніше наведену програму. *По-перше,* у ній чітко спостерігається принцип «вкладеності» операторів, тобто сходинкова структура. Наочність такої програми явно виграє! *По-друге,* перед закриваючою операторного дужкою **(end)** не стоїть символ «;». І справді, ви ж не ставите кому в тексті перед закриваючою дужкою, коли перелічуєте в ньому в дужках декілька слів. Хоча в Pascal це помилкою. Саме через це не поставлено і символ «;» після останньої процедури **readln.**

Спробуйте відповісти на запитання: при виконанні якої умови буде виконано оператор max := b? Дійсно, за умовою, протилежною a > b, тобто при виконанні умови a <= b!

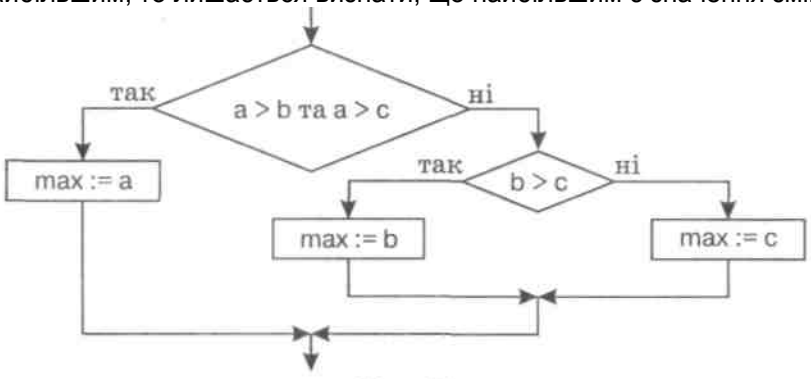
Розглянемо пошук найбільшої з трьох попарно різних величин *a, b, c*

```
program max_A_B_C;
vara, b, c, max: real;
 begin
write ('Задайте три будь-які числа: ');
 readln (a, b, c); if (a > b) and (a > c)
then
begin
writeln ('Перше число більше за інші два.');
 max := a
end
else
if (b > c) then
 begin
writeln ('Друге число більше за інші два.');
max := b
end
else
 begin
writeln (Третє число більше за інші два.');
max := c
end;
writeln ('Це число - ', max:10:5);
readln
end.
```

У цьому прикладі ми бачимо використання двох вкладених умовних операторів.

Внутрішній оператор умовного переходу буде виконано тоді і тільки тоді, коли значення логічного виразу зовнішнього умовного оператора матиме значення **false.** Це означатиме, що значення змінної a не є найбільшим,

тому у вкладеному умовному операторі перевіряється лише значення змінної *b.* Якщо ж і її значення не є найбільшим, то лишається визнати, що найбільшим є значення змінної с. Розглянемо схему цього алгоритму



Спробуємо записати інший варіант алгоритму пошуку найбільшої з трьох величин.

```
program max ABC;
vara, b, c, max: real; begin
write ('Задайте три будь-які числа: ');
readln (a, b, c); if (a > b) and (a > c) then
begin
writeln {'Перше число більше за інші два.');
max := a end;
if (b > a) and (b > c) then
begin
writeln ('Друге число більше за інші два.');
max := b
end;
if (c > a) and (c > b) then
begin
writeln (Третє число більше за інші два.');
 max := c
end;
writeln ('Це число - ', max:10:5);
readln
end.
```

Отже, у першому алгоритмі внутрішній оператор розгалуження виконається тільки тоді, коли нас «пропустить» до нього зовнішній оператор розгалуження. Цей принцип можна порівняти із ситом.

У другому варіанті алгоритму всі оператори розгалуження є послідовними. Тобто всі вони виконуватимуться, перевірятимуть значення своїх логічних виразів і вирішуватимуть, виконувати чи ні вказані в них дії. Крім того, треба зауважити, що використані тут оператори розгалуження мають скорочену форму.

Звичайно, що з погляду ефективності виконання алгоритму перевагу має перший варіант - у ньому може бути виконано менше перевірок для досягнення необхідного результату. Але, зрештою, все залежить від конкретної умови задачі і вашого бачення її реалізації. Десь ви виграєте на етапі виконання програми, але програєте в її написанні, а десь програма буде зрозумілішою, але кількість виконуваних дій у ній буде більшою.

Розглянемо приклад. 1.

Обчислити і вивести значення складеної функції y. $$y = \begin{cases} \ln|x|, & x < -1, \\ \sin(x), & -1 \le x < 1, \\ \cos(x), & x \ge 1. \end{cases}$$

```
program Myfunction;
uses Crt;
var x,y:real;
begin clrscr; writeln('Введіть x'); readln(x);
if x < -1 then y:=ln(abs(x)) else
if (x>=-l) and (x<l) then y:=sin(x)
else y:=cos(x); writeln('x=',x:5:2,'     y=',y:5:2);
readln
end.
```

2. Визначити, чи є задане тризначне число *паліндромом* (паліндром читається однаково ліворуч праворуч і праворуч ліворуч, наприклад, паліндромами є числа 121, 282, слово «наган»). Для розв'язання цієї задачі можна використати просту умову - перша цифра числа повинна рівнятися останній:

```
program palindrom;
var x : Integer;
begin
Write ('Уведіть ціле число:');
readln(x) ;   {Уведення числа x}
if x mod 10 = x div 100  then
Write('Уведене число є паліндромом')
else  Write('Уведене число не
є паліндромом');
end.
```

Запитання для перевірки засвоєння знань

1. *Який алгоритм називається розгалуженим?*
6. *Які умови називаються складеними?*
7. *Які логічні операції ви знаєте?*
8. *Як записують скорочену форму команди розгалуження?*
9. *Які операції відношення можна записувати в умові?*

Домашнє завдання

1. Уведіть вік користувача і його стать(1 - жіноча, 2 - чоловіча). Складіть
2. програму, за допомогою якої красиво привітайтеся з користувачем залежно від його віку й статі.
3. Уведіть дійсне число x. Складіть програму, за допомогою якої визначите, чи є воно коренем рівняння:

а). $15x + 100 = 0$;
б). $105 x - 10,18 = 0$;
в). $12 x^3 - 2x^2 + 6x + 1 = 0$;
г). $17 x^4 - 12x^2 + 2x - 1 = 0$.

4. Задано вік двох приятелів. Складіть програму, за допомогою якої визначите, хто з них старше.
5. Задано вік трьох друзів. Складіть програму, за допомогою якої визначите, хто із друзів молодше.
6. Задано число x. Складіть програму, за допомогою якої збільшить число x на 10, якщо воно від'ємне, у всіх інших випадках зменшить задане число на 5.
7. Задано число x. Складіть програму, за допомогою якої збільшить число x на 15, якщо воно від'ємне.

Повне розгалуження

Задача 1

Умова: Дано значення дійсних величин a, b, c Знайти:

$$\min((a+b+c)/2, \sqrt{1/(a^2+1)+1/(b^2+1)+1/(c^2+1)})$$

```
 Program Example_1;
Uses crt;
Var a,b,c  :  real;
Rez1, Rez2, Min  :  real;  {a,b,c - вхідні дані;  Rez1,  Rez2 -проміжні обчислення; Min - результат
виконання програми}
Begin
Clrscr;  {Очищення екрану}
Write('Введіть числа a,b,c:  ') ;
Readln(a,b,c);
Rez1:=(a + b +  c)  /2;
Rez2:=sqrt(1/(sqr(a)+1)+ 1/ (sqr (b)+1)+1/(sqr(c)+1)) ;
If Rez1<Rez2 Then Min:=Rez1
Else Min:=Rez2;
Writeln('Min=',Min:8:2);
Readkey;  {Затримка зображення на екрані}
End.
```

Задача 2

Умова: *Дано значення дійсної величини x. Визначити:* $\frac{x-5}{x^3+x-2}$

```
Program Example_2;
Uses  crt;
Var X,Rezultat:real;
Begin
Clrscr;     {Очищення екрану}
Write('Введіть значення X: ');
Readln(X);
If  X*X*X+X-2<>0 Then
begin
Rezultat:=(X-5)/(X*X*X+X-2);
Writeln('Rezultat=',Rezultat:8:2);
end
Else
Writeln('Обчислення неможливі - ділення на нуль!');
Readkey;
End.
```

Задача 3

Умова: *При даному значенні x обчислити:* $\sqrt{x^3-\sqrt{x-1}}$

Для розв'язання цієї задачі необхідно пам'ятати, що не можна знайти квадратний корінь з від'ємного числа *(зверніть увагу учнів на те, що у прикладі присутні два квадратних кореня).*

```
Program Example_3;
Uses crt;
Var X, Rezultat: real;
Begin
Clrscr;
Write('Введіть значення X:  ') ;
Readln(X);
If  (X>=1)  and  (X*X*X-sqrt(X-1)>=0)  Then
begin
Rezultat:=sqrt(X*X*X-sqrt(X-1));
Writeln('Rezultat=',Rezultat:8:2);
end
else
Writeln('Обчислення неможливі - від' ємний підкореневий вираз!');
Readkey;
End.
```

Задача 4

За рейтинговою системою оцінка визначається таким чином: якщо загальний бал учня становить не менше 92% від максимального, то виставляється оцінка 12, якщо не нижче 70%, то — оцінка 8, якщо ж не нижче 50%, то — оцінка 5, в інших випадках - оцінка 2, Визначте оцінку учня, якщо він набрав N балів, а максимальне значення загального балу становить S. У цій задачі можна використати повну або скорочену форму команди розгалуження.

Пропонується розв'язування зі скороченою формою.

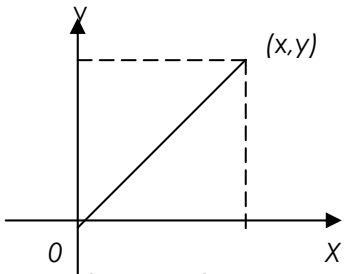
```
Program Example_4;
Uses crt;
Var N,S,Grade:integer;
{N - бали, що набрав учень; S - максимальне значення сумарного балу; Grade - оцінка учня}
Begin
Clrscr ;
Write('Введіть максимальне значення сумарного балу, що може набрати учень: ');
Readln(S);
Write('Введіть кількість балів, що отримав учень: ');
Readln(N);
If (S<=0) or (N<=0) or (N>S)
Then writeln('Помилка вхідних даних')
```

```
Else
Begin
N:=round(N/S*100); {Знаходження %-відношення балів учня до максимально можливого}
If N>=92 then Grade:=12;
If (N<92) and (N>=70) then Grade:=8;
If (N<70) and (N>=50) then Grade:=5;
If (N<50) then Grade:=2;
Writeln(‘ Учень отримав оцінку - ' , Grade);
End;
Readkey;
End.
```

Задача 5

Умова: На площині дано дві точки (x_v y_t) та (x_2, y_2). Визначити, яка з них знаходиться далі від початку координат.

Для розв'язання цієї задачі необхідно скористатися теоремою Піфагора для знаходження відстані від початку координат до заданої точки (див. рисунок):



Очевидно, що відстань від початку координат до точки з координатами *(x,y)* буде обчислюватись наступним співвідношенням:

$$S = \sqrt{x^2 + y^2}$$

Зверніть увагу: через те, що кожна з координат у формулі підноситься до квадрату, неважливо, в якій чверті координатної площини буде знаходитись точка. Програма для розв'язання даної задачі має вигляд:

```
Program Example_5;
Uses  crt;
Var X1, Y1, X2, Y2:real;
{X1,Y1, X2, Y2 - координати даних точок}
S1, S2:real;
{S1, S2 - відстані відповідно до першої та другої точки}
Begin
Clrscr;
Write('Введіть координати першої точки: ');
Readln(X1,Y1);
Write('Введіть координати другої точки: ');
Readln(X2,Y2);
S1:=sqrt(sqr(X1)+sqr(Y1));
S2:=eqrt(sqr(X2)+sqr(Y2)) ;
If  S1<S2
Then Writeln('Друга точка далі від початку координат')
Else Writeln('Перша точка далі від початху координат') ;
Readkey;
End.
```

Запитання для перевірки засвоєння знань

6. Опишіть правила оформлення складних виразів.
7. Назвіть пріоритети виконання операцій в програмах?
8. Що таке вкладеність операторів?
9. У яких випадках використовують логічні оператори or, а у яких and?
10.Поясніть відмінність у застосуванні команд Write і Writeln.

Домашнє завдання

5. Задано число *p*. Складіть програму, за допомогою якої, якщо *p* менше нуля, значення *y* обчислите по формулі $y = p^3$. Якщо *p* більше або дорівнює нулю, то значення *y* обчисліть по формулі *y* = 1 - *p*.
6. Відомі: денний заробіток двірника й заробіток садівника за тиждень. Складіть програму, за допомогою якої визначите, хто з них більше заробляє в рік.
7. Відомо, скільки в середньому за день заробляє інженер і скільки в середньому за місяць заробляє лікар. Складіть програму, за допомогою якої визначите, хто з них більше заробляє грошей за рік.
8. Задано дійсне число x. Складіть програму, за допомогою якої перевірте виконання нерівності **3/4 < X < 7/8.**

Розвязування задач на використання вказівки розгалуження

Задача 86 (задачник Т.П.Караванової)

Чебурашка вирішив купити килими, щоб застелити кімнату, в якій він мешкав разом з Геною. Їхня прямокутна кімната виявилася розмірами A x *B, де A та B — цілі числа. Коли Чебурашка запитав у магазині, які килими є у продажу, то продавець повідомив, що є квадратні килими зі стороною C, де C — ціле число. Яку кількість килимів потрібно придбати Чебурашці, щоб накрити максимальну площу кімнати? Килими неможна накладати та підгинати. Визначити, яка площа кімнати буде не накритою килимами. Передбачити ситуацію, коли розміри килиму перевищують розміри кімнати.*

Очевидно, що якщо довжина сторони килиму більша за будь-яку зі сторін кімнати, то застелити її цими килимами неможливо. Крім того, для знаходження кількості килимів, що вміщуються по одній зі сторін кімнати без їх підгинання, необхідно поділити націло довжину кімнати на довжину килиму. Загальна кількість килимів знаходиться за формулою: $K=K_1 \cdot K_2$, де K_1 та K_2 — кількості килимів, що вміщуються вздовж двох суміжних сторін кімнати.

Площа, що не закрита килимами, визначається як різниця між площею кімнати та площею всіх куплених килимів,

Використані змінні: A, *B* — розміри кімнати; C—розмір килиму; K_1, K_2 - кількість килимів вздовж однієї та другої стінки відповідно; K - загальна кількість килимів; S —площа кімнати, що не накрита килимами.

Програма, що реалізує алгоритм розв'язання даної задачі, має вигляд:

```
Program Example_86;
Uses ort;
Var  a,b,c,S:word; K,K1,K2   :   word;
Begin
  Clrscr;   {Очищення екрану}
  Write ( 'Введіть розміри кімнати:   ') ;,
  Readln(a,b);
  Write('Введіть розміри килима: ');
  Readln(c);
  If   (c > a)   or   (c > b)
  Then writeln( 'Кімнату неможливо накрити такими килимами')
  Else
  Begin
  K1:=a div c;  K2:=b div c;
  K   := K1*K2;       S   := a*b - K*c*c;
  Writeln( 'Кількість  куплених килимів   ',   K);
  WriteIn('Площа кімнати,   що не накрита килимами  ',   S);
   End; Readkey;
   End.
```

Задача 89

Від річкового вокзалу відійшли одночасно у протилежних напрямках теплохід та турист. Теплохід рухався зі швидкістю V_1 км/год, а турист по стежці вздовж річки зі швидкістю V_2 км/год. Якщо через N годин турист передумає і вирішить попливти річкою назад за теплоходом зі швидкістю V_3 км/год, то чи встигне він підсісти на теплохід, який має за графіком зупинку через Y годин після початку руху і стоїть на цій зупинці Z годин? Вважати на те, що всі події відбувалися протягом однієї доби.

Якщо турист протягом N годин рухався в протилежному напрямку від теплоходу, то відстань між ними в той момент, коли турист вирішив наздогнати теплохід, була наступна:

$S=(V_1 + V_2) \cdot N$ де V_1 та V_2 — швидкості теплоходу та туриста відповідно.

Швидкість, з якою турист почне наздоганяти теплохід, — $(V_3 – V_1)$ км за годину, де V_3 — швидкість, з якою турист попливе навздогін теплохода. Час, який буде у туриста для наздоганяння, $(Y-N + Z)$ годин, тому що зупинка

в теплохода буде за розкладом через У годин після початку руху, але *N* годин він уже плив, а Z годин теплохід буде стояти на цій зупинці. Тоді за цей час турист пройде відстань: $S_t=(V_3 – V_1)\cdot(Y – N + Z)$

Отже, турист встигне підсісти на теплохід тільки в тому випадку, якщо відстань S_t буде не менше, ніж відстань, на яку теплохід перегнав туриста. Програма, що реалізує запропонований алгоритм, має вигляд:

```
Program Example_89;
 Uses crt;
Var V1,V2,V3:real;  N,Y,Z   :   real;
Begin Clrscr;
Write('Введіть швидкості теплоходу та туриста:   ') ; Readln (VI, V2) ;
Write ('Введіть час, черев який турист підсів на теплохід:')
 Readln(N);
Write('Введіть швидкість,   з якою турист плив за теплоходом, час зупинки теплоходу, та тривалість
зупинки:')
Readln(V3,Y,Z);
If  (V1<=0)or(V2<=0)or(V3<=0)or(N<=0)or(Y<=0)or(Z<=0) Then
 writeln('Помилкові вхідні дані')
 Else
Begin
S:=(V1+V2)*N;
St:=(V3-V1)*(Y-N+Z);
If  St >= S
Then writeln('Турист встигне на теплохід.')
Else
writeln('Турист не встигне на теплохід.');
End; Readkey;
End.
```

Задача № 90

Умова: *Жили собі дід і баба, і був у них город прямокутної форми. Довжина городу була А м, а ширина складала В м. Якось дід посварився з бабою і вирішив поділити город порівну. Тепер у діда квадратний город зі стороною С м, відрізаний скраю, а решта дісталася бабі. Визначити, чи не залишилася баба ошуканою та якої форми дістався їй город - прямокутної чи квадратної?*

Взагалі задача має дуже простий розв'язок: адже бабуся не буде ошуканою в тому випадку, якщо площа городу, що залишилася для неї, не буде меншою, ніж площа дідусевого городу, тобто $C^2<=A\cdot B - C^2$. Та це тільки на перший погляд. Насправді в даній задачі може бути велика кількість винятків.

Наприклад, якщо дідусь захоче відрізати собі город зі стороною більшою, ніж сторона всього городу, то це неможливо зробити взагалі. Якщо ж він відріже, то город, що залишиться, може мати квадратну, прямокутну або іншу форми.

Програма, що реалізує запропонований алгоритм, має вигляд:

```
Program Example_90;
  Uses crt;
  Var A,B,C:real;
  Begin  Clrscr;
  Write('Введіть розміри городу:   ');  Readln(A,B);
  Write('Введіть довжину сторони дідусевого городу: ');  Readln(C);
  If (A<=0)or(B<=0)or(C<=0) then writeln ('Помилкові вхідні дані')
  Else
  Begin
  If (C>A) or (C>B)
  then writeln('Дідусь не зможе відрізати город такого розміру')
  else
  begin
  If A*B-sqr(C)<=sqr(C) then writeln('Бабуся ошукана.')
  else writeln('Бабуся не ошукана.');
  If (A<>C) and (B<>C)
  Then writeln('Город залишився іншої форми')
  Else
  If ((A=C)and(B/2=C))or((B=C)and(A/2=C))
   then writeln('У бабусі квадратний город.')
  else writeln('У бабусі прямокутний город.') ;
```

```
end;
 End;
 Readkey; End.
```

Задача №91

Умова Трьом Товстунам подали на десерт кремові тістечка. Маса одного тістечка — Хкг, а маса Товстунів відповідно X_1 кг, X_2 кг та X_3 кг. Перший Товстун з'їв N тістечок. Кожний наступний Товстун з'їдав у два рази більше від попереднього, але при цьому він не міг з'їсти більше половини своєї власної ваги. Скільки тістечок було з'їдено Товстунами за обідом?

Зверніть увагу на те, що другий та третій Товстуни за умовою можуть з'їсти тістечок у два рази більше, ніж попередній Товстун, але не можуть з'їсти більше половини своєї ваги. Тому фактично в задачі необхідно перевірити, чи не перевищує кількість тістечок, що може з'їсти кожний Товстун, дозволену масу, і у відповідності до цього підрахувати кількість тістечок, що були з'їдені.

Наприклад, якщо другий Товстун може з'їсти 2 • N тістечок, то вага цієї їжі буде *2·N·X кг.* Але за умовою він не може з'їсти більше половини своєї ваги, тобто більше ніж $X_1/2$ кг. Тому якщо вага тих тістечок, що Товстун може з'їсти, не перевищує поріг $X_1/2$ кг, то ми до загальної кількості тістечок додаємо всі можливі, тобто *2·N,* якщо ж перевищує, то ми додаємо тільки ту кількість тістечок, що не дає змоги перевищити припустимий поріг, тобто $X_1/2/X$ (дозволена вага їжі поділена на вагу одного тістечка). Якщо в цьому випадку число вийде нецілим, то це означає, що Товстун з'їв тістечко не повністю. Щоб такого не трапилось, ми робимо відкидання дробової частини після ділення за допомогою функції *trunc.*

Програма, що реалізує цей алгоритм, має вигляд:

```
Program Example_91;
Uses crt;
Var X,X1,X2,X3:real; N,Counter : integer;
{N - кількість тістечок, що з'їв перший Товстун; Counter -загальна кількість з'їдених тістечок}
Begin
Clrscr;
Write('Введіть вагу тістечха:   ');
Readln(X);
Write('Введіть вагу Товстунів   (1-го,   2-го та 3-го):   ');
Readln(X1,   X2,   X3);
Write('Введіть кількість тістечок, що з'їв перший Товстун ') ;
Readln(N);
If   (X<=0)or(X1<=0)or(X2<=0)or(X3<=0)or(N<=0) Then
 writeln('Помилкові вхідні дані')
Else
 Begin
Counter:=N;   {З'їв перший Товстун)
If N*2*X<=X2/2     Then Counter:=Counter+2*N
Else
 Counter := Counters- trunc(X2/2/X);
If N*4*X<=X3/2    Then Counter:=Counter+4*N
Else
Counter:= Counter* trunc(X3/2/X);
Writeln('Кількість з'їдених тістечок:   ' , Counter);
End;
Readkey;
End.
```

Запитання для перевірки засвоєння знань

6. *Що таке глобальний блок програми?*
7. *Поясніть особливості використання локальних змінних.*
8. *Які операції можна виконувати зі змінними цілого типу?*
9. *Для чого в програмах використовують коментарі?*
10. *Поясніть використання функції* ***trunc.***

Домашнє завдання:

3. Задано довжини сторін трикутника. Складіть програму, за допомогою якої визначіть, чи є цей трикутник рівнобедреним, рівностороннім чи різностороннім.
4. Задано три числа a, b, c. складіть програму, за допомогою якої визначіть, чи існує трикутник з такими довжинами сторін. Якщо да, то знайдіть його периметр.

27. Розвязання задач на використання вказівки розгалуження

Розв'язування задач

Задача 120 (задачник Т.П.Караванової)

Дано натуральне число N (N k 1000). *Визначити суму першої і останньої цифр даного числа.*

Для розв'язання цієї задачі ми скористаємося стандартними операціями цілочисельного ділення та остачі від ділення цілих чисел (операції div та mod). Нагадаємо, що результатом ділення числа націло на 10 буде ефект відкидання «молодшої» цифри числа (відповідно при діленні на числа 100,

1000,10000 тощо будемо «відкидати» дві, три або чотири цифри числа). Результатом ж операції знаходження залишку від ділення на 10 буде остання цифра числа (відповідно при знаходженні залишку від ділення на 100, 1000, 10000 будемо отримувати дві останні, три останні, чотири останні цифри числа).

Наприклад:

234 div 10 = 23

9213 div 100 = 92

52 mod 10 = 2

2845 mod 1000 = 845.

Виходячи з усього сказаного, програма буде мати вигляд:

```
Program Example_120_2;
Uses crt;
Var N,  First,  Last  :  word;
{First - перша цифра числа;  Last - остання цифра числа)
Begin
Clrscr;
Write('Введіть число: ');
Readln(N);
Last := N mod 10;
If (N>=0) and (N<10) then First:=0;
If (N>=10) and (N<100) then First:=N div 10;
If (N>=100) and (N<1000) then First:=N div 100;
If (N=1000) then First:=1;
Writeln ('Сума першої та останньої цифр дорівнює' , First+Last) ; Readkey;
End.
```

Задача.

Популярність діалогових програм, що дозволяють зробити вибір із запропонованого користувачу «меню», всім відома. Розглянемо задачу, яка за генератором випадкових чисел визначає виграш в лотереї таким чином: 1 - відеомагнітофон; 2 -музичний центр; 3 - комп'ютер; 4 - комп'ютерний клас.

Основою даного алгоритму є перевірка значення цілої змінної, наприклад, ch, яке визначає порядковий номер виграшу. Наведемо словесний вигляд алгоритму:

- Якщо значення змінної ch дорівнює 1, то виграно перший за номером приз і переходимо до п.5. У протилежному випадку переходимо до п.2.
- Якщо значення змінної ch дорівнює 2, то виграно другий за номером приз і переходимо до п.5. У протилежному випадку переходимо до п.3.
- Якщо значення змінної ch дорівнює 3, то виграно третій за номером приз і переходимо до п.5. У протилежному випадку переходимо до п.4.
- Якщо значення змінної **ch** дорівнює 4, то виграно четвертий за номером приз і переходимо до п.5.
- Кінець алгоритму.

```
program prize;
uses CRT;  var n: byte;
ch: char;
begin
writeln ('Призи лотереї:');
writeln ('1. Відеомагнітофон');
writeln ('2. Музичний центр');
writeln ('3. Комп'ютер');
 writeln ('4. Комп'ютерний клас');
writeln ('Натисніть будь-яку клавішу для запуску лототрона');
repeat until KeyPressed;
ch:=ReadKey; randomize; n:=random(3)+1;
```

```
write ('Ваш виграш під номером ', n,'. Це - ');
if n=l then writeln ('Відеомагнітофон')
else if n=2 then writeln ('Музичний центр')
else if n=3 then writeln ('Комп'ютер')
else writeln ('Комп'ютерний клас');
repeat until KeyPressed
end.
```

Прокоментуємо деякі моменти програми. Послідовність викликів стандартних процедури і функції randomize; n:=random(3)+1 дозволяють за допомогою генератора випадкових чисел змінній n надати будь-якого випадкового значення в інтервалі від 1 до 4. Підключення модуля **CRT** дозволяє використати функції **KeyPressed** та **ReadKey.** За допомогою першої ми можемо затримати виконання програми до натискання користувачем будь-якої клавіші на клавіатурі. Але, оскільки код цієї клавіші буде залишатися в буфері клавіатури і це стане на заваді повторного використання цієї ж функції в кінці програми, то операція **ch:=ReadKey** допоможе «вичитати» цей код із буфера і спорожнити його.

Задача 130

Дано дійсні додатні числа a, b, c, x, y. Визначити, чи пройде цеглина з ребрами a, b, c у прямокутний отвір зі сторонами x та y. Проштовхувати цеглину дозволяється лише так, щоб кожне з її ребер було паралельним чи перпендикулярним кожній зі сторін отвору.

Для розв'язання цієї задачі пропонується впорядкувати розміри отвору та розміри цеглини за зростанням, тобто досягти того, щоб було *a<=b<=c* та x <= y. Тоді перевірка зведеться до порівняння розмірів отвору з найменшими розмірами цеглини (адже ми можемо цеглину розвернути будь-яким боком, щоб проштовхнути її у отвір).

```
Program Example_130;
Uses crt;
Var a,b,c,x,y,S:real;
{S - допоміжна змінна для обміну місцями значень двох змінних}
Begin Clrscr;
Write ( 'Введіть розміри цеглини: ') ; Readln (a,b,c) ;
Write (‘ Введіть розміри отвору: ') ; Readln (x, y) ;
If (a<=0)or(b<=0)or(c<=0)or(x<=0)or(y<=0)
Then writeln ('Помилка вхідних даних.')
Else
Begin {Впорядкування розмірів цеглини)
If a>b  Then Begin S:=a; a:=b; b:=S; End;
If a>c  Then Begin S:=a; a:=c; c:=S; End;
If b>c  Then Begin S:=b; b:=c; c:=S; End;
{Впорядкування розмірів отвору}
If x>y  Then Begin S:=x; x:=y; y:=S; End;
If (a<=x) and (b<=y) Then writeln( 'Цеглина пройде у отвір.')
else writeln('Цеглина не пройде у отвір.')
End;
Readkey;
End.
```

Запитання для перевірки засвоєння знань

1. *Чим відрізняється застосування функцій **Read** і **Readln.***
6. *Як можна перетворити змінну дійсного типу у змінну цілого типу?*
7. *Над якими типами даних можна виконати операцію div?*
8. *Чим відрізняються операції **div** і mod?*
9. *Коли використовують підпрограми?*
10. *Що називають логічними виразами? На які два типи вони поділяються?*
11. *Які знаки співвідношень використовуються для запису простих логічних виразів?*
12. *Назвіть логічні операції.*
13. *Поясність використання логічних операцій за таблицями істинності.*
14. *Що називають складеним оператором? Які правила його запису?*
15. *Яким чином організоване розгалуження у Паскалі?*
16. *Чим відрізняються повна та скорочена форми оператора умовного переходу?*
17. *Запишіть загальний вигляд повної форми розгалуження.*
18. *Запишіть загальний вигляд скороченої форми розгалуження.*
19. *Намалюйте схеми алгоритмів обох варіантів розгалуження.*

Домашнє завдання:

Складіть програму, за допомогою якої обчисліть значення функції:

$$a)\ y=\begin{cases}1+3x^2, & x<-2;\\ 1-2x^3, & x\ge -2;\end{cases} \quad b)\ y=\begin{cases}x^2+2, & x\le 1;\\ x^2-2x+4, & x>1;\end{cases}$$

$$c)\ y=\begin{cases}\dfrac{4x^2-15}{2x^3-5}, & x\ge 1;\\ \sqrt{2x-5}, & 0\le x<1;\\ |x^3-16|, & x<0;\end{cases} \quad d)\ y=\begin{cases}\sqrt{x^2+5}, & x>0;\\ |2x^3-2x^2|, & -10<x\le 0;\\ \dfrac{5x-1}{x^4+11}; & x\le -10.\end{cases}$$

Оператор вибору

1. Вказівка вибору

Вказівка вибору це оператор який є узагальненням оператора *if* і дає змогу зробити вибір із довільного числа наявних варіантів. Він складається з виразу, що називається *селектором,* і списку параметрів, кожному з яких передує список констант вибору (список може складатися і з однієї константи). Як і в операторі *if,* тут може бути присутнім слово *else,* що має той же зміст. Формат опису:

case < вираз-селектор > of
< список констант вибору1 > : < оператор 1 >;
< список констант вибору 2 > : < оператор 2 >;
< список констант вибору п > : < оператор п >
[else < оператор п+1 >] end;

Оператор *case* працює наступним чином: спочатку обчислюється значення виразу-селектора, потім забезпечується реалізація того оператора, константа вибору якого дорівнює поточному значенню селектора. Якщо жодна з констант не дорівнює поточному значенню селектора, виконується оператор, що знаходиться за словом *else.* Якщо слово *else* відсутнє, активізується оператор, що знаходиться за словом *end,* тобто перший оператор за межею дії *case.* Селектор має належати до одного з перелічувальних типів (цілого, булівського або літерного). Дійсні та рядкові типи використовувати в якості селектора заборонено. Список констант вибору складається з довільної кількості значень або діапазонів, відділених один від одного комами. Межі діапазону записуються двома константами через складений символ діапазону «..». Тип констант у будь-якому випадку повинен збігатися з типом селектора. Щоб краще зрозуміти використання оператора вибору, розглянемо кілька типових задач.

Задача 134

Розробити діалогову програму, яка запитує вік користувача і визначає, до якої вікової категорії він належить:

6) від 1 до 10 років - дитина;
7) від 11 до 15 років - підліток;
8) від 16 до 20 років - юнак (юнка);
9) від 21 до ЗО років - молода людина;
10)після 31 року - доросла людина.

Особливих пояснень ця задача не потребує, адже її можна розв'язати і за допомогою команди розгалуження. Однак зробимо її за допомогою команди вибору, причому, щоб скористатися гілкою *Else,* будемо вважати,що людина може мати вік не більше 150 років (не зафіксовано рекордів коли людина прожила понад 150 років). Якщо ж користувач введе число, що не входить у дозволений діапазон, будемо вважати, що він пожартував.

```
Program Example_134 ;
Uses crt;
Var Years:byte;  {Years  - вік користувача}
Begin
Clrscr;  {Очищення екрану}
Write('Введіть Ваш вік:  ');
Readln(Years);
Write('Ви ');
Case Years of
0..10:  Writeln (' - дитина.') ;
11..15:  Writeln('- підліток.');
16..20:  Writeln('- юнак  (юнка).');
21..3O:  Writeln('- молода людина.') ;
31..150: Writeln('- доросла людина.')
Else writeln(', пожартували? Людина стільки не живе!') ;
End;
Readkey;  {Затримка зображення на екрані}
```

```
End.
```

Задача№151

Розробити програму виведення інформації про день тижня, (вихідний чи робочий), якщо задано його номер від 1 до 7 (1 - понеділок).

```
Program Example_151;
Uses  crt;
Var Day:byte;   {Day - номер дня тижня}
Begin
Clrscr;
Write('Введіть номер дня тижня:   ') ;
Readln(Day);
Case Day of
1..5:   Write('Це робочий день   ');
6,7:    Write('Це вихідний день ')
Else     write('Це не день  ');
End;
Writeln('тижня.');
Readkey;
End.
```

Розглянемо ще один приклад застосування команди вибору.

Нехай населені пункти позначені номерами від 1 до 8. Вартість одного квитка до конкретного пункту *k* визначається так:

$$Cina = \begin{cases} 22, & k = 1, \\ 25, & k = 2,3,4, \\ 30, & k = 5,6, \\ 35, & k = 7,8. \end{cases}$$

Скільки коштуватимуть *m* квитків до населеного пункту, номер якого вводять з клавіатури?

```
program Kvytky; uses Crt;
var k,m,cina:integer;
begin
clrscr;
writeln('Введіть номер пункту та кількість квитків:');
readln(k.m);
case k и
1          : cina:=22;
2..4       : cina:=25;
5,6        : cina:=30
else         cina:=35
end;
write(m,  ' квитків до пункту ', k,  ' коштують ');
writeln(m*cina);
readln
end.
```

Якщо під час виконання програми ввести дані так: 3 5, то на екрані отримаємо: 5 квитків до пунку 3 коштують 125.

Дайте відповідь на запитання:

6. Чим константи відрізняються від змінних?
7. Який тип даних в програмах використовують найчастіше?
8. Коли можна використовувати діапазон даних?
9. Яке призначення алгоритмічної конструкції вибір?
10. Який вигляд має команда case і як вона працює?

Виконайте вправи:

4. Запишіть у вигляді діапазону: а) 2, 3, 4, 5, 6; б) 121, 122, 123, 124; в) ‘a’, ‘b’, ‘c’; г) ‘a’, ‘б’,’в’,’г’.
5. Запишіть діапазон у вигляді списку: а) 10..15; б) -5..2; в) ‘k’..’m’.
6. Для заданого номера дня тижня вивести його назву.

Домашнє завдання:

Вивчити теоретичний матеріал уроку.

Задайте відстані до міст A, B, C, D. Нехай на 100 км потрібно 7 л бензину. Комп'ютер запитує водія про пункт призначення (треба буде ввести одну букву) і повідомляє про небхідну кількість бензину.

3. Напишіть інструкцію CASE, яка виводить на екран назву пори року. Змінна month містить номер місяця.
4. Замініть інструкцію

```
if day=7
then write('Неділя') else
if day=6
then write('Субота')
else      write('Робочий день');
```

інструкцією CASE. Відповіді:

1.
```
case month of
  12,1,2:write ('Зима! ' ) ; 3..5:write('Весна!');
  6. .8:write('Літо! ') ;
  9. .11:write('Осінь!') ; end;
```

2.
```
case day of
  7:write('Неділя');
  6:write('Субота');
  else write('Робочий день');
  end;
```

Розв'язування задач.

Задача1. Ціна купальника у травні становить *a* грн. У червні ціна буде збільшена на *p%,* у липні дорівнюватиме травневій, а в серпні буде зменшена на *q%* порівняно з травневою. Скласти програму, яка після введення номера літнього місяця (month) визначатиме ціну (c) купальника.

Розв'язок задачі запишемо у такому вигляді:

$$c = \begin{cases} a(1+p/100), & якщо \quad month = 6, \\ a, & якщо \quad month = 7, \\ a(1-q/100), & якщо \quad month = 8. \end{cases}$$

Розглянемо програму. Змінну month оголосимо не як integer, а як діапазон 6..8 — звуження типу integer згідно з умовою задачі.

```
program Shopping;
var month : 6..8; a,p,q,c : real; when:string;
begin
write('Введіть ціну, відсотки p та q, номер місяця: ');
readln(a,p,q,month);
case month of
6: begin c := a * (1 + p/100); when :='червні" end;
7: begin c := a; when := 'липні' end;
8: begin c := a * (1 - q/100); when := 'серпні' end end;
writeln('Купальник у  ', when, ' коштує   ', c)
end.
```

Виконаємо програму. Нехай нас цікавить ціна купальника в серпні (month=8). Введемо дані так: 9.8 20 30 8. Отримаємо результат: Купальник у серпні коштує 6.86.

Задача 2. Виконайте програму Dialog. Уведіть число — кількість років — і прочитайте відповідне повідомлення.

```
program Dialog; var Vik : integer; begin
write ('Скільки Вам років? ');
read(Vik);
case Vik of
12,13:writeln('Вам ще рано читати цей розділ');
14,15:writeln('Вам ще не можна дивитися фільми для
дорослих');
16,17:writeln('Добре вчіться - батьки будуть пишатися Вами'); 18,19:writeln('Мінздоров'я
попереджає...');
20..23:writeln('Паскаль вивчати вже пізно - пора заміж!')
else
```

```
writeln('Закрийте цю книжку! Читайте Н.Вірта!');
end ; readln
end.
```

У програмі Dialog використано діапазон значень 20..23 замість списку чисел 20, 21, 22, 23.

Завдання. Поекспериментуйте з однією з наведених вище програм, модифікувавши її на свій розсуд та ввівши свої дані.

Спробуйте розв'язати самостійно:

4. Дано ціле число *n* (1 < *n* < 12), яке визначає порядковий номер місяця в році. За введеним значенням *n* надрукувати назву відповідного місяця.
5. Розробити програму виведення кількості днів у місяці, якщо останній задається цілим числом від 1 до 12.
6. Розробити програму видачі текстового варіанта шкільних оцінок:

5) 1, 2, 3 - початковий рівень;
6) 4, 5, 6 - середній рівень;
7) 7, 8, 9 - достатній рівень;
8) 10, 11, 12 - високий рівень.

Домашнє завдання:

4. Дано ціле число *n* (1 < *n* < 4), яке визначає порядковий номер кварталу року (січень, лютий, березень -1 квартал і т. д.). За вказаним значенням *n* надрукувати перелік місяців, які відносяться до цього кварталу.
5. За даним цілим значенням змінної *k* (1 < *k* < 6), яка визначає день тижня, надрукувати свій розклад уроків.
6. Дано ціле число га (1 < n < 4), яке визначає пору року. За вказаним значенням *n* надрукувати перелік місяців, які відносяться до цієї пори року.

Задача 165

Дано натуральне число N (N < 100), яке позначає вік людини. Додати до цього числа відповідно слова: «рік», «роки», «років», наприклад: 1 рік, 12 років, але 43 роки.

Очевидно, що для того, щоб правильно дописати відповідне слово, необхідно виділити останню цифру числа, що позначає вік людини. Тоді, якщо це цифра «1», то дописується слово «рік», якщо цифри «2», «3» або «4» - дописується слово «роки», а в усіх останніх випадках - дописується слово «років». Виключенням являється діапазон між 10 та 20 роками: в цих випадках завжди пишеться слово «років».

```
Program Example_165;
Uses crt;
Var Years:byte;  {Years - вік людини}
Begin Clrscr; Write('Введіть Ваш вік:  ') ;
Readln(Years);
If Years>100 Then writeln('Помилкові вхідні дані.')
Else
Begin
Write('Вам ',Years);
If (Years>=10) and (Years<=20) Then writeln('років')
Else
Case Years mod'10 of
1: writeln('рік.');
2..4 : writeln('роки.');
0,5..9: writeln('років.');
End;
End;
Readkey;
End.
```

Розв'яжіть самостійно:

1. Введемо такі позначення для відмінків в українській мові: «називний» - «н» або «Н»;

«родовий» - «р» або «Р»; «давальний» - «д» або «Д»; «знахідний» - «з» або «З»; «орудний» - «о» або «О»; «місцевий» - «м» або «М»; «кличний» -«к» або «К».

Розробити програму, яка за введеним позначенням відмінка видаватиме запитання, на які відповідає іменник у вказаному відмінку, наприклад:

«називний» - «хто?, що?».

2. Використовуючи позначення відмінків із попередньої задачі, розробити програму, яка за введеним позначенням відмінка видаватиме запитання, на яке відповідає прикметник у вказаному відмінку, наприклад:
«називний» - «який?».
3. Введемо позначення для визначення родів в українській мові:
чоловічий рід - «ч»;
жіночий рід - «ж»;
середній рід - «с». Розробити програму, яка за введеним позначенням роду видаватиме закінчення відповідних йому слів у називному відмінку, наприклад: «жіночий рід» - «-а, -я».

Домашнє завдання:
Розробити програму-довідник, яка за введеним значенням радіуса *R* пропонуватиме користувачу послуги в обчисленні:
1)1- довжини кола;
5) 2 - площі круга;
6) 3 - об'єму кулі;
7) 4 - площі поверхні кулі.

Оператор безумовного переходу. Мітки

6. Оператор безумовного переходу.
Для визначення поняття оператора безумовного переходу ознайомимося з поняттям мітки.
Міткою називається описаний в розділі міток* Label *ідентифікатор або беззнакове число від 0 до 9999.
Мітками помічаються виконувані у програмі дії або службові слова **end.** Після кожної мітки повинен стояти символ «:». Кожна мітка у програмі повина бути унікальною. До речі, мітки 0010 та 10 вважаються однаковими!
Наведемо приклади міток:
Label StopLabel, 10, 909, Label_1, Label_2.
Тепер розглянемо **загальний вигляд оператора безумовного переходу:**
goto <мітка>.
Використання оператора безумовного переходу можна продемонструвати на прикладі захисту програми від уведення недопустимих даних.
Нехай за умовою задачі треба задавати лише додатні значення змінних *x, y, z.* Фрагмент програми виглядатиме приблизно так:

```
……
label myjabel;
 var x, y, z: real;
……..
begin
myjabel: write (' Задайте три додатні числа:');
readln (x, y, z);
if (x <= 0) or (y <= 0) or (z <= 0) then begin
writeln (' Ви помилилися, повторіть: ') goto myjabel;
end; { продовження програми}
```

У цьому прикладі використано можливість запису в програмі своїх власних коментарів. Для цього використовуються фігурні дужки «{...}», тобто все те, що взято у фігурні дужки, у Pascal-програмі не обробляється, ігнорується. Цією можливістю буде зручно користуватися згодом під час налагодження програм для тимчасового відключення виконання деяких її фрагментів.

2. Порожній оператор.

Загальний вигляд порожнього оператора: ;.
Тобто якщо ви в програмі випадково поставите підряд «;;», то це компілятором Pascal не буде вважатися помилкою, оскільки між цими двома символами він розпізнає порожній оператор! Але хіба лише для цього потрібен порожній оператор? Інколи виникає необхідність передати керування на неви-конувану частину програми, якою є, наприклад, службове слово **end.** А оскільки мітку можна ставити лише на виконуваній частині, то для цього існує порожній оператор: **label** myjabel;

```
begin
{початок програми }
goto my_label;
{ продовження програми }
My_label: ;
end.
```

З*адача* 1. Обчислити периметр (p) і площу трикутника (s) за трьома відомими сторонами a, b, c Програма повинна перевіряти правильність вхідних даних, тобто, чи існує трикутник.

```
program Trykutnyk;
label 222;
var a, b, c, p, s, piv, z : real;
begin
222 : write ('Введіть значення сторін:');
readln (a,b,c);
p := a+b+c; piv := p/2;
z := piv*(piv-a)*(piv-b)*(piv-c);
{z, piv - додаткові змінні, тут використано формулу Герона}
 if z>0 then
begin
s := sqrt(z);
writeln ('Периметр=', p:8:2, ' Площа=', s:8:2)
end;
if z<=0 then
begin
writeln ('Трикутник не існує. Введіть інші дані');
 goto 222
end; readln
end.
```

Виконаємо програму. На запит комп'ютера введемо такі дані: 18, 25, 6 (дані треба вводити через пропуск: 18 25 6). Отримаємо: Трикутник не існує. Введіть інші дані Введіть значення сторін: 18 25 10 Периметр= 53.00 Площа= 74.67

Завдання. Поекспериментуйте з програмою, ввівши дані так: 3 5 7. *Зауваження.* Концепція сучасного (структурного) програмування не рекомендує використовувати команду переходу **goto** в програмах. Цю команду можна замінти іншими алгоритмічними конструкціями, наприклад, командою while, яку вивчатимемо пізніше.

Дайте відповіді на запитання:

7. Яка дія і призначення команди неповного розгалуження?
8. Який вигляд має команда неповного розгалуження?
9. Яка дія і призначення команди переходу?
10. Який загальний вигляд має команда переходу?
11. Що таке мітка?
12. Як оголошують мітки?

Домашнє завдання.

15. Уведіть два числа і більше замініть сумою цих чисел.
16. Уведіть ціле число з діапазону 2..5. Виведіть його значення словом.
17. Уведіть число — номер дня тижня. Виведіть слово — назву цього дня.
18. Розгляньте 4-5 номерів поїздів і пункти їх призначення. Уведіть номер поїзда. Виведіть на екран назву кінцевого пункту.

Організація циклів

Вказівка повторення. Організація циклів.

Давайте спробуємо на природних явищах і прикладах з повсякденного життя визначити поняття циклу.

Якщо вас запитають, що таке цикл, то, напевно, ви, не замислюючись, відповісте, що це повторюваність чогось.

І це абсолютно правильно!

Визначити повторюваність пір року в природі — це цикл, кругообіг води в природі—це цикл, зміна дня і ночі—це цикл і багато інших процесів у природі повторюються, утворюючи циклічність.

Цикли в математиці — це явище, яке дуже часте зустрічається.

Наприклад, поки натуральні числа менші 10, то їх треба підсумовувати.

Іншими словами, ми знаходимо суму чисел від 1 до 10.

У цьому прикладі повторюється додавання натуральних чисел, поки виконується умова (числа менші за 10).

Такі цикли називаються циклом з передумовою, оскільки умова записується перед виконанням повторюваної групи операторів.

Увага! Цикл у програмуванні — це багаторазово виконувана група команд.

Як же ж цикли полегшують життя програмістам! Уявіть собі на хвилинку, що вам довелося б писати повторення одних і тих самих фрагментів програм багато разів! У Pascal передбачено три різновиди операторів циклу. Всі вони різні за своїм записом і застосуванням.

Загальний вигляд оператора циклу з передумовою:

while <логічний вираз> do P,

де логічний вираз приймає одне з двох значень **true** або **false,** P - простий чи складений оператор. Цикл з передумовою працює за таким принципом: *повторення оператора* P *буде виконуватися доти, поки логічний вираз в операторі циклу отримує значення* true. *Якщо тільки на деякому кроці циклу логічний вираз набуде значення* **false,** *цикл припинить свою роботу.*

Одне важливе зауваження: оскільки виконувані дії знаходяться за всіма службовими словами оператора циклу з передумовою, то не можна забувати про операторні дужки у випадку, коли тіло циклу складається з кількох таких дій.

Оператор дуже простий і зрозумілий. Схема алгоритму оператора повторення з передумовою не викличе у вас ніяких непорозумінь

Приклад 1 програми з оператором *While*

Обчислити суму S=1+1/2+1/3+…+1/50, використовуючи оператор While

```
Program Example_1 ;
VAR S: REAL; N:INTEGER;
BEGIN
S:=0; N:=1;
WHILE N<=50 DO
   BEGIN
        S:=S+1/N;
  N:=N+1;
   END;
WRITELN(' S=',S);
END.
```

Приклад 2 *Дано натуральне число N. Визначити кількість цифру числі.*

```
 Program Example_2 ;
Uses crt;
Var N: longint; Counter:  integer;
Begin
Clrscr;
Write ('Введіть число: '); Readln(N);
Counter := 0;
While N > 0 do
Begin
Counter:=Counter+1;      {Підрахунок кількості цифр)
N:=N div 10; {Відкидання останньої цифри}
End;
Writeln('Кількість цифр у заданому числі дорівнює', N);
Readkey;
```

End.

Загальний вигляд оператора циклу з післяумовою:

repeat P until <логічний вираз>;

де значення всіх параметрів такі самі, як і в попередньому операторі.

Цикл з післяумовою працює таким чином: *повторення оператора* **P** *відбувається доти, поки логічний вираз отримує значення* **false.** ***Якщо тільки на деякому кроці циклу*** *логічний вираз набуде значення* **true,** *цикл завершить свою роботу.*

Чи відчули ви відмінність між операторами **while ... do** та **repeat ... until?** Перший оператор спочатку перевіряє значення логічного виразу, а потім вирішує, виконувати йому опе-ратор, чи ні, а другий, навпаки, спочат-ку виконує вказаний оператор, а потім перевіряє, чи треба його виконувати ще раз.

Розглянемо приклад: Обчислити суму S=1+1/2+1/3+…+1/50, використовуючи оператор Repeat

```
 Program Example_3 ;
VAR S: REAL; N:INTEGER;
BEGIN
S:=0; N:=1;
REPEAT
        S:=S+1/N;
          N:=N+1;
UNTIL N>50;
WRITELN(' S=',S);
END.
```

Ще один приклад застосування оператора Repeat

Перевірка коректності введення. Дано три числа, що задають міри кутів трикутника. Визначити, чи можна побудувати трикутник, що має задані кути. Якщо ні, користувач має ввести інші дані.

```
Program Example_4 ;
Uses  crt;
Var a,b,c:   integer;
Begin
Clrscr;
Repeat
Write('Введіть міри кутів трикутника:  ' ) ;
Readln(a,b,c);
Until  (a>0)and(b>0)and(c>0)and(a+b+c)=180;
End.
```

Загальний вигляд оператора циклу з параметром:

for <параметр циклу> := $X_{поч}$ **to (downto)** $X_{кінц}$ **do** *P,*

де параметр циклу - змінна зчисленого типу, $X_{поч}$ - початкове значення параметра циклу, $X_{кінц}$ - кінцеве значення параметра циклу, *P* - простий чи складений оператор.

Службове слово **to** означає, що зміна значення параметра циклу йде від $X_{поч}$ до $X_{кінц}$ в порядку збільшення, а **downto** - в порядку зменшення.

Приклад програми з оператором *For*

Знайти суму всіх натуральних чисел від 1 до 100.

```
 Program Example_5;
  Uses crt;
 Var Sum,   i:   integer;
 Begin
 Clrscr;
 Sum:= 0;
  For i:= 1  to  100 do  Sum:» Sum + i;
 Writeln('Sum =',  Sum);  Readkey;
 End.
```

Розглянемо ще один приклад: обчислити суму S=1+1/2+1/3+…+1/50, використовуючи оператор For

```
 Program Example_6;
 VAR S: REAL; N:INTEGER;
```

```
BEGIN
S:=0;
FOR I:=1 TO 50 DO
    S:=S+1/I;
WRITELN(' S=',S);
END.
```

Запитання для перевірки засвоєння знань

3. Що називається циклом?
4. Які типи циклів вам відомі?

Домашнє завдання.

Вивчити теоретичний матеріал уроку.

4. Скласти програму, яка виводить на екран 50 чисел. Кожне число в окремому рядку.
5. Вивести на екран квадрати чисел від 1 до 10. Скласти програму в 3-х варіантах, використовуючи різні оператори циклу.

Цикли з параметром

1. Побудова таблиць. Розглянемо задачу: вивести 11 екран таблицю квадратів і кубів чисел від 20 до 30

```
20      400     8000
21      441     9261
22      484     10648
...
30      900     27000
```

Цю задачу розв'язують за допомогою циклічного алгоритму, яки можна утворити засобами команд **if** і **goto** так:

```
label dali, stop; var i:integer;
begin
i := 20;
dali: if i > 30 then goto stop;
writeln(i:6, i*i:7, i*i*i:8);
i := i+1;
goto dali;
stop: end.
```

Виявляється, що задачу можна розв'язати значно простіше за допомогою команди **for:**

```
var i: integer; {Це готова до виконання програма}
begin
for i := 20 to 30 do writeln(i:6, i*i:7, i*i*i:8);
end.
```

Задача 2. Один долар коштує 5,43 гривні. Вивести у вигляді таблиці вартість 1, 2,..., 10 доларів.

```
program Bank;
var d: integer; gr : real;
begin
writeln('Долари      Гривні');
for d := 1 to 10 do begin
gr := 5.43* d; writeln(d:4, gr:15:2)
end; readln
end.
```

На екрані отримаємо таку таблицю:

Долари	Гривні
1	5.43
2	10.86
3	16.29
4	21.72
5	27.15
6	32.58
7	38.01
8	43.44
9	48.87
10	54.30

Завдання 2. Виведіть таку ж таблицю для євро чи іншої валюти.

2. Обчислення елементів послідовності. Розглянемо випадок, коли елементи числової послідовності описуються формулою, нехай,

a_i =3+cos2i.

Задача. Обчислити та вивести на екран номери і значення перших n=10 елементів.

Це можна зробити за допомогою наступної програми:

```
program Elements;
var i,n: integer; a : real;
begin
n := 10;
for i := 1 to n do begin
a := 3 + cos(2*i); writeln(i:4, a:15:2)
end
end.
```

3. Пошук потрібних елементів. Задача пошуку потрібних елементів з-поміж усіх елементів послідовності розв'язують ***методом перегляду і аналізу всіх елементів.*** Для відбору потрібних елементів використовують деяку умову (логічний вираз). Усе це роблять за допомогою команди **if,** яка знаходиться в тілі команди циклу.

З а д а ч а . Нехай елементи числової послідовності описуються формулою a=2-2cos3i, *i* = 1,2,...,12. Вивести на екран номери і значення лише додатних елементів.

```
program Find;
var i,n: integer; a : real;
begin
n := 12;
for i := 1 to n do begin
a := 2 - 2*cos(3*i);
if a > 0 then writeln (i:4, a:15:2)
end
end.
```

Завдання. Виведіть на екран елементи, які більші, ніж 1 і менші, ніж 2.

Довідка 1. Якщо s — змінна типу char, то вивести на екран усі символи латинського алфавіту можна так: **for** S := 'A' to 'Z' **do** write(S);

Довідка 2. якщо крок зміни параметра циклу дорівнює -1, то форма написання команди циклу «для»має вигляд:

for <параметр> := <вираз 1> **downto** <вираз 2> **do** <команда>.

Наприклад, вивести на екран числа від 1 до **10** у зворотному порядку: 10 9 8 ... 1 - можна так: **for** i := **10** **downto** 1 **do** write(i:3).

Закріплення матеріалу.

7. Що таке цикли?
8. Який вигляд має команда циклу **for?**
9. Що таке параметр циклу?
10. Опишіть дію команди циклу **for.**
11. Як може змінюватися значення параметра в команді циклу **for?**
12. Чим відрізняється команда **for-to** від команди **for-downto?**

Вправи та задачі для самостійного розв'язання.

1. Визначте результати виконання таких команд (усно):

а) a:=5; **for** i:=1 to 2 **do** a:=a*i-2; a:=a+1 (відповідь: a=5);

б) **a:=1; for** i:=1 **to** 3 **do begin** a:=a+i; a:=a-1 **end;**

в) a:=0; **for** i:=1to 4 **do** a:=a+i; a:=a+2;

г) p:=1; **for** b:=8 **downto** 5 **do** p:=p+b; p:=p+1;

д) s:=0; **for** n:=7 **downto 4 do begin** s:=s+n; s:=s+1 **end;**

Складіть програми до наведених задач.

1. Для чисел від 1 до 10 обчисліть квадратні корені, кубічні корені та корені четвертого степеня. Результати наведіть у вигляді таблиці.
2. Виведіть на екран десять рядків таблиці відповідності між двома наступними мірами, знаючи що а) 1 фунт = 0,45359237 кг; б) 1 стоун 6,35029 кг; в) 1 унція = 28,353495 г; г) 1 драхм = 1,77185 г; д) 1 карат = 0,2 г; є) 1 гран - 0,068 г.
3. Виведіть таблицю переведення температури з градусів за шкалою Цельсія (C) у градуси за шкалою Фаренгейта (F) для значень градусів від a до b з кроком 1 за формулою F=1,8C+32.
4. Див. умову задачі № 3. Переведіть градуси за шкалою Фаренгейта у градуси за шкалою Цельсія.
5. Виведіть на екран 15 перших елементів числової послідовності, починаючи з першого, загальний елемент якої має вигляд 4 - 2sin2i.

Домашнє завдання.

1. Яка характерна особливість усіх трьох операторів циклу?
2. Визначте результати виконання таких команд:

а) a:=4; **for** i:=1 to 2 **do** a:=a*i-1; a:=a+2;
б) a: = 1**; for** i:=1 **to** 3 **do begin** a:=a+2*i; a:=a-2 **end;**
в) a:=-1; **for** i:=1 to 4 **do** a:=2*a+i; a:=a+2;
г) p:=30; **for** b:=7 **downto** 4 **do** p:=p-b; p:=p+5;
д) s:=0; **for** n:=6 **downto 3 do begin** s:=s+2*n; s:=s-1**end;**

6. Тіло вільно падає з висоти *h.* Виведіть таблицю значень висоти протягом перших *k* секунд падіння.

Дайте відповіді на запитання:

4. Що називається циклом?
5. Які типи циклів ви знаєте?
6. В чому особливість циклу з параметром?

Розв'язування задач.

Задача № 183

Компанія бабусь поїхала на мотоциклах на курси комп'ютерної грамотності. Попереду на мотоциклі без глушника їхала одна бабуся, за нею — дві, потім — три і т.д. Скільки бабусь їхало на заняття, якщо приголомшені пішоходи всього нарахували N рядів? Чи змогли бабусі зайняти всі місця у класі, якщо там стояло k рядів по L комп'ютерів у кожному? Скільки вільних місць залишилося?

Розв'язання: Зверніть увагу на те, що фактично ця задача зводиться до знаходження суми всіх натуральних чисел від 1 до *N.* У кінці задачі для повторення команди розгалуження учням пропонується визначити кількість зайнятих бабусями та вільних місць. Програма розв'язання даної задачі має такий вигляд:

```
Program Example_183;
 Uses  crt;
Var i,N,Sum:word;  {i - параметр циклу, N - хількість рядів мотоциклів,  Sum - кількість бабусь,  що приїхали на курси}
Place,k,L:word; (k - кількість рядів у комп'ютерному класі, L - кількість комп'ютерів у кожному ряду. Place - кількість місць, якої вистачило для бабусь}
Begin
Clrscr;
Sum:=0;
Write('Введіть кількість рядів мотоциклів: ');
Readln(N);
For i:=1 to N do Sum:=Sum+i;
Writeln('Кількість бабусь, що приїхали на курси ' ,Sum);
Writeln('Кількість комп''ютерів на курсах ',к*L);
If Sun<k*L
Then writeln('Бабусі не змогли зайняти всі місця.')
Else writeln( 'Бабусі зайняли всі місця.') ;
Place:=Sum - к*L;
If P1ace>0
Then writeln ('Бабусям не вистачило ' , Place,' місць.') ;
Readkey;
End.
```

Зауваження. У будь-якій програмі цикл **for** можна замінити рівносильним циклом **while,** але не навпаки. Тому цикл **while** вважається більш універсальним.

Порада. Якщо вам наперед відома кількість повторень оператора циклу, то доцільніше використовувати цикл із параметром. Якщо кількість повторень залежить від виконання деякої умови і постановка задачі передбачає обов'язкове виконання циклу хоча б один раз, то в цьому випадку рекомендується лише цикл з післяумовою. Якщо ж наперед невідома кількість повторень циклу, і, за умовою задачі, можливе невиконання циклу жодного разу, то прогноз єдиний - вам підходить лише цикл з передумовою! У циклах з перед/післяумовою створені вами умови повинні містити хоча б одну змінну величину, значення якої змінюється в тілі цього циклу. Інакше ви маєте нагоду створити «вічний двигун».

Розглянемо такий приклад. У математиці існує поняття факторіала. Факторіал обчислюється за такою формулою:

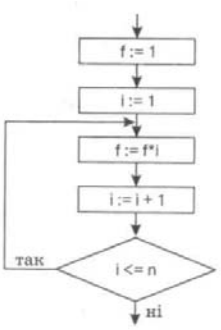
n! = 1*2*3*4*5* ... **n*. Тобто n! - добуток усіх натуральних чисел від 1 до *n.*

```
 program factorial;
var i, f: integer;
begin
```

```
f:=1;
write ('Задайте значення числа n: ');
readln (n);
for i := 1 to n do f :=f*i;
writeln ('Значення факторіала числа ', n:5, 'дорівнює: ', f: 10)
end.
```

Накопичення добутку відрізняється від накопичення суми тим, що початкове значення обов'язково повинне дорівнювати 1!

Схема алгоритму цієї програми зображена на малюнку .



Якщо ми лише незначним чином змінимо нашу програму, ввівши виведення результату в тіло циклу, то отримаємо зовсім інший ефект. Результатом її виконання буде виведення значень факторіала на кожному кроці. Такий принцип називається *по-кроковим виведенням* результатів.

```
for i := 1 to n do begin
f := f*i;
write ('Значення факторіала числа ', i:5);
writeln (' дорівнює: ', f:10)
end;
```

Як розібратися, коли який оператор циклу використовувати? Саме в цьому й полягає майстерність програміста - оцінити постановку задачі, скласти алгоритм, залучити всі свої аналітичні здібності й остаточно вибрати оптимальний варіант циклічного оператора.

Розглянемо задачу: Дана послідовність з N чисел. Визначити середнє арифметичне непарних від'ємних чисел, добуток чисел, кратних 5 із проміжку [-20,20], кількість нульових чисел и замінити числа, менші ніж 3, на 100.

```
Program prim;
Const N=20;
Var Sr :real; S, kol, kolo, i, N, p, x: integer;
 Begin
S:=0; p:=1; kol:=0; kolo:=0;
 For i:=1 to n do
Begin
Writeln('введіть число последовності'); Readln(x);
If (x mod 2 <> 0) and (x<0) then Begin S:=s+x;  Kolo:=kolo+1; End;
If (x mod 5 =0) and (x >=-20) and (x <20) then p:=p*x;
If x=0 then kol:=kol+1;
Ifx<3 then Begin x:=100; Writeln('число змінилося' ,x); End;
End;
Sr:=s/kolo; Writeln('середнє=', sr:8:4, 'добуток = ', p, 'кількість нульовых чисел=', kol);
End.
```

Розглянемо задачу: Вивести на екран таблицю множення чисел (таблицю Піфагора)

```
Program Pifagor;
Var j, i: integer;
Begin
For i:=1 to 9 do
Begin
For j:=1 to 9 do
Write(i*j:3);
Writeln;
End;
Readln
End.
```

Домашнє завдання.

Розв'язати задачі:

5.Виведіть на екран у вигляді таблиці номери і значення перших десяти елементів числової послідовності, загальний елемент якої має вигляд $7 - 5\sin i^2$.

6.Серед перших 20 елементів числової послідовності $3 - 3\sin i^2$ виведіть на екран номери і значення лише від'ємних елементів, тут $i = 1,2,...20$.

7.Розгляньте елементи числової послідовності $5 - 3\cos 2i$ від 10-го до 20 го і виберіть серед них (виведіть їхні номери на екран) більші, ніж 4.

8.Перші десять елементів (i=1,2,...10) числової послідовності $2 — 2|\sin 3i|$ перетворіть за таким правилом: додатні

елементи збільшить у два рази, від'ємні елементи зменшить на 2. Виведіть у вигляді таблиці номери елементів, їхні попередні та нові значення.

Обчислення суми, добутку та кількості.

7. Що таке цикли?
8. Який вигляд має команда циклу for?
9. Що таке параметр циклу?
10. Опишіть дію команди циклу for.
11. Як може змінюватись значення параметра в команді циклу for?
12. Чим відрізняється команда for-to від команди for-downto?

1. Обчислення суми. Під час обчислення суми початкове значення змінної, де нагромаджуватиметься сума, наприклад s, має дорівнювати нулеві. Для цього використовують команду присвоєння s := 0.

З*адача* 1. Скласти програму для обчислення суми цілих чисел від 1 до 100.

```
program Suma1;
var s, number : integer;
begin
s := 0;
for number := 1 to 100
do
s := s + number;
writeln("сума = ', s:5)
end.
```

Змінна number отримуватиме такі значення: 1, 2, 3, ...,100, а змінна s - такі: 0, 1, 3, 6, 10, 15, 21,…, 5050.

Відповідь: 5050.

Завдання 1. Обчисліть суму чисел від **100** до **200.**

2. **Обчислення добутку.** Під час обчислення добутку початкове значення змінної, де нагромаджуватиметься добуток, наприклад d, має дорівнювати одиниці (d := 1).

Задача 2. Обчислити добуток перших п'яти елементів послідовності, які задані виглядом загального елемента *a*= 2+|sin3i|, i=1,2,...,5.

```
program dobutok;
var i, n : integer; a, d : real;
begin
d := 1; n :=5;
for i := 1 to n do begin
a := 2 + abs(sin(3*i)); d := d *a
end;
writeln('добуток =', d:10:2); readln end.
```

Завдання 2. Обчисліть добуток елементів більших, ніж 2,5.

3. Обчислення кількості. Під час обчислення кількості початкове значення змінної, де нагромаджуватиметься кількість, наприклад к, має дорівнювати нулеві (k := 0), а в тілі циклу має бути команда k:=k+1.

З*адача 3.* Скільки елементів послідовності a_i.=1-cosi, де *i* = 1, 2,...20, задовольняють умову $0<a_i.<1$?

```
program Kilkist;
var i, n, k : integer; d : real;
begin
k := 0; n :=20;
for i ;= 1 to n do
begin  a := 1 - cos(i);
qif (a>0) and (a<l) then k := k + 1
end;
writeln('кількість=', k:2); readln
end.
```

Домашнє завдання.

4. Обчисліть суму додатних елементів числової послідовності 3sin2i, де i=1,2,...,15.

5. Визначіть скільки серед елементів послідовності cos3i, i= 1,2,...,12, є від'ємних.
6. Обчисліть добуток елементів послідовності 2-sin2i, i=1,2,...10, значення яких є більші, ніж 1 і менші, ніж 2.
5. Обчисліть суму від'ємних значень елементів послідовності 1-2cos3i, i=1,2,…,20.
9. У якій з послідовностей більше додатних елементів: 1) 2sin3i, i=1,2,...,10 чи 2) 2-3cosi, i=1,2,...,15?
10. Обчисліть z=sin1+sin2+sin3+...+sinn, де n=30.
11. Обчисліть $y=x+x^3/3+x^5/5+...x^{21}/21$, де x=0,5.

Методи перебирання варіантів.

1. Аналіз чисел. Розглянемо задачу виведення на екран лише тих чисел, які володіють деякою властивістю.

Задача1 (Аналіз чотиризначних чисел). Визначити всі чотиризначні числа, сума цифр яких дорівнює їхньому добутку.

Задачу розв'язують ***методом повного*** *перебирання* усіх можливих варіантів з використанням алгоритмічної конструкції «вкладені цикли **for»**.

Позначимо чотири цифри шуканих чисел як i, j, k, m, а їхню кількість — s. Наступна програма дає розв'язок задачі:

```
program Numbers;
var i,j,k,m,s,a : integer;
 begin
s := 0;
for i := 1 to 9 do
for j := 0 to 9 do
 for k := 0 to 9 do
for m := 0 to 9 do
if i+j+k+m = i*j*k*m then b
egin
s:= s+1;
a := 1000*i + 100*j + 10*k + m;
writeln(a)
end;
writeln('усього чисел є ', s); readln
end.
```

Завдання 1. Виконайте програму і переконайтеся, що таких чисел є 12 серед 9000: 1124, 1142, ..., 4211. Розв'яжіть цю задачу для тризначних чисел.

2. **Задача про решту.** Застосуємо метод перебирання варіантів для розв'язування задачі про видачу решти.

Задача2 (про решту). У касі є монети номіналом 3, 5, 10. Скількома способами касир може видати покупцеві решту на деяку суму Num?

Задачу розв'язуємо шляхом перебирання можливих варіантів і вибирання з-поміж них потрібних способів видачі решти. Тут, щоб обмежити перебирання варіантів, уведемо величини Xmax, Ymax, Zmax — максимальні кількості монет для видачі решти монетами одного номіналу. Введемо також величину p — ознаку того, що задача має розв'язок. Програма матиме вигляд

```
program Change; uses Crt;
var Num.Xmax, Ymax, Zmax, x,y,z, p:integer;
begin
clrscr;
write(' У ведіть суму: ');
readln(Num);
P :=0;
Xmax := Num div 3;
Ymax := Num div 5;
Zmax := Num div 10;
writeln('  3  5  10 Усього монет');
writeln(' ..........................................................');
for x := 0 to Xmax do
for y := 0 to Ymax do
for z := 0 to Zmax  do
if 3*x+5*y+10*z = Num then
begin
 writeln(x:3, y:3, z:3, x+y+z:8);
```

```
  p :=p+1
  end;
  writeln;
  if p = 0 then writeln("Немає варіантів')
  else writeln('Усього способів є ', p); readln
  end.
```

Виконаємо програму для решти 23. Результати будуть такі:

```
Уведіть число? 23
3  5 10 Усього монет
1  0  2        3
1  2  1        4
1  4  0        5
6  1  0        7
Усього способів є 4
```

Завдання 2. Поекспериментуйте з програмою для різних значень решти. Модифікуйте програму для номіналів монет: 1**,** 2, 5, 10.

3. Задача про високосні роки. Високосний рік має 366 днів, а звичайний — 365. Високосним є рік, значення якого ділиться на 4, не ділиться на 100, але ділиться на 400.

Задача 3. Визначити, скільки днів було в двадцятому столітті.

```
  program AboutDays;
  var i: 1901. .2000;
  Vysokosnyi : boolean; KilDniv : longint;
  begin
  KilDniv := 0; for i :- 1901 to 2000 do
  begin
  Vysokosnyi := (i mod 4 = 0) and (i mod 100<> 0)
  or (i mod 400 = 0);
  if Vysokosnyi then KilDniv :=KilDniv + 366
  else KilDniv := KilDniv + 365
  end;
  writeln('Було ', KilDniv, ' днів');
  writeln ('Бажаємо успіхів у XXI столітті')
  end.
```

Завдання 3. Виконайте програму і поекспериментуйте з нею. Скільки днів було у XVI столітті, скільки днів буде у XXI?

Домашнє завдання.

10. Обчисліть s: s:=0; **for** i:=1 to 2 do **for** j:=1 to 3 do s:=s+i*j.
11. Уведіть чотиризначне число, наприклад, 1998. Обчисліть суму його цифр. Підказка: використайте операції **div** і **mod.**
12. Виведіть на екран усі двозначні числа, які діляться на 3 або на 5.
13. Визначіть, скільки є «щасливих» автомобільних номерів в одній серії з комарами від 0000 до 9999. Номер є щасливим, якщо сума першої половини цифр дорівнює сумі другої.
14. Визначіть, скільки є «щасливих» автобусних квитків в одній серії з номерами від 000000 до 999999.
15. Визначіть цифри *a, b,* с, які задовольняють рівняння *abb + cab = bac.* Дослідіть можливість скорочення повного перебору.
16. Визначіть кількість тризначних чисел, сума цифр у яких дорівнює деякому заданому числу *n.* Виведіть їх на екран. Використовуйте лише два вкладені цикли.
17. Скільки днів пройшло від початку року до деякої заданої дати?
18. Скільки днів пройшло від вашого дня народження?

Цикли з передумовою (1).

Приклад1. Вивести у вигляді таблиці числа від 20 до 30 та їхні квадрати і куби за допомогою команди **while** можна так:

```
  i := 20;
  while i <= 30 do
  begin
  writeln(i:4, i*i:6, i*i*i:8);
  i :=i + 1
  end;
```

Особливість команди **while.** У циклі «доки» на відміну від циклу «для» початкове значення параметра треба задавати "вручну" перед командою циклу (команда i:=20). Крім цього в тілі циклу треба записати команду зміни значення параметра (i := i + 1).

Зауваження. У будь-якій програмі цикл **for** можна замінити рівносильним циклом **while,** але не навпаки. Тому цикл **while** вважається більш універсальним, ніж цикл for.

Задача 1. Обчислити суму перших ста натуральних чисел, використовуючи цикл **while.**

Програма Suma2 має такий вигляд:

```
program Suma2;
var number,s : integer;
begin
s := 0;
number := 1;
while number <= 100 do
begin
s := s + number; inc (number) end;
writeln (s);
end.
```

2. Табулювання функції. Табулювання функції — це побудова таблиці значень функції *y = f(x)* для різних значень аргументу *x,* де *x* змінюється на деякому проміжку *[a; b]* з кроком *h.*

З а д а ч а 2. Протабулювати функцію *y* = xsinx на проміжку [1; 2] з кроком *h* = 0,1.

```
program Tabul;
var x, a, b, h, y : real;
begin
write('Введіть a, b, h (тут 1  2  0.1):  ');
readln(a, b, h);
writeln('  x    y=xsinx  ');
x := a;
while x <= b + h/2 do begin
y := x * sin(x);
writeln(x:6:l, y:9:2);
x := x + h
end
end.
```

Умова x <=b не завжди забезпечує попадання в останню точку *x=b* через особливості машинної арифметики. Тому тут використано умову x<= b + h/2. Виконавши алгоритм, отримаємо таблицю значень заданої функції:

X	y=xsinx
1.0	0.84
1.1	0.98
1.2	1.12
1.3	1.25
1.4	1.38
1.5	1.50
1.6	1.60
1.7	1.69
1.8	1.75
1.9	1.80
2.0	1.82

3. Цикли з наперед невідомою кількістю повторень. Цикли із заздалегідь невідомою кількістю повторень не можна записати за допомогою команди **для.** їх треба записувати за допомогою циклу «доки». Ось чому ***цикл «доки» є більш універсальний, ніж цикл «для».***

Задача 3. Визначити дійсне додатне число *a,* для якого виконується співвідношення a/2 = 0 у комп'ютерній арифметиці дійсних чисел. Таке число характеризує нижню додатню межу типу даних **real.**

```
program MinRealNumber;
var a : real;
begin
a:=1;
while a/2 > 0 do a := a / 2;
writeln('a =', a)
```

```
end.
```

Відповідь: a - 2.9E-39.

Виконайте вправи :

1. Якого значення набуде змінна після виконання команд:
а) p:=4; while p<10 do p:=2*p+1; p:=p+1 (відповідь: p = 20);
б) p:=4; while p<10 do begin p:=2*p+1; p:=p+1 end;
в) p:=5; while p>2 do p:=(p-3)*2; p:=p-3;
2. Якого значення набуде змінна після виконання команд:
а) p:=7; while p>=5 do p:=(p+3)/2; p:=p-1;
б) p:=7; while p>=5 do begin p:=(p+3)/2; p:=p-1 end;

Домашнє завдання.

Дайте відповідь на запитання:

4. Яке призначення циклу «доки»?
5. Який загальний вигляд має команда циклу **while?**
6. Чому цикл «доки» вважають більш універсальним, ніж цикл «для»?

5. Визначте суму парних чисел від 1 до 100.
6. Обчисліть добуток непарних чисел від 1 до 13. ,
7. Протабулюйте функцію cos2x: на проміжку [-2;2] з кроком 0,25 і обчисліть суму додатних значень функції.
8. Протабулюйте функцію sin3x на проміжку [-1;1] з кроком 0,2 і обчисліть кількість від'ємних значень.

1. Пошук значень функції, які володіють деякою властивістю

Розглянемо задачу, де потрібно протабулювати функцію і проаналізувати її значення.

Задача1. Протабулювати функцію *y* = sinx на проміжку [0; π], з кроком *h* = 0,1 і обчислити середнє арифметичне значень функції більших, ніж 0,1 і менших, ніж 0,6.

Оскільки крок зміни параметра циклу дробове число, використаємо цикл **while.**

```
program FindSerednie;
var x, y, s, si, h, xk: real; n : integer;
begin
x := 0; xk := pi; h := 0.1; s :=0; n:=0;
while x <= xk + h/2 do begin
y := sin(x) ;
writeln(x:3:1, y:6:2);
if (y > 0.1) and (y<0.6) then
begin s := s + y; n := n + 1 end;
x := x + h;
end;
if n > 0 then
begin si := s / n; writeln ('Середнє значення =', s1) end
else writeln('Таких значень немає n=0')
end.
```

Завдання. Визначіть кількість від'ємних і кількість додатних значень функції, якщо крок *h* = 0,05.

2. Пошук максимального чи мінімального значення. Щоб серед багатьох даних визначити максимальне (чи мінімальне), потрібно спочатку перше дане прийняти за шукане і порівняти його з наступним Якщо наступне дане більше (менше), то взяти його за шукано порівняти з тими даними, що залишилися і т.і.

З*адача* 2. На Олімпійських іграх у фіналі 5 спортсменів (men) змагаються у стрибках. Скласти програму, яка потребуватиме ввести номер спортсмена (n), результат чергового стрибка (rez) і виведе найкращий (максимальний) результат (rezmax) і номер спортсмена-переможця (nchemp).

```
program Compet;
const men = 5;
var n, nchemp, sportsman : integer;
rez, rezmax: real;
begin
rezmax := 0; nchemp :=0;
for sportsman := 1 to men do
begin
write('Введіть номер спортсмена і його результат:'); readln(n, rez);
if rez>rezmax then begin rezmax := rez; nchemp := n end;
end;
writeln('Переможцем став', nchemp);
writeln('Найкращий результат ', rezmax); readln
```

```
end.
```

Задача 3. Протабулювати функцію *y* = sinx на проміжку [0; π], кроком *h* = 0,1 і визначити її мінімальне значення (ymin) і точку, в якій воно досягається.

```
program FindMinimum;
var x, y, ymin, xmin, h, xk: real; n : integer;
begin
x := 0; xk := pi; h := 0.1;
ymim := sin(x);
xmin := x;
while x <= xk + h/2 do
begin
y := sin(x);
writeln(x:3:1, y:9:2);
if y < ymin then
begin ymin := y;   xmin := x end;
x := x + h ;
end;
writeln('ymin= ', ymin:5:3, ' в точці xmin= ', xmin:3:1)
end.
```

Завдання. Визначіть максимальне значення функції.

3. Задача про найбільший спільний дільник. *Найбільший спільний дільник (НСД)* двох чисел *a* і *b* визначають за допомогою алгоритму Евкліда. Алгоритм ґрунтується на таких властивостях цілих чисел: якщо *a* > *b,* то НСД(a, *b)* = НСД(a-b, *b);* якщо a < *b,* то НСД(a, *b)*=НСД(a, *b-a);* якщо a = *b,* то НСД(a, *b)* = *a.* Застосуємо цю властивість у циклі. Розглянемо програму Evklid.

```
program Evklid;
var a,b : integer;
 begin
Write('Введіть два числа:');
readln(a.b);
while not (a=b) do
if a > b then a :- a - b else b := b - a;
writeln (a); readln
end.
```

Виконаємо трасування програми для чисел 18 і 24: НСД(18, 24) =НСД(18, 6) = НСД(12, 6) = НСД(6, 6) = 6.

4. Задача про прості числа. Розглянемо задачу, для розв'язування якої застосуємо додаткову змінну-прапорець, значення якої сигналізує про досягнення мети.

Задача 4. Визначити, чи задане натуральне число *a* є простим.

Розглянемо програму SimpleNumber. Тут спочатку перевіряється, чи число парне, а далі — чи воно ділиться без остачі на менші від нього непарні числа. Якщо так, то число не є простим, тому аналіз закінчують достроковим виходом з циклу «поки». Якщо число не ділиться на жодне менше непарне число, то робимо висновок: число просте.

```
program SimpleNumber;
var a, i : integer; flag : boolean;
begin
 write('Уведіть число '); readln(a);
flag := true;   { це ознака простого числа }
if a mod 2 = 0 then flag := false;
i := 3;    {тепер аналізуємо ділення на непарні числа}
while (i < a) and flag do
if a mod i = 0 then flag :=false
else i := i+2;
if flag then writeln('Число просте')
else writeln('Число не просте')
end.
```

Завдання. Переконайтеся, що в заданому алгоритмі перебір варіантів можна скоротити, замінивши у циклі **while** логічний вираз i < a виразом i<=sqrt(a).

Домашнє завдання

5. Протабулюйте функцію y=cos2x на проміжку [-2;2] з кроком 0,25 і обчисліть середнє арифметичне від'ємних значень.

6. Протабулюйте функцію y=2sin2x на проміжку [-2;2] з кроком 0,25 і визначіть: а) максимальне значення; б) мінімальне значення.
7. Протабулюйте функцію y=xcos2x: на проміжку [-2;2] з кроком 0,2 і визначіть максимальне значення функції та значення аргументу, для якого воно досягається.
8. Протабулюйте функцію y=sinxcos2x на проміжку [-4;4] з кроком 0,5 і обчисліть суму квадратів максимального і мінімального значень функції.
5. Числа *a, b, c* називаються «числами Піфагора», якщо $a^2+b^2=c^2$. Визначіть *n* наборів «чисел Піфагора».

Цикли з післяумовою.

Задача 179

На дверях ліфта висіло загрозливе попередження про те, що двері самі зачиняються в той самий момент, коли зайвий за вагою пасажир переступить поріг ліфта. Котрий пасажир постраждає, якщо ліфт витримує вагу не більше S кг, а вага пасажирів, що стоять у черзі до ліфта, дорівнює відповідно $a_p a_y a_y$ *... a ?*

Розв'язання: У цій задачі зручніше використовувати цикл з післяумовою, тому що спочатку необхідно дати можливість «увійти» пасажиру в ліфт, а потім перевіряти, чи витримає його ліфт. Умовою виходу з циклу буде перевищення сумарної ваги пасажирів, що увійшли в ліфт, деякого заданого критичного значення. Для зберігання ваги чергового пасажира в цій задачі ми будемо використовувати одну й ту саму змінну *(A)*, оскільки після перевірки вага пасажира нас уже не цікавить. Програма має вигляд:

```
Program Example_179;
Uses crt;
Var N:word;  {N - номер пасажира,  що увійшов у ліфт}
Sum,A,S:real;  {Sum - сумарна вага пасажирів, що знаходяться в ліфті, A - вага чергового
пасажира, що увійшов до ліфта, S - критична вага, що може бути піднята ліфтом}
Begin Clrscr;
Sum:=0;
N:=0;  {На початку роботи програми в ліфті немає пасажирів}
Write('Введіть  критичну вагу,  що піднімає ліфт:  '); Readln(S);
Repeat
Write(*Введіть вагу чергового пасажира:  '); Readln(A);
Sum:=Sum+A; N:=N+1;
Until  Sum>S;
Writeln('Постраждає ' ,N,'-й пасажир.');
Readkey;
End.
```

Задача 181

Умова: Капосний папуга навчився висмикувати у дідуся Василя волосся, яке ще залишилося у того на голові. Почавши з однієї волосини, він щодня збільшував порцію вдвічі. Через скільки днів дідусеві не знадобиться гребінець, якщо спочатку в нього на голові було аж N волосин?

Розв'язання: Аналогічно до попередньої задачі, аналізувати наявність волосся на голові слід після того, як папуга вже висмикнув чергову порцію волосся. А «знущання» над дідусем скінчиться тоді, коли гребінець йому стане непотрібним, тобто кількість волосся на голові дорівнюватиме нулю.

Зверніть увагу, що в цій задачі змінна *S* використовується для підрахунку чергової порції волосся, що підлягає висмикуванню капосним папугою.

```
Program Example_181;
Uses crt;
Var S,N,Sum:longint;  {S - кількість волосся, що буде висмикнутим,  Sum - кількість волосся,  що
залишилося в дідуся на голові,  N - початкова кількість  волосся}
Day:word; {Day -  номер дня,  який папуга знущається над дідусем}
Begin
Clrscr;
Write('Початкова кількість волосся а дідуся на голові:  ') ;
Readln(N) ;
If N=0 Then writeln('Дідусь  уже лисий,  папузі  нічого робити!')
Else
begin
Day:=0;   Sum:=N;   S:=1;
{Початкова кількість волосся, що буде висмикнуте папугою}
Repeat
Sum:=Sum-S;  {Зменшення дідусевого волосся}
S:=S*2;
```

```
Day:=Day+1;  {Підрахунок номеру дня)
Until Sum<=0;
Writeln('Папуга знущався над дідусем  ',Day,'  днів. ' );
End;
Readkey;
End.
```

Задача 209

Умова: На скільки років необхідно покласти в банк суму X грошових одиниць, щоб одержати суму N грошових одиниць (N > X), якщо банк нараховує 200 % річних?

Розв'язання. Очевидно, що умовою виходу з цього циклу буде отримання заданої суми грошей. Якщо за умовою задачі *N > X, то* кожну перевірку ми будемо виконувати після того, як до вкладеної суми додамо щорічний банківський процент. Отже, програма має вигляд:

```
Program Example_209;
Uses crt;
Var X,N:real;  {X - початковий внесок,  N - бажана сума)
Rez:real;  {Rez  - результуюча сума на рахунку)
Years : longint;  {Years - термін перебування грошей в банку)
Begin
Clrscr;
Write('Введіть початкову суму внеску:  ') ;
Readln(X);
Write('Введіть бажану суму внеску:  ');
Readln(N);
If N<=X Then writeln('Ви вже маєте бажану суму!')
Else
Begin Rez:=X; Years:=0;
Repeat
Rez:=3*Rez; {200% річних збільшують  за рік внесок втричі}
 Years:=Years+l;
Until  Rez>=N;
Writeln ('Ви отримаєте бажану суму через ' , years,' років') ;
End;
Readkey;
End.
```

Задача 231

Скласти програму, яка б допомогла працівникам ДАІ визначати кількість порушників перевищення швидкості на трасі, якщо відомо, що на даному проміжку траси встановлено обмеження на швидкість V_{max} а прилад фіксує швидкість автомобілів V_1 $У_2$ ..., V_n.

Розв'язання: В даній задачі ніяким чином не обумовлена умова виходу з циклу, тому є пропозиція: процес підрахування порушників необхідно закінчити тоді, коли чергове введене число буде недодатнім (дійсно, з від'ємною або нульовою швидкістю автомобіль рухатися не може). Для тимчасового зберігання значення швидкості чергового автомобіля ми будемо знову використовувати одну змінну. Програма, що виконує задані обчислення, має наступний вигляд:

```
Program Example_231;
Uses crt;
Var V,Vmax:real;
{V - швидкість автомобіля, Vmax - макс, дозволена швидкість}
 Count:longint;  {Count - кількість порушників)
Begin
Clrscr;
Count:-0;  {На початку роботи порушники відсутні)
Write ('Значення максимально дозволеної швидкості:  ');
Readln(Vmax);
Vmax:=abs (Vmax) ; {Знаходження модуля для виключення помилки введення від'ємної,
максимальної швидкості}
Repeat
Write('Значення швидкості чергового автомобіля: ');
Readln(V);
```

```
If V>Vmax then Count:=Count+I;
Until V<=0;
Writeln('Кількість порушників ',Count);
Readkey;
End.
```

Задача 251

Дано натуральне число n і дійсні числа a_1, a_2, ..., a_n. Відомо, що в заданій послідовності є хоча б одне нульове значення. Розглядаючи члени послідовності, що розташовані до першого нульового члена, визначити середнє арифметичне членів.

Розв'язання: Для розв'язання цієї задачі значення *n* є зайвим, якщо серед членів послідовності буде хоча б один нульовий елемент, тому ми враховувати цю змінну не будемо (у випадку відсутності нульового члена послідовності змінна *n* може використовуватись як додаткова для виходу з циклу, щоб виключити зациклення програми).

Отже, оскільки ми не знаємо, коли зустрінеться нульовий елемент, у програмі знаходиться сума всіх чисел послідовності (змінна *sum)* та кількість введених чисел (змінна *count),* а після виходу з циклу вже знаходиться безпосередньо середнє арифметичне членів послідовності як результат ділення суми на кількість чисел, зменшену на одиницю. Зменшення на одиницю відбувається тому, що фактично в тілі циклу буде підраховано один зайвий нульовий елемент (останній).

Програма має вигляд:

```
Program Example_251_5;
Uses crt;
Var count:word;  {count - кількість членів послідовності до першого нульового елемента}
a, Sum : real;   {a - черговий член послідовності,  Sum - сума членів послідовності до першого «0»)
 SA:real;   {SA - середнє арифметичне}
Begin   Clrscr;
Sum:=0;  count:=0;   {Початкові значення дорівнюють «0»}
repeat
write('Введіть черговий член послідовності:   ') ;
readln(a);
Sum:=Sum+a; count:=count+1;
until a=0;
SA:=Sum/(count-1);
Writeln('Середнє арифметичне = ',SA:8:2);
Readkey;
End.
```

Запитання для перевірки засвоєння знань

6. *Що таке цикл з передумовою?*
7. *Що таке цикл з післяумовою?*
8. *Як в програмі уникнути «зациклення» ?*
9. *Які операції називаються бінарними?*
10. *З яких розділів складається програмний блок?*

Домашнє завдання:

1. Складіть програму обчислення добутку парних чисел, менших 15.
2. Складіть таблицю значень функції y = 5 x-2 на відрізку [1; 20] із кроком h = 2.3.
3. Складіть програму обчислення суми 80 перших членів арифметичної прогресії, якщо a_1 = 10; d = 3.

Обробка рекурентних послідовностей.

1. Поняття про рекурентні формули. Прикладами рекурентних формул є формули для обчислення *наступного елемента* арифметичної чи геометричної прогресії, якщо відомо *попередній елемент* і різниця чи знаменник прогресії:

$a_{n+1}=a_n+d$, $n=$ 0,1,2,... $b_{n+1}=b_n*q$,n=0,1,2,...

У задачах про прогресії зазвичай відомо значення першого елемента, різниці чи знаменника, що достатньо для їхнього розв'язування.

2. Поняття про метод ітерацій. ***Ітераціями*** називаються послідовні уточнення значення деякої величини, отримані в результаті застосування рекурентних формул.

З а д а ч а 1. Обчислити із заданою точністю квадратний корінь від числа *n,* де *n* — це номер варіанта. Якщо n =1, 4, 9, 16, 25, 36, то число *n* збільшити на 50. Обчислення виконати з точністю 0,01. Визначити скільки було ітерацій. Виконати обчислення ще два рази, задавши точність **0,001** та **0,0001.**

Результати трьох експериментів записати у таблицю спостережень:

Число		Корінь	Точність	Кількість ітерацій
3)	*n* ?	**0,01**	?	
4)	*n* ?	**0,001**	?	
3)	*n* ?	**0,0001**	?	

Теоретичні відомості. Квадратний корінь будь-якого додатного числа *n* можна визначити за такою рекурентною формулою:

$b_{i+1} := (b_i + n/b_i)/2$, i=1,2,..., де $b_1 = n$.

Алгоритм:

3. Позначити через b_1 будь-яке початкове наближення до шуканої величини, ввести *n* і присвоїти *b1:= n;*
4. Позначити через *e* задану точність і присвоїти

e := 0,01**,** i := 1,

5. Обчислити *b2* := *(b1+ n/b1)/2.*
6. Доки |*b2 – b1*|> *e,* виконати обчислення

b1 := *b2,*
b2 := *(b1+ n/b1)/2,*
i := i + 1.

5. Після виходу з циклу «доки» останнє значення (b2) прийняти за відповідь.

Завдання. Складіть відповідну програму самостійно.

3. Числа Фібоначчі. Цю задачу сформулював і розв'язав італієць Фібоначчі у 1228 році.

Задача 2. Є пара кроликів. Вона дає приплід — нову пару кроликів — на третій місяць, а пізніше — щомісяця. Скільки пар кроликів буде наприкінці року?

Розв'яжемо задачу. Є послідовність чисел, де перші два числа — це 1 та 1, треба знайти інші. Третім числом буде 2. Розв'язок дають числа, які отримують за таким правилом: кожне наступне число (c) в послідовності є сумою двох попередніх (a та b). Ці числа називаються числами Фібоначчі. Визначимо десять чисел послідовності.

```
program Rabbits;
var a, b, c, i : integer;
begin
write('Числа Фобіначчі: 1  1');
a:= 1; b := 1; for i := 1 to 10 do
begin
c := a + b; a := b; b := c;
write(c:3)
end
end.
```

Отримаємо числа Фібоначчі:

1 1 2 3 5 8 13 21 34 55 89 144.

Домашнє завдання.

5. Перший член і різниця арифметичної прогресії дорівнюють відповідно a_0 і *d* (задайте їх довільними). Виведіть десять перших членів прогресії.
6. Перший член і знаменник спадної геометричної прогресії дорівнюють відповідно b_0 і *q*. Виведіть десять перших членів прогресії.
7. . Обчисліть 12 значень елементів послідовності, яка утворюється за допомогою такої рекурентної формули: $a_{n+2} = a_{n+1} - a_n$ $a_0=1$, $a_1=2$.
8. Задача Фібоначчі № 2. Є одна пара кролів. Пара починає щомісяця народжувати нову пару на третій місяць життя. Одна пара кролів живе 6 місяців. Скільки пар кролів буде через 12 місяців?

Складання алгоритму з одним циклом.

6. Чому не потрібно брати в операторні дужки групу операторів між словами repeat- until?
7. Якого типу може бути параметр циклу і?
8. Записати блок-схеми типів циклічних алгоритмів.
9. В якому випадку виникає ситуація зациклення програми?
10. Чи можна змінювати параметр цикле for в тілі циклу?

1. Паскаль-Рулетка. У наступному прикладі число повторень циклу заздалегідь невідомо, тому замість циклу з лічильником краще використовувати одну з різновидів циклу з перевіркою умови. Пропонуємо пограти в просту, але азартну гру на вгадування цілого числа від 1 до 10. Нехай програма «загадає» таке число, а користувач уведе передбачуване значення. Якщо число вгадане, програма привітає переможця, а якщо ні — попросить його повторити спробу ще раз. Кожна безуспішна спроба знижує призові бали. На самому початку гравцеві призначається 10 призових балів. Опис алгоритму:
- вибрати випадкове ціле число від 1 до 10;
- вивести запрошення на уведення цілого значення;
- якщо уведене число менше задуманого, сповістити про це гравцеві, інакше повідомити його про те, що уведене число більше задуманого;
- повторювати уведення цілого значення доти, поки число не буде вгадано;
- вивести привітання переможцеві й повідомити його про набране число балів;
- завершити роботу.

Не виключена можливість того, що число буде вгадано відразу. У цьому випадку вже не треба виводити підказку гравцеві, тому варто використовувати цикл із передумовою while. . .do:

```
program roulette;
var number, guess, bonus : Byte;
begin
 bonus := 10;
Randomize;
number := Random(11);
WriteLn('Задумане ціле число від 0 до 10. Вгадайте!');
Wri teLn;
WriteLn('Уведіть ціле число від 0 до 10');
ReadLn(guess);
while guess <> number do
begin
Dec(bonus);
WriteLn( 'Ви не вгадали.');
Wri teLn;
if guess < number then
WriteLn('Ваше число менше задуманого')
else
WriteLn('Ваше число більше задуманого') ;
WriteLn('Спробуйте ще раз!');
ReadLn(guess);
end;
WriteLn('Поздоровляю! Ви вгадали й набрали ', bonus, ' балів');
WriteLn('Натисніть <Enter>');
 ReadLn;
 end.
```

У цій програмі використовуються нові оператори. Це Randomize - початкова установка спеціальної процедури - «генератора» випадкових чисел Random(n), що видає випадкові цілі числа від 0 до n - 1, а також Dec (bonus) - виклик процедури, що зменшує на одиницю значення змінної bonus.

2. Спробуємо розбагатіти. Розглянемо приклад використання циклу з післяумовою. Нехай хтось, маючи певну грошову суму, відкрив рахунок у банку. Банк щорічно нараховує певний відсоток від внеску (це називається «дисконтною ставкою відсотка»), відповідно збільшується й сума внеску. Уважається, що цей відсоток не залежить від часу й від величини внеску. Така схема називається «правилом складних відсотків». Необхідно написати програму, що розраховує величину внеску й виводить цю величину для кожного року доти, поки величина внеску не подвоїться. Розглянемо алгоритм розв'язання даної задачі:

5. Увести первісну величину внеску, дисконтну ставку відсотка й рік, коли гроші поклали в банк.
6. Розрахувати нову величину внеску.
7. Вивести рік і величину внеску цього року.
8. Повторювати кроки 2 і 3 доти, поки величин внеску не подвоїться.

Текст програми:

```
program rockafeller;
var balance, balance_initial, rate, interest : Real; year : Word;
begin
WriteLn('Уведіть рік внесення грошей у банк '); ReadLn(year) ;
```

```
WriteLn('Уведіть величину внеску');
ReadLn(balance);
WriteLn('Уведіть ставку відсотка (0.0-1.0)'); ReadLn(rate);
balance_initial := balance;
WriteLn('Рік    Внесок');
WriteLn('========== );
repeat
interest := rate * balance;
balance := balance + interest;
Inc(year) ;
WriteLn(year :6,  '  ',  balance)
until   balance  >  2  *  balance_initial;
WriteLn('Натисніть <Enter>');
ReadLn;
end.
```

В даному випадку тіло циклу виконується хоча б один раз, тому використовується цикл із післяумовою. Процедура Inc(year) збільшує на одиницю значення змінної year.

3. Гра Баше на 15 предметах. Гра Баше відома у Франції. Правила гри Баше такі. Є 15 однакових предметів (за звичай це дерев'яні палички). У грі беруть участь двоє. Суперники ходять по черзі, за кожний хід граючий може взяти 1, 2 або 3 предмети. Програє той, хто вимушений взяти останній предмет. Пропускати хід неможна. Припускаючи, що нашим суперником по грі Баше буде комп'ютер, напишемо для нього програму.

Насамперед придумаємо алгоритм виграшної стратегії, при якій перший суперник починає й виграє. Очевидно, що дані 15 предметів можна розбити на 5 груп, що містять не більше ніж по 4 предмети.

При такому розподілі починаючий гравець бере перші 2 предмети, і далі, скільки б не взяв другий гравець (1,2 або 3 предмети), перший буде добирати до 4 предметів так, щоб разом вони за два напівходи вибрали одну групу. Після чотирьох ходів перший гравець залишає суперникові 1 предмет і виграє.

```
program bashe;
var a, b, m : Integer;
begin m := 15; a := 2; m := m - a;
while m > 1 do
begin
Write(' я взяв ' , з);
if a = 1 then
Write(' предмет')
else
Write('  предмета');
WriteLn('  залишилося  ',  m) ;
WriteLn('Суперник,  ваш хід:');
ReadLn(b);
 a  := 4  -  b;
m  := m  -  (a + b) ;
end;
Wri tel_n( 'Залишився  ' ,  m,  '  предмет, Ви програли');
end.
```

Змініть цю програму так, щоб можна було й грати удвох з людиною.

4. У пошуках досконалості. Число називається *досконалим,* якщо воно дорівнює сумі всіх своїх дільників, включаючи 1, наприклад 6=1+2 + 3, 28 =1+2 + + 4 + 7 + 14. Поставимо задачу визначити, чи є задане число досконалим. Алгоритм розв'язання цієї задачі досить простий. Необхідно перебрати всі натуральні числа від 1 до половинки (або і її цілої частини) від числа, що перевіряється, і, якщо остача від ділення розглянутого числа на дане дорівнює нулю, додати чергове значення до суми. Значення, більші половинки числа, що перевіряється, не має змісту розглядати, тому що вони не будуть його дільниками.

```
program  sover;
var a, i , s : Integer;
begin
Write('Уведіть ціле число a:');
ReadLn(a);
s := 0;
for i := 1 to a div 2 do {Підрахунок суми дільників}
```

```
if a mod i = 0 then
begin
s : = s + i ; Write(' + ', i)
end;
if s = a then
Writeln('Число ', a, ' досконале')
 else
Wri teln(' Число ', a, ' не досконале ') ;
end.
```

Домашнє завдання.

Скласти програму, яка б знаходила суму, добуток, середнє арифметичне всіх тризначних цілих чисел

Розв'язування задач на використання вказівки циклу.

5. Розкажіть про організацію циклів: з передумовою, з післяумовою, з параметром;
6. Що має містити алгоритм циклічної структури?
7. Порівняйте можливості трьох типів циклів.
8. Які правила слід виконувати при використанні циклу з параметром?

Приклад 1. Складіть програму знаходження кількості всіх тризначних чисел, які діляться на кожну зі своїх цифр.

```
Program Variant_1a;
Var i,  j,  L:  byte; c,  k:  word;
Begin k:=0;
For i:  = 1  to  9 do
For j:  s 1 to 9 do
For L:  = 1  to  9 do
Begin
c:  =  100*i+10*j+L;
if (c mod i=0) and (c mod j=0) and (c mod L=0)
then begin
k:=k+1;
write ('c, ' ') ;
end
end
writeln;
wrieln ('к =', к) ;
 end.
```

У другому варіанті розв'язання ми замість генерації числа будемо перебирати всі трицифрові числа від 111 (всі попередні обов'язково мали хоча б один 0 у своєму складі) до 999.

```
Program Variant_lb;
Var k, i: word; c, d, c: byte;
Begin
k:=0; i: =111;
while 1 < 999 do
Begin
c: = i mod 10; {цифра одиниць}
d: =i div 10 mod 10; {цифра десятків}
e: = i div 10 mod 10; {цифра сотень}
if (i<>0) and (d<>0) and (e<>0)
then if (i mod c=0) and (i mod d=0) and (i mod e=0) then begin
k:==k+1;
write  ('i,  ' ');
end
i: =i+1 ;
end;
writeln;
wrieln  ('k =',  k) ;
End.
```

Наведемо ще один варіант розв'язання з використанням циклу «повторювати».

```
Program Variant_Iv;
Var
k, i: word; c, d, c: byte;
Begin
k:=0;
i: = 111;
repeat
c:=i div 100; {цифра сотень}
d:=i div 10 - c *10; {цифра десятків}
e:=i - c*100 - d*10; {цифра одиниць }
if (c<>0) and (d<>0) and (e<>0)
and (i mod c=0) and (i mod d=0) and (i mod e=0)
then begin
k:=k+1;
write ('i, ' ');
end
i:=i+1; until i > 999; writeln;
wrieln ('k=', k) ;
End.
```

При визначенні подільності чисел націло ми користувалися функціями int (x) , div (a, b) *і операцією залишок від ділення* **mod.** *Значенням функції* **int** (x), *є найближче ціле число, що не перевищує задане. Функція* **div** (a, b) *видає цілу частину від ділення двох цілих чисел.* Функцію **div** (a, b) *надалі запишемо як операцію за аналогією з* ***функцією mod.***

Домашнє завдання.

1. Складіть алгоритм знаходження суми цифр усіх тризначних чисел, використовуючи розглянуті команди організації циклу.
2. Складіть алгоритм, що визначає, у скільки разів сума тризначних чисел менша від суми двозначних чисел.
3. Складіть алгоритм, що визначає суму парних цифр усіх двозначних чисел.
4. Складіть алгоритм, що визначає кількість «щасливих» квитків серед шестизначних чисел. (Число називають «щасливим», якщо сума перших трьох цифр дорівнює сумі наступних трьох цифр.)

Вкладені цикли.

У деяких випадках важливо повторити підзадачу кілька разів усередині більш загальної задачі. Один зі способів написання такої програми - включити цикл у набір інструкцій, що повторюються всередині іншого циклу. Така структура, що складається з циклу в циклі, називається вкладеними циклами.

Якщо в програмному коді використовуються вкладені цикли, то для підвищення наочності коду прийнято кожний наступний рівень вкладення зміщувати відносно попереднього.

Правило вкладення циклів: внутрішній цикл цілком укладається в тіло зовнішнього циклу

Приклад. Обчислення значення змінної *Y = 2*K+N* при всіх значеннях змінних *N*=1,2,3 і *K*= 2,4, 6, 8.

Якщо перебирати всі значення *N* і *K,* ми повинні отримати 12 значень змінної *Y.*

Скласти програму можна в такий спосіб: для кожного значення *N* перебрати всі значення *K* від 2 до 8, тобто *N* використати як параметр зовнішнього циклу, *K* - як параметр внутрішнього циклу. Фрагмент обчислення У має вигляд:

```
Program Prim;
 var N, к, Y : integer;
 Begin
For N:=1 то 3 Do  Begin
K:=2;
While K<=8 Do Begin
Y:=2*K+N;
WriteLn (N:3, K:3, Y:3);
K:=K+2
End;
End;
End.
```

Параметр *N* змінюється з кроком 1, тому зовнішній цикл організований з використанням оператора For; параметр *K* змінюється з кроком 2, тому внутрішній цикл є циклом While.

Створення та реалізація програми з вкладеним циклом

(Самостійна робота)

Задача. Вивести на екран всі прості числа на інтервалі *[A; B].*

Простими є числа (крім 1), які діляться без остачі тільки на 1 і на самого себе. Розглянемо алгоритм перевірки, чи є число С простим:

- Вводиться змінна Prap, яка відіграє роль прапорця (за її значенням можна визначити, чи відбулась деяка подія). Змінній Prap присвоюється початкове значення False.
- Перебираються всі числа від 2 до (С div 2). Якщо число і є дільником числа С (тобто С ділиться на і без остачі), то прапорцевій змінній Prap присвоюється значення True.
- Після завершення перебору можливих дільників перевіряється значення змінної Prap. Якщо значення Prap в процесі повторень циклу змінилось на True, то це означає, що С має хоча б один дільник і не є простим.

2. Запишемо програму для визначення, чи є число *С* простим. Зверніть увагу: задання значення змінної С навмисно пропущене.

```
var A, B, C, I,: integer;   Prap : Boolean;
Begin { задання значення змінної С }
 Prap := False; For I := 2 To C div 2 Do
if C Mod I = 0 Then Prap := True;
 If Not Prap    Then WriteLn (c,  ' - просте число')
Else WriteLn (C, ' - складене число');
End.
```

5. За умовою задачі потрібно перебрати всі числа в інтервалі *[A; B],* тобто С послідовно набуває значень з діапазону *A..B.* Для кожного С від *A* до *B* потрібно виконати алгоритм перевірки, чи є число С простим.

6. Внесемо зміни до програми. Значення *A* і *B* вводяться з клавіатури:

```
var A, в, с, i : Integer; Prap : Boolean;
Begin
 write ('A, B =>'); ReadLn (A, B);
```

5. Додаємо керуючий рядок циклу *For,* тілом циклу якого є програмний код перевірки, чи є число С простим:

```
For C := A to B Do Begin
Prap := False;
For I := 2 To C div 2 Do
If C Mod I = 0 Then Prap := True;
if Not Prap
Then WriteLn (C, ' - просте число');
End;
```

6. Збережіть файл. Виконайте програму для різних значень *A* і *B.*

Домашнє завдання.

Дайте відповіді на запитання:

5. Чи може оператор, повторюваний у циклі, сам бути циклом?

6. У чому полягає правило вкладення циклів?

7. Наведіть приклади задач, при розв'язуванні яких виникає необхідність використання вкладених циклів.

Задача. *Скласти програму, яка друкує всі 3-цифрові натуральні числа, сума цифр яких дорівнює їхньому добутку і визначає кількість таких чисел. Для розв'язання поставленої задачі зовсім не обов'язково перебирати всі натуральні числа від 100 до 999, тому що нас цікавить не саме число, а цифри в його десятковому записі. Тому можна перебирати всі можливі сполучення цифр, з яких складається десятковий запис трицифрового числа і перевіряти для кожного сполучення умову задачі. Створіть новий файл. Запишіть програму і виконайте її. Проаналізуйте результат роботи програми.*

```
var A, B, C, K : integer;
Begin
K := 0;
For A := 1 To 9 Do
For B := 0 то 9 Do
For C := 0 To 9 Do
Lf A + B + C = A* B * C  Then
  Begin
WriteLn (100*A+10*B+ C);
K := к + 1;
```

```
End;
WriteLn ('K=', K);
End.
```

5) Внесіть зміни у програму з тим, щоб знайти кількість усіх трицифрових натуральних чисел, сума цифр яких дорівнює даному цілому числу. Виконайте програму, проаналізуйте результат.

6) Сума цифр двоцифрового числа дорівнює 11. Якщо до цього числа додати 27, то вийде число, записане тими ж цифрами, але в зворотному порядку. Знайдіть це число.

Ідея розв'язування та ж сама, що й у п.1: перебираємо всі можливі сполучення цифр A і B в десятковому записі двоцифрового числа AB, і для кожного сполучення перевіряємо умову задачі. Умову задачі можна записати у вигляді логічного виразу:

(10*A+B)+27 = 10*B+A.

Після внесення змін програма має такий вигляд:

```
Var A, B : Integer;
Begin
For A := 1 To 9 Do
For B := 0 To 9 Do
If (10*A+B)+27 = 10*B+A Then WriteLn (10*A+B);
End.
```

5) Сума квадратів цифр двоцифрового числа на 1 більша від потроєного добутку цих цифр. Після ділення цього двоцифрового числа на суму його цифр у частці виходить 7 і в остачі 6. Знайдіть це число.

Умову задачі можна записати у вигляді логічного виразу: (A*A+B*B=3*A*B+1) And ((10*A+B) div (A+B)=7) And ((10*A+B) mod (A+B)=6) *або рівносильного виразу:*

(A*A+B*B=3*A*B+1) And (10*A+B = (A+B)*7+6).

6) Внесіть зміни у програму, виконайте її, перевірте, чи отримали ви правильний результат.

Розглянемо приклад, де доводиться використовувати два цикли, один усередині іншого.

***Приклад** 1.* Скласти програму виведення всіх натуральних чисел, менших за *n,* квадрат суми цифр яких дорівнює заданому числу m.

Ідея розв'язання. Вводиться натуральне число, до якого треба виводити всі натуральні числа, що задовольняють задану умову. Нехай, наприклад, користувач уведе число — 21.

Друге число, яке треба ввести користувачу, — це число, якому дорівнює квадрат суми цифр натуральних чисел.

Зрозуміло, що це число має бути точним квадратом, воно може дорівнювати: 4, 9, 16, 25, 36 і т.д.

Припустимо, що користувач увів число 4.

Треба знайти всі натуральні числа від 1 до 21, квадрат суми цифр яких дорівнює 4. Починаємо з чисел: 1, 2, 3, 4, 5,..., 21, вибирати ті, які задовольняють задану умову.

Перше з них 2, тому що $2^2 = 4$, друге 11, тому що $(1 + 1)^2 = 2^2 = 4$, третє 20, тому що $(2 + 0)^2 = 2^2 = 4$.

Інших натуральних чисел до 21, що задовольняють задану умову, *немає.*

Усі відібрані числа треба вивести на екран: 2, 11 і 20.

Алгоритм 1. Розділ описів

Змінні: *n, m, k, a, p, s.* Тип цілий.

n — для границі значень натуральних чисел, *m* — для числа, з яким порівнюється квадрат суми цифр (точний квадрат), *k* — для організації перебору натуральних чисел від 1 до *n,* ***a*** — для тимчасового зберігання натурального числа, перед тим, як буде визначатися сума його цифр, *p* — для зберігання виділеної цифри числа, s — для зберігання суми цифр числа.

2. **Розділ операторів**

Уведення значень *n і m.* Установити початкове значення для *k* (ця змінна *«перебирає»* усі натуральні числа від 1 до *n, k* = 1).

Цикл, *поки* k <= n.

У циклі: встановити початкові значення для суми s (s:=0); запам'ятати число в змінну *a* (a:= к), оскільки після знаходження суми його цифр саме число буде зіпсовано.

Цикл для підрахунку суми цифр, *поки* k <> 0.

У циклі: виділяти по одній цифрі числа відомим способом; накопичувати суму цифр; зменшувати число на останню цифру (відкидати її цілочисельним діленням на 10).

Закінчити цикл для підрахунку суми цифр.

Перевірка виконання умови, тобто: ***якщо*** квадрат суми цифр дорівнює заданому числу, ***тоді*** вивести це натуральне число на екран.

Перейти до перевірки наступного числа.

Закінчити основний цикл перевірки чисел.

8. Закінчити програму.

Отже, маємо програму:

```
Program Problem; uses WinCrt; var
n,m,k,a,p,s: integer;
Flag: boolean;
 begin
write ('Уведіть натуральне число, до якого ');
write('виводити шукані числа ');
readln(n);
writeln('Уведіть число, з яким порівнюєте квадрат');
write('його суми цифр. Воно має бути точн. квадрат. ');
readln(m);
write('Шукані числа: ') ;
k:= 1;  Flag:=False;
while  к  <= n  do
begin
s: = 0;  a: = к;
while к <> 0 do
begin
p:= к mod 10;  s: = s  + p; k:= к div 10 end;
if  sqr(s)  = m then begin
write(a,   '  ') ;
Flag:=True;    end;
к  := a +1
end
if NOT Flag
then writeln('Таких чисел для заданих вхідних даних не  існує')
end.
```

У програмі два цикли. Один — зовнішній, для натуральних чисел, другий — внутрішній, для підрахунку суми цифр числа.

Зверніть увагу, що у випадку некоректних вхідних даних (***n*** — дуже мале, або ***m*** — не є точним квадратом числа) програма не може видати результат. Щоб запобігти цьому, пропоную використання змінної логічного типу **Flag**. Для цієї задачі міркування можуть бути такими: на початку програми змінній **Flag** надається значення **Fals e** (числа ще не знайдені). Всередині циклу якщо знайдеться хоч одне число, що задовольняє умову, разом з виведенням його на екран змінній **Flag** присвоюється значення **True**. Таким чином, якщо змінна **Flag** після виконання програми не зміниться, це свідчить про те, що необхідні числа не знайдені. В цьому випадку слід видати про це повідомлення.

5. Практичне завдання

Скласти програму пошуку всіх натуральних чисел $n \le 100\,000$, сума цифр яких дорівнює заданому натуральному числу ***a***.

```
 Program Task;
uses WinCrt;
var a,p,s  :  integer;  n,b:  longint;
begin
write('Уведіть натуральне число ') ;
readln(a);
if a>45 then writeln ('Таких чисел не існує')
else begin
b:= 1;
writeln('Натуральні числа, сума цифр  ');
write('яких дорівнює числу  ',  a,  ' наступні:  ');
while b < 100000 do
begin
s  :=  0;  n  := b;
while n <> 0 do
begin
p:= n mod 10;  s:= s  +p; n:= n div 10 end;
if s = a  then write(b,  ',  ') ;
```

```
b  := b +1
end;
end;
end.
```

Налагодити програму і написати алгоритм етапів розв'язання.

Вправи для роботи на закріплення

7. Знайти всі тризначні числа, при цілочисельному діленні кожного з яких на 11 отримуємо частку, що дорівнює сумі квадратів значень окремих цифр числа.
8. Тризначне десяткове число закінчується цифрою 3. Якщо цю цифру перемістити через два знаки ліворуч, тобто з цієї цифри буде починатися запис нового числа, то це нове число буде на одиницю більше від потроєного вихідного числа. Знайдіть це число.
9. Знайдіть усі тризначні числа, що дорівнюють сумі кубів своїх цифр.
10. Шестизначне десяткове число починається ліворуч цифрою 1. Якщо цю цифру перенести зі свого місця на останнє місце праворуч, то значення утвореного числа буде втроє більшим від вихідного. Знайдіть вихідне число.
11. Дано ціле число *n* > 10 Написати програму одержання ***m*** останніх цифр десяткового запису числа *n.*
12. Знайти чотиризначне число, що дорівнює квадрату числа, вираженого двома останніми цифрами цього чотиризначного числа.

Домашнє завдання.

3. Знайти чотиризначні числа, кожне з яких ділиться націло на 11 і сума цифр яких дорівнює 11.
4. Знайти чотиризначні числа, які, якщо приписати праворуч до числа 400, дають повний квадрат.

54. Випадки застосування циклів з післяумовою.

Відмінності між циклом while і циклом repeat:

1. Оператори, що знаходяться в циклі **while,** повторюються доти, доки умова істинна. Послідовність операторів, що знаходяться в циклі **repeat,** повторюється доти, доки умова хибна.

Отже, у циклі **while** використовується умова продовження циклу, а в циклі **repeat** — умова закінчення циклу.

8. У циклі **while** повторюється один оператор (кілька операторів треба поєднувати в складений оператор за допомогою операторних дужок **begin... end),** а в циклі **repeat** можна використовувати кілька операторів без операторних дужок.

9. У циклі **while** спочатку перевіряється умова, а після цього залежно від значення умови (якщо істинна) оператор чи група операторів після слова do.

10.У циклі **repeat** послідовність операторів виконується один раз, а після цього перевіряється умова, тобто ця послідовність *завжди* виконується *хоча б один* раз, а в циклі **while** оператори, що складають тіло циклу, можуть узагалі не виконуватися жодного разу (у випадку, якщо умова відразу хибна).

Розглянемо приклади.

Приклад 1. Знайти найменше натуральне число, що дає при діленні націло на 2, 3, 4, 5,6 відповідно остачу 1, 2, 3, 4, 5.

Розв'язання. Береться найменше натуральне число — одиниця — і знаходяться остачі від ділення його на 2, 3, 4, 5 і 6; якщо залишки дорівнюватимуть 1, 2, 3, 4 і 5, тоді це число є шуканим, його треба видати на екран і закінчити програму, у протилежному випадку треба брати наступне натуральне число 2 і перевіряти його, і так далі.

Отже, маємо програму:

```
Program Problem1;
uses Crt;
var n : integer;
begin
n := 0;
repeat
n : -  n + 1 ;
until (n mod 2=1) and (n mod 3=2) and (n mod 4 = 3) and (n mod 5=4) and (n mod 6=5);
writeln('Шукане ціле число ', n)
end.
```

Ще один приклад, що демонструє роботу циклу з післяумовою.

Приклад 2. Числа, що однаково читаються зліва направо, і справа наліво, називаються паліндромами. Наприклад, числа 42324 чи 1331 — паліндроми. Складіть програму, що буде знаходити числа-паліндроми із заданого проміжку.

Розв'язання. Перевернути число і порівняти отримане число із заданим.

Нехай дано число *a,* уведемо ще одну змінну *b,* якій буде присвоєно значення змінної *a* (для чого це робиться, ви довідаєтеся пізніше): b:=a;

Заведемо ще одну змінну a_1 для нового числа, у якому цифри вже будуть переставлені.
Початкове значення цієї змінної — нуль: a1:=0;
Чому значення цієї змінної дорівнює нулю, стане ясно з програми.
Далі організуємо цикл repeat, у якому відбуватиметься перестановка цифр числа *b:* repeat
a1:= a1*10 +b mod 10;
b:= b div 10 until b = 0;
Отже, у циклі, також як і в циклі while... do..., виділяється остання цифра:
b mod 10; (наприклад, 343 mod 10 = 3); змінній a_1 надається значення: a1:= a1*10 +b mod 10; 0 * 10 +3 =3;
«відкидається» остання цифра заданого числа за допомогою операції цілочисельного ділення: b:= b div 10; 343 div 10 = 34;
перевіряється умова: *b* = 0, 34 = 0, умова хибна, значить цикл продовжується.
Виділяється остання цифра вже нового числа:
b mod 10 = 34; 34 mod 10 = 4;
число a_1 уже рівне 3, збільшується у 10 разів і до результату додається наступна цифра 4:
a1:= a1*10 +b mod 10;
3 * 10 +4 = 34;
«відкидається» остання цифра числа b:
b:= b div 10;
34 div 10 = 3;
перевіряється умова: b = 0,3 = 0; умова хибна, значить цикл продовжується.
Виділяється остання цифра числа: b mod 10; 3 mod 10 = 3; формується нове число: a1:= a1*10+b mod 10; 34*10+3=343;
«відкидається» остання цифра числа і виходить нове число: b:= b div 10 ; 3 div 10 = 0;
перевіряється умова: b = 0, 0 = 0; умова істинна, значить цикл закінчує свою роботу.
Тепер стає ясно, чому введена інша змінна b для заданого числа: її значення в циклі змінюється від початкового до нуля і, щоб зберегти початкове число в змінній a, і вводиться так звана, «робоча» змінна b.
Після закінчення циклу перевертання числа порівнюється початкове число, що «зберіглося у змінній a, і число, що вийшло після перестановки цифр і «накопичилося» у змінній a_1 .
Якщо a = a_1 , тоді значення a видається на екран, оскільки це число є *паліндромом.*
Далі, значення a збільшується на 1, тобто береться для розгляду наступне натуральне число і знову продовжується зовнішній цикл. Число перевертається, отримане нове число — a_1 порівнюється з початковим a і так далі.
Зовнішній цикл закінчується, коли значення o стає рівним правій границі інтервалу — *n.*
Отже, маємо програму:

```
Program Problem2;
uses Crt;
var m, n, a, b, al : longint;
begin
write('Уведіть ліву границю проміжку ');
readln(m) ;
write('Уведіть праву границю проміжку '); readln(n);
 a := m;
writeln('Числа паліндроми з [', m, ';', n, ']');
repeat
b := a; a1 := 0;
repeat a1 := a1*10 +b mod 10; b := b div 10
until b=0;
if al = a then write(a, ' ') ;
a := a + 1
until a > n;
end.
```

Програми, складені з циклом з передумовою (while... do...), легко можна переробити на програми з циклом з післяумовою (repeat... until...):
Отже, маємо програми
1. Для підрахунку суми цифр числа:

```
Program Sum; { Сума цифр числа }
uses Crt;
var n, s, a : longint;
begin
write('Уведіть ціле число ') ;
```

```
readln(n);
a  := n;  s  := 0;
repeat
s := s +n mod 10; n := n div 10
until n = 0;
writeln('Сума цифр числа ', a, ' дорівнює ', s)
end.
```

2. Для перестановки першої й останньої цифр у числі:

```
Program Transpose;
uses Crt;
var n, n1, p, a, i: longint;
begin
write('Уведіть натуральне число'); readln(n);
a:= n; i:= 1; p:= n mod 10;
{остання цифра введеного числа}
repeat
i:= i*10;  n:= n div 10
until n<10;
n1:= a  -  n*i  - p +n  +p*i;
writeln('Число після перестановки цифр  ',  n1)
end.
```

Домашнє завдання

5 *Поясніть роботу вкладених циклів.*
6 *За яким принципом організовуються вкладені цикли ?*
7 *Чи можна два вкладених цикли замінити двома послідовними циклами? Що при цьому зміниться?*
8 *Чи може бути, на ваш погляд, більше, ніж два вкладених цикли?*

Знайти найменше натуральне число, кратне 131, з парною кількістю цифр. Скласти програму.

Підсумки. Задачі для самостійного розвязання.

Підсумки курсу

3. Розглянуто поняття алгоритму.
4. Визначено форми подання алгоритмів:
- *словесні;*
- *словесно-формульні;*
- *формульні;*
- *графічні (у вигляді блок-схем).*

3. Визначено типи алгоритмів:
- *лінійні;*
- *розгалужені;*
- *циклічні;*
- *змішані.*

4. Визначено властивості алгоритму:
- *дискретність;*
- *скінченність;*
- *зрозумілість;*
- *визначеність;*
- *масовість.*

8. З'ясовано поняття формального виконання алгоритму.
9. Дано визначення алгоритмічної мови:
10. Визначено поняття:
- *алфавіт мови;*
- *синтаксис мови;*
- *елементи мови;*
- *символи;*
- *службові слова;*
- *команди;*
- *об'єкти мови:*
- *константи, змінні (числові, символьні, масиви);*
- *допоміжні алгоритми (підпрограми - функції або процедури);*

- *вирази.*

10.Введено типізацію величин і пояснено її необхідність.

11.Введення поняття « відношення між величинами ».

12. Визначено форму запису словесного алгоритму й алгоритму з використанням величин.

13. Розглянуто основні конструкції алгоритмічної мови:

- *присвоювання;*
- *розгалуження;*
- *повторення.*

Задачі.

Скласти і виконати програму, задавши вхідні дані самостійно.

24. Квіткова клумба має форму круга. Обчислити її периметр і площу за заданим радіусом.

25. Обчислити периметр і площу прямокутного трикутника за заданим катетом та гострим кутом.

26. Обчислити довжину кола і площу круга за заданим діаметром.

27. Ділянка лісу має форму рівнобічної трапеції. Обчислити її периметр і площу за заданими сторонами.

28. Ресторан закуповує щодня масло m_1 кг по 8.50 грн. за кілограм, сметану $т_2$ кг по 2.40 грн., вершки $т_3$ кг по 4.10 грн. Визначити суми, потрібні для купівлі окремих продуктів, і загальну суму.

29. Скільки секунд мають доба, тиждень, рік?

30. Обчислити кінетичну $E=mv^2/2$ та потенціальну $P=mgh$ енергії тіла заданої маси $т$, яке рухається на висоті h зі швидкістю v.

31. Ціни на два види товарів зросли на р відсотків. Вивести старі та нові ціни.

32. Обчислити площу поверхні $S=4\pi r^2$ та об'єм $V=4\pi r^3/3$ сфери за заданим радіусом r.

33. Швидкість світла 299792 км/с. Яку відстань долає світло за годину, добу?

34. Увести врожайність трьох сортів пшениці (36, 40, 44 т/га) і розміри трьох відповідних полів (у га). Скільки зібрали пшениці з кожного поля і з трьох полів разом?

35. Радіус Місяця 1740 км. Обчислити площу поверхні $S=4\pi r^2$ та об'єм планети $V=(4/3)\pi r^3$.

36. Обчислити довжину гіпотенузи та площу прямокутного трикутника за заданими двома катетами.

37. Обчислити об'єм та площу бічної поверхні куба, якщо відоме ребро.

38. Увести продуктивності роботи трьох труб, які наповнюють басейн, і час їхньої роботи. Скільки води набрано в басейні?

39. Яку площу і периметр матиме квадрат, описаний навколо круга заданої площі S.

40. Тіло падає з прискоренням g. Визначити пройдений тілом шлях $h=gt^2/2$ після першої та другої секунд падіння.

41. Обчислити периметр і площу прямокутного трикутника за заданими катетами.

42. Телефонні розмови з трьома населеними пунктами коштують c_1, c_2, c_3 коп/хв. Розмови тривали t_1, t_2, t_3 хв відповідно. Яку суму нарахує комп'ютер до оплати за кожну і всі розмови?

43. Обчислити площу бічної поверхні $S=2\pi rh$ та об'єм $V=\pi r^2h$ діжки за заданою висотою h та радіусом основи r.

44. Квіткова клумба має форму квадрата. Обчислити її периметр і площу за заданою стороною.

45. Обчислити катет та площу прямокутного трикутника за заданими гіпотенузою та другим катетом.

46. Обчислити сторону та площу $S=d^2/2$ квадрата, якщо відома його діагональ d.

24.Обчислити площу бічної поверхні $S=\pi rl$ та об'єм $V=\pi r^2h/3$ конуса за заданою висотою h, твірною l та радіусом основи r.

25.Поїзд їхав t_1 год зі швидкістю v_1 км/год, t_2 год зі швидкість v_2 і t_3 год зі швидкістю v_3. Визначити пройдені шляхи з різною швидкістю і повний шлях.

Розгалуження.

11. Задано довжини сторін трикутника. Складіть програму, за допомогою якої визначте, чи є трикутник рівнобедреним, рівностороннім або різностороннім.

12. Задано три числа *a, b, c*. Складіть програму, за допомогою якої визначте, чи існує трикутник з такими довжинами сторін. Якщо так, то знайдіть його периметр.

13. Задано три числа *a, b, c*. Складіть програму, за допомогою якої визначте, чи існує трикутник з такими довжинами сторін. Якщо так, то знайдіть його площу.

14. Задано довжини суміжних сторін паралелограма й кут між ними. Складіть програму, за допомогою якої визначте, чи є цей паралелограм квадратом, ромбом, прямокутником або не є ні однієї з перерахованих фігур.

15. Задано сторону рівностороннього трикутника й радіус кола. Складіть програму, за допомогою якої визначте, що більше - площа трикутника або площа кругу.

16. Задано сторону рівностороннього трикутника й сторона квадрата. Складіть програму, за допомогою якої визначте, що більше - площа трикутника або площа квадрата.

17. Задано сторону рівностороннього трикутника й сторона квадрата. Складіть програму, за допомогою якої визначте, що більше - периметр трикутника або периметр квадрата.

18. Задано сторону куба й радіус кулі. Складіть програму, за допомогою якої визначите, що більше - об'єм куба або об'єм кулі.

19. Задано виміри прямокутного паралелепіпеда й діаметр кулі. Складіть програму, за допомогою якої визначите, що більше - об'єм паралелепіпеда або об'єм кулі.

20. Задано сторону квадрата й радіус кола. Складіть програму, за допомогою якої визначите, що більше - площа квадрата або площа кругу.

Вибір. Скласти програму для розв'язування наведеного нижче завдання двома способами, використовуючи: 1) команду case; 2) команду if. Придумати і задати вхідні дані так, щоб вибір був з 4-7 альтернатив.

1. Ввести номер учня зі списку. Вивести його прізвище.
2. Є дані про автомобілі чотирьох моделей. Як вхідне дане ввести номер моделі і отримати характеристики: рік випуску і ціну.
10. Ввести номер поїзда. Вивести назву пункту призначення.
11. Ввести першу букву назви країни. Вивести назву її столиці.
12. Ввести номер дня тижня. Вивести його назву.
13. Ввести номер трамвая. Вивести назви його кінцевих зупинок.
14. Ввести першу букву назви країни. Вивести назву континента.
15. Ввести номер місяця. Вивести назву пори року.
16. Ввести номер учня у списку. Вивести його ім'я.

Цикли.

1. Складіть програму, за допомогою якої знайдіть добуток:
   а) перших 500 натуральних чисел;
   б) тризначних чисел, кратних 11.
2. Складіть програму, за допомогою якої знайдіть суму:
а) непарних тризначних чисел;
б) парних п'ятизначних чисел;
в) парних чотиризначних чисел;
г) перших 500 натуральних чисел, кратних 3;
д) перших 300 натуральних чисел, кратних 15;
е) тризначних чисел, кратних 7;
ж) чотиризначних чисел, кратних 3.
3. Складіть програму, за допомогою якої знайдіть суму 3 + 9 + 27 + 81 + ... + 19683.
4. Знайдіть суму та середнє арифметичне 10 довільних чисел, які вводяться з клавіатури.
5. З клавіатури вводяться 20 довільних чисел. Знайдіть суму тільки від'ємних доданків.
6. Складіть програму для знаходження суми чисел, кратних 7, які розташовані в інтервалі (100, 400).
7. Вводиться послідовність із N цілих чисел. Знайти:
А) найбільше число;
Б) найменше число;
В) суму всіх додатніх чисел;
Г) кількість нулів;
Д) суму всіх відємних чисел.
8. Протабулюйте функцію y=xsin4xcos2x на проміжку [-4;4] з кроком 0,2 і обчисліть суму квадратів максимального і мінімального значень функції.
9. Протабулюйте функцію y=sin5xcosx на проміжку [-2;2] з кроком 0,2 і обчисліть середнє арифметичне від'ємних значень.
10. Протабулюйте функцію z=cos(2x-3y), змінюючи x на проміжку [0;2] з кроком 0,25, а y на проміжку [0;1] з кроком 0,2. обчисліть середнє арифметичне додатних значень функції.

Використана література:

1. Янковой В.А., Крыжевская А.Н. Язык программирования Turbo Pascal 7.0: в помощь учителю информатики. – Одесса:ОНМЦ ОКИТ, 2002. 132с.
2. Я.М. Глинський, В.Є. Анохін, В.А. Ряжська Паскаль Turbo Pascal і Delphi: Навчальний посібник. – Львів, 2006.190с.
3. Бондаренко О.О. Інформатика Turbo Pascal Спецкурс. - Шепетівка: «Аспект», 2007. 269с.
4. Я.М. Глинський. Інформатика. Алгоритмізація і програмування: Навчальний посібник для загальноосвітніх навчальних закладів. – Львів, 2003. 198с.
5. Т.П. Караванова. Інформатика. Основи алгоритмізації та програмування: 777 задач з рекомендаціями та прикладами. – Київ:»Генеза», 2006. 285с.
6. Т.П. Караванова. Інформатика. Основи алгоритмізації та програмування (процедурне програмування): Навчальний посібник. - Шепетівка: «Аспект», 2007. 192с.
7. Бондаренко О.О., Мірошниченко А.А. Інформатика. Основи програмування мовою Паскаль для 8 – 9 класів. - Шепетівка: «Аспект», 2004. 94с.
8. С. Немнюгин, Л. Пектолаб Изучаем Turbo Pascal – «Питер», 2008. 310с.
9. Л.А. Куценкова. Сборник задач по информатике. – Минск:УП «Технопринт», 2003. 138с.
10. Н.Б. Культин. Программирование в Turbo Pascal 7.0 и Delphi. – Санкт-Петербург: «BHV – Санкт-Петербург», 1998. 232с.