Житомирський державний педагогічний університет імені Івана Франка

Кафедра математики та інформатики

О.М. Спірін, О.М. Кривонос

ПОЧАТКИ АЛГОРИТМІЗАЦІЇ ТА ПРОЦЕДУРНОГО ПРОГРАМУВАННЯ

Методичний посібник
для студентів вищих педагогічних навчальних закладів
фізико-математичних спеціальностей

Житомир - 2002

УДК 681.142.2(075.5)
ББК 32.973.2р+73я73
С72

Затверджено вченою радою Житомирського педуніверситету,
протокол № 8 від 25.02.2002р.

Рецензенти:
кандидат фіз.-мат. наук, доцент Лящено Б.М.
методист лабораторії інформатики та обчислювальної техніки
Житомирського ОІППО Лисогор І.А.

Спірін О.М., Кривонос О.М.

С72 Початки алгоритмізації та процедурного програмування: Методичний посібник для студ. вищих пед. навч. закл-ів фізико-математичних спец-тей. – Житомир: ЖДПУ, 2002.- 93 с., іл.

ISBN 966-7603-50-4

Посібник призначений для використання студентами під керівництвом викладача на лекціях, практичних заняттях та лабораторних роботах. Посібник містить теоретичні відомості з початків алгоритмізації, процедурного програмування на мовах MSX-Basic, Pascal; приклади виконання завдань, тексти лабораторних робіт (контрольні питання, завдання, вимоги до захисту лабораторних робіт). Викладений матеріал відповідає діючій програмі з основ інформатики і обчислювальної техніки для спеціальностей "математика і фізика", "фізика і математика" у частині, що стосується розгляду питань алгоритмізації та процедурного програмування.

Для студентів фізико-математичних спеціальностей вищих педагогічних навчальних закладів, вчителів інформатики загальноосвітніх шкіл.

УДК 681.142.2(075.5)
ББК 32.973.2р+73я73

ISBN 966-7603-50-4

ЗМІСТ

ПОЧАТКИ АЛГОРИТМІЗАЦІЇ

1. ЗАГАЛЬНІ ВІДОМОСТІ

1.1. Етапи розв'язання задач з використанням ЕОМ

Процес розв'язку більшості задач з використанням ЕОМ можна розбити на декілька характерних етапів: постановка задачі, описання алгоритму, запис і трансляція програми, налагодження програми, експлуатація програми.

І етап. Постановка задачі

Для того, щоб розв'язати задачу, пов'язану з дослідженням реального об'єкта, необхідно спочатку описати цей об'єкт у математичних термінах, тобто побудувати його *математичну модель*. Математична модель дозволяє звести розв'язування реальної задачі до розв'язування задачі математичної. У найпростішому випадку для задач на обчислення математична модель являє собою математичну формулу.

ІІ етап. Описання алгоритму

Після того, як математична модель задачі побудована, необхідно знайти та описати спосіб розв'язку цієї задачі. У найпростішому випадку розв'язок задачі можна одержати у явному вигляді, тобто у вигляді формули, яка пов'язує вхідні дані та результати. Але і тоді, крім знаходження відповідної формули, необхідно вирішити ряд інших технологічних питань, враховуючи те, що задача буде розв'язуватися на ЕОМ. Наприклад, якого типу мають бути вхідні дані, у якому порядку їх вводити до ЕОМ, як і з якою точністю одержати проміжні величини, у якій формі і з якою точністю вивести результати тощо. Тому описання алгоритму – наступний етап розв'язку задачі.

Побудова алгоритму носить творчий характер – кожна нова задача вимагає нових підходів і нових способів розв'язку, але для того, щоб побудувати алгоритм, який можна легко зрозуміти, модифікувати й удосконалити, необхідно дотримуватися певної дисципліни та технології його конструювання.

Зрозуміло, що той, хто описує алгоритм, насамперед сам повинен вміти і знати як розв'язується задача.

III етап. Запис і трансляція програми

Якщо алгоритм складено, з погляду на розв'язок задачі на ЕОМ користувачу залишається більш технічна, аніж творча робота. Враховуючи характер задачі необхідно вибрати певну мову програмування для запису програми та записати її текст. Зрозуміло, що користувач повинен володіти потрібним обсягом знань з використання команд (операторів) обраної мови.

Кожна мова програмування має свій транслятор, що дозволяє автоматично перекласти текст програми на машинну мову. При цьому одержується текст програми на машинній мові, який еквівалентний тексту програми на мові програмування, тому що ці тексти реалізують один і той же алгоритм.

IV етап. Налагодження програми

На цьому етапі виявляються можливі помилки, які допущені на попередніх етапах. Синтаксичні помилки у тексті програми автоматично виявляються ще на етапі трансляції і користувач вносить відповідні зміни до тексту. Може статися так, що текст програми записано вірно, а помилка допущена при складанні алгоритму - програма працює, але видає неправильні результати. Якщо ж помилку допущено на етапі постановки задачі, то програма працює правильно, але розв'язує іншу задачу.

Отже, *метою налагодження* є одержання правильної програми, результатам роботи якої можна було б довіряти.

Суть налагодження полягає у тому, що користувач розробляє систему тестів, за допомогою якої перевіряється робота програми у різних можливих режимах. Кожен тест має набір вхідних даних, для яких відомий результат. Тест намагаються вибрати так, щоб не тільки встановити сам факт помилки, але й локалізувати її, тобто виявити та звузити частину програми, що містить помилку.

До складу систем програмування включають спеціальні можливості налагодження програм. Користувач формулює завдання, а система виконує це завдання і видає користувачу необхідну інформацію про те, як веде себе програма. Така інформація значно полегшує пошук та виправлення помилок.

V етап. Експлуатація програми

Якщо розроблена програма розрахована на тривалу експлуатацію, розрахована на розв'язання серйозних задач, то необхідно її супроводжувати. Адже протягом використання програми можуть змінитися, наприклад, вимоги до розв'язуваної задачі, операційна система, на базі якої виконується дана програма тощо. Це може вимагати внесення

змін до програми, а внесені зміни – нового налагодження програми. Такі роботи виконує розробник програми або група супроводу.

1.2. Поняття про алгоритм

Алгоритм – це точне і повне описання послідовності виконання скінчених дій, необхідних для розв'язку задачі даного типу. Слово "алгоритм" виникло від algorithmi – латинської форми написання імені великого математика аль-Хорезмі, який у IX ст. описав правила виконання арифметичних дій над числами у десятковій системі числення.

Основні властивості алгоритмів:

- *дискретність.* Обчислювальний процес, що описується алгоритмом, має бути розбитий на послідовність окремих дій. Таке описання являє собою послідовність чітко відокремлених одна від одної вказівок, що утворюють перервну (дискретну) структуру алгоритмічного процесу – тільки виконавши вимоги однієї вказівки, можна перейти до наступної;
- *детермінованість.* Алгоритм не може містити вказівок, смисл яких сприймається неоднозначно. Крім того, після виконання чергової вказівки, цілком однозначно має бути визначено, яку вказівку потрібно виконувати наступною. Отже, враховуючи однозначність послідовності дій, із яких складається алгоритм, застосування його до одних і тих же вхідних даних має забезпечувати один і той же результат;
- *масовість.* Алгоритм складається не для розв'язку однієї конкретної задачі, а для цілого класу задач даного типу;
- *результативність.* При точному виконанні всіх вказівок алгоритму обчислювальний процес має зупинитися за скінчену кількість кроків і при цьому має бути одержаний певний результат роботи.

Описати алгоритм – означає розбити його на окремі етапи, визначити, яку роботу необхідно виконати на кожному етапі, вказати порядок виконання цих етапів.

Для того, щоб алгоритм був зрозумілий не тільки його автору, а й будь-якому виконавцю, у тому числі й ЕОМ, розроблені загальноприйняті способи описання алгоритмів. Найуживаніші серед них: словесне описання у термінах природної мови, графічне описання та описання у термінах деякої алгоритмічної мови.

Графічне описання алгоритмів

Розглянемо описання алгоритмів у вигляді блок-схем. Блок-схема алгоритму являє собою систему блоків, зв'язаних лініями потоку.

Блоки зображуються у вигляді плоских геометричних фігур: овалу (блок "початок-кінець алгоритму"), паралелограма (блок вводу-

виводу), прямокутника (функціональний блок[1]), ромба (логічний блок), шестигранника (блок підготовки – послідовної зміни параметру), прямокутника з подвійними бічними сторонами (блок виконання типового процесу – підпрограми) тощо.

Лінії потоку вказують на порядок виконання послідовності операцій алгоритму (порядок переходу від блоку до блоку). При цьому напрямок лінії потоку зверху вниз і зліва направо приймається за головний і стрілкою не позначається. Напрямок лінії знизу вверх і справа наліво позначається стрілкою. Місце злиття ліній потоку допускається позначати точкою.

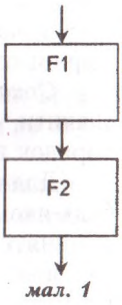
1.3. Структурний підхід до алгоритмізації

Розробка алгоритмів розв'язування задач на ЕОМ, ставить деякі природні вимоги до алгоритмів: алгоритм має бути зрозумілим і легко сприйматися не тільки його розробником, необхідно, щоб алгоритм можна було легко перевірити, а також модифікувати його без суттєвої перебудови всієї структури.

Тому при розробці алгоритмів дотримуються особливої методики, що називається *структурним підходом*. За цим підходом алгоритми, як в конструкторі, "збираються" з трьох базових структур: *слідування*, *розгалуження*, *цикл*, кожна з яких має один вхід і один вихід.

Базова структура "слідування"

Ця структура (мал. 1) складається із двох функціональних блоків F1, F2 і означає, що два функціональних блоки можуть розміщуватися один за одним.



мал. 1

Базова структура "розгалуження"

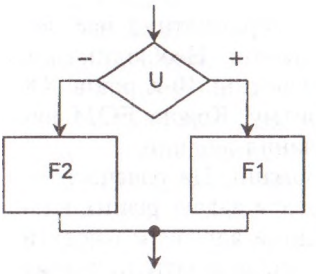
Ця структура (мал. 2) складається з логічного блоку, у якому перевіряється деяка умова U, та функціональних блоків F1, F2. При цьому функціональний блок F2 може бути відсутнім. Якщо умова U виконується, то відбувається вихід з блоку за стрілкою "+", якщо ні - за стрілкою "-".

Базова структура "цикл"

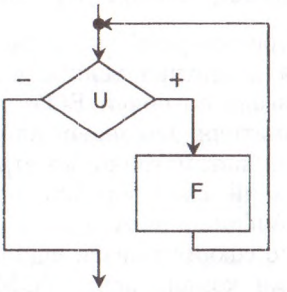
До складу структури входить логічний блок з перевіркою умови U і функціональний блок F, який називається *тілом циклу*. Зі структури "цикл" слідує, що тіло F може виконуватися неодноразово.

---

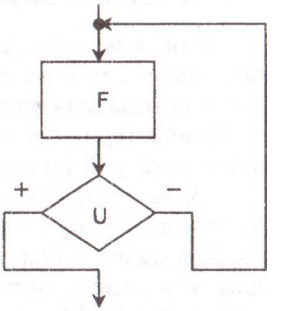
[1] Цей блок ще називають блоком обробки або обчислювальним блоком. У найпростішому випадку це арифметичний блок.



мал. 2



мал. 3



мал. 4

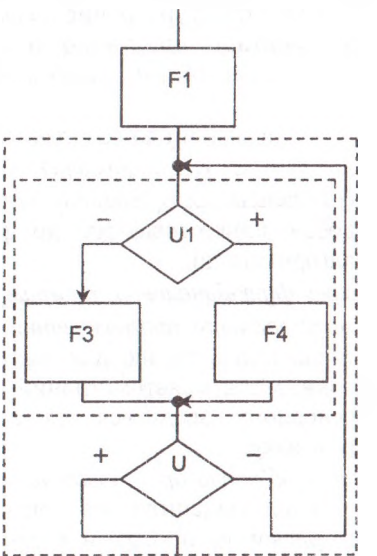
У залежності від взаємного розташування логічного та функціонального блоків, розрізняють цикли двох видів:

- ***цикл-поки***: тіло циклу розташоване після перевірки умови (мал. 3). Поки умова виконується, виконується і тіло. Тому можливий випадок, коли тіло циклу жодного разу не виконається. Тут U являється умовою продовження циклу (кожен раз, коли U істинно, тіло F виконується);
- ***цикл-до***: Тіло циклу розташоване до перевірки умови (мал. 4), тому тіло циклу завжди виконається хоча б один раз, незалежно від виконання умови. Тут U являється умовою виходу з циклу (як тільки U стає істинним, виконання тіла F припиняється).

Складання блок-схем

За структурним підходом дотримуються домовленостей: при складанні блок-схем дозволяється використовувати тільки три вказані базові структури. При цьому один з функціональних блоків можна заміняти будь-якою з базових структур. Такі домовленості дозволяють конструювати складні за змістом алгоритми, розвиваючи їх не тільки "в ширину", але і "в глибину".

Складемо блок-схему, скориставшись структурою "слідування", в якій на місці блоку F2 вставимо структуру "цикл-до", а в останній замість блоку F вставимо структуру "розгалуження". Матимемо структуру, як на мал. 5.



мал. 5

Сконструйовані у такий спосіб алгоритми мають чітку і зрозумілу структуру, бо складаються з обмеженої кількості блоків, побудованих аналогічно.

1.4. ЕОМ як виконавець алгоритму. Мови програмування

Щоб виконати алгоритм розв'язку задачі за прийнятний час, від виконавця вимагається достатньо велика швидкодія. Необхідність у таких виконавцях призвела до появи ЕОМ в середині 40-х років XX ст. Комп'ютери – автоматичні виконавці алгоритму. Кожна ЕОМ виконує задану їй програму автоматично без втручання людини.

ЕОМ – універсальний засіб обробки інформації. Це означає, що на одному і тому ж комп'ютері можна розв'язувати задачі різних класів. Для цього необхідно сформулювати відповідний алгоритм і подати його в термінах системи команд даної ЕОМ, тобто задати програму роботи на ЕОМ. Зміна програми призводить до зміни задачі, що розв'язується, але не вимагає зміни складу або конфігурації апаратного забезпечення комп'ютера.

Мови, на яких записуються програми для ЕОМ, умовно можна розбити на *мови програмування низького рівня* та *мови програмування високого рівня*.

Мова низького рівня – машинно-орієнтована мова програмування, що забезпечує явний доступ до архітектури елементів ЕОМ: регістрів, абсолютних адрес, портів введення-виведення тощо. Така мова призначається для символічних записів машинних команд.

Мова високого рівня – мова програмування, управляючі конструкції і структури даних якої відображають природні для людини поняття, а не структуру обчислювальної машини. Така мова не має машинної орієнтації, практично не залежать від конкретної ЕОМ, тому програми, що написані такою мовою, можна виконувати на будь-яких машинах.

За підходами до розв'язку задач мови високого рівня поділяються на:

- *процедурні (алгоритмічні) мови*. Наприклад, Basic, Pascal, C, HAM (навчальна алгоритмічна мова). Характерною ознакою таких мов є добра пристосованість до розв'язування тих задач, що підлягають алгоритмізації;
- *мови функціонального програмування*. Наприклад, Lisp;
- *мови логічного програмування*. Наприклад, Prolog. Характерною ознакою таких мов є те, що для них відсутнє традиційне поняття алгоритму – вони містять автоматичний розв'язувач задач і розв'язання задачі в основному зводиться до описання умови задачі, а не процедури розв'язку;
- *мови об'єктно-орієнтованого, модульного програмування тощо.*

Слід зазначити, що при розв'язування реальних практичних задач алгоритми та програми є досить складними. Раніше згадувалось про структурний підхід до розробки алгоритмів (див. стор. 6), який дозволяє полегшити роботу з вказаними задачами. Але існує узагальнення

такого підходу, яке полягає у використанні принципів структурного програмування.

Структурне програмування – це процес побудови алгоритмів та програм, що виконується у такій послідовності:

1. Попередній аналіз задачі з метою розбиття її на окремі прості частини (модулі). Для цього спочатку складають загальну схему алгоритму, а потім її деталізують.

2. Послідовна (зверху до низу) деталізація частин та складання програм для кожного з модулів. Виділяють основну частину та частини нижнього рівня. Кожну частину розробляють окремо: спочатку частини верхнього рівня, а потім – нижнього. У кінці частини з'єднують між собою.

Для структурного програмування характерно:

1. Використання трьох базових структур алгоритмів (див. стор. 6) при роботі з кожним модулем.

2. Коментування текстів програм.

3. Мінімальне використання вказівок переходу (операторів типу goto).

4. Передбачення і використання системи перевірки правильності програм.

Однією з тенденцій сучасного розвитку більшості мов є їх універсалізація. Наприклад, розроблені діалекти процедурних мов, що дозволяють реалізувати об'єктно-орієнтований підхід (Visual Basic, Delphi).

Більшість сучасних діалектів мов є системами програмування, тобто інтегрованим комплексом програм, що дозволяє не тільки записувати текст програми та транслювати її у машинний код, а й налагоджувати програму; забезпечує додаткові зручності користувачу при програмуванні (вбудована допомога, готові фрагменти та приклади програм, дружній інтерфейс тощо).

Зазначимо, що для різних мов або різних діалектів однієї і тієї ж мови трансляція тексту програми у машинний код може здійснюватися двома способами:

- *компіляція* (весь текст програми одразу переводиться у машинний код і перевіряється синтаксис всієї програми);
- *інтерпретація* (у машинний код текст програми переводиться послідовними фрагментами і для кожного фрагменту окремо перевіряється синтаксис).

ВСТУП ДО ПРОГРАМУВАННЯ

1. ПОРІВНЯЛЬНЕ ОПИСАННЯ АЛГОРИТМІЧНИХ МОВ

Характеристики алгоритмічних мов розглядатимемо, базуючись на роботі з: навчальною алгоритмічною мовою (НАМ) та практичною реалізацією її "Е-практикум, мехмат МГУ, 1981" для класу навчальної обчислювальної техніки "УАМАНА"; мовою програмування Basic (версія MSX-Basic); мовою програмування Pascal (версія 5.5).

1.1. Алфавіт

Навчальна алгоритмічна мова

Алфавіт НАМ містить:

1. Великі і малі латинські літери, великі і малі російські літери.
2. Арабські цифри від 0 до 9.
3. Знаки операцій: + , - , * , / , ** .
4. Знаки відношення: < , >= , > , <= , = , <> .
5. Розділові знаки: ( ) . , " := ␣(так позначатимемо знак "пропуск")

Basic

Алфавіт мови MSX-Basic містить:

1. Великі і малі латинські літери від A до Z, арабські цифри.
2. Знаки операцій: + , - , * , / , ^ , \ .
3. Знаки відношення: < , >= , > , <= , = , <>.
4. Розділові знаки: ( ) . , ; " ' : ␣
5. Спеціальні символи: $, %, #, &, @, !, ?
6. Зарезервовані (службові) слова: ABS, BEEP, CLS, DATA, REM тощо.

Pascal

Алфавіт мови Pascal містить:

1. Великі і малі латинські літери від A до Z.
2. Арабські цифри від 0 до 9.
3. Знаки операцій: + , - , * , / , < , >= , > , <= , = , <> , := .
4. Обмежувачі: . , ; ' : [ ] { } ..
5. Зарезервовані (службові) слова: AND, ARRAY, BEGIN тощо.

1.2. Константи, змінні. Типи даних

Усі дані програми інтерпретуються як константи або змінні. *Константи* не змінюють свого значення до завершення роботи програми. Константи бувають числовими та символьними. *Змінні* на відміну від констант можуть необмежену кількість разів змінювати своє значення і використовуються у програмі як носії вхідних даних, проміжних величин та результатів.

Для кожної змінної (константи), яка використовується у програмі, в оперативній пам'яті ЕОМ відводиться певна область у відповідності до вказаного імені та типу значень. Для того, щоб надати змінній конкретне значення (число, символ тощо), у кожній з розглядуваних мов програмування є вказівка (оператор) присвоєння. Така вказівка включає, поряд з іншим, символ =.

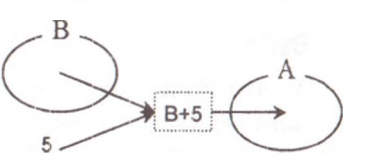
ЕОМ виконує вказівку присвоєння так:

- вираз, що записаний справа від символу = , обчислюється при поточних значеннях змінних;
- результат обчислення даного виразу поміщається у відповідну область оперативної пам'яті, відведеної для зберігання змінної, ім'я якої вказане зліва від символу =. Якщо до цього вказана область містила якесь значення, то воно зникає.

Прокоментуємо виконання вказівки присвоєння на прикладах:

а) A=5. Змінній A присвоїти 5, тобто у область оперативної пам'яті, відведену під змінну A, уміщується число 5;

б) A=B+5. До поточного значення змінної B додається константа - число 5 та результат додавання присвоюється змінній A, тобто з області оперативної пам'яті, відведеної під змінну B, береться те значення, яке міститься на момент звернення; до взятого значення додається число 5; результат додавання уміщується в область оперативної пам'яті, відведену під змінну A. Схематично це можна подати так як на мал. 6;



мал. 6

в) S=S+5. До поточного значення змінної S додається число 5 та результат додавання повертається у змінну S, тобто з області оперативної пам'яті, відведеної під змінну S, береться те значення, яке міститься на момент звернення; до взятого значення додається число 5; результат додавання повертається назад у ту ж область оперативної пам'яті.

Відомо, що будь-яка програма обробляє певні дані. На практиці зустрічаються різноманітні типи даних: цілі і дробові числа, символи, рядки, масиви, множини тощо. Під *типом* будемо розуміти множину значень, які може набувати змінна, і сукупність дій, які можна виконувати з такими значеннями.

Навчальна алгоритмічна мова

Ім'я змінної у Е-практикумі може складатися з цифр, великих і малих російських та латинських літер, але без пропусків. Російські та латинські літери однакових обрисів не розрізняються, тобто вважаються однаковими. Але, враховуючи те, що візуально важко визначити однаковість обрису літер різних мов, в іменах змінних рекомендується використовувати літери однієї мови. Починатися ім'я повинне з літери. Вказівка присвоєння значення змінній у НАМ має формат:

< ім'я змінної > := < вираз >. Наприклад, a := b + 5.

Е-практикум оперує з даними таких типів: вещ – дійсні числа, цел – цілі числа, лит – рядкові величини. Максимальна кількість символів (довжина), що входять до складу рядкової величини, рівна 16.

При обчисленні арифметичних виразів дійсні і цілі числа не розрізняються – всі обчислення проводяться з точністю до 5 значущих цифр. Модуль числа при цьому змінюється від 10^{-150} до 10^{150}. При присвоєнні цілочисельній змінній дійсного значення відбувається відкидання дробової частини.

Результати обчислень у цілих числах від –99999 до 99999 зображуються точно, без округлення. Більше число округляється і зображується у вигляді п'ятизначного числа, за яким слідує літера Е та степінь числа десять, на яку потрібно помножити це п'ятизначне число. Наприклад, число 1500000 можна записати як 1.5E+06.

Типи аргументів та результатів описуються у заголовку програми, типи проміжних змінних можуть міститися у рядку нач.

Basic

У мові Basic використовуються *константи* чотирьох типів:

- рядкова константа (довільна алфавітно-цифрова послідовність до 255 символів, яка у загальному випадку поміщається у подвійні лапки);
- ціле число (від –32768 до 32767);
- дійсне число подвійної точності (до 14 значущих цифр);
- дійсне число одинарної точності (до 6 значущих цифр).

Дійсні числа подаються у десятковому виді у фіксованій або плаваючій формі і лежать в межах $1∃10^{-64}$ до $1∃10^{63}$. Запис числа у плаваючій формі складається з мантиси, літери Е і десяткового порядку. Наприклад, число $2{,}75∃10^{-9}$ матиме вигляд 2.75E-9.

Ім'я змінної являє собою послідовність символів, яка складається з літер латинського алфавіту та цифр. Ця послідовність обов'язково має починатися з літери і може мати довільну довжину. Але розпізнаються імена за першими двома символами, тому імена ABC, AB1, AB55 позначатимуть одну й ту ж змінну. Зауважимо, що великі та малі лі-

тери латинського алфавіту не розрізняються. Оператор присвоєння значення змінній у мові Basic має формат:

LET < ім'я змінної > = < вираз >.

Службове слово LET є необов'язковим і, як правило, не записується. Наприклад, A = B + 5.

Для змінної тип може вказуватися в її імені останнім знаком: $ - рядкова величина, % - ціле число, # - дійсне число подвійної точності, ! - дійсне число одинарної точності. За замовчуванням Basic оперує з даними – дійсними числами подвійної точності.

Крім того, тип можна оголосити командою DEF:

DEFSTR - рядкова величина;
DEFINT - ціле число;
DEFDBL - дійсне число подвійної точності;
DEFSGN - дійсне число одинарної точності.

Pascal

Для іменування констант, типів, змінних тощо на мові Pascal використовуються *ідентифікатори* – неподільні послідовності символів алфавіту. Довжина ідентифікатора може бути довільною, але розрізняються перші 63 символи. Ідентифікатор починається з літери або символу підкреслення і не повинен містити пропусків. Після першого символу допускаються літери, цифри і знак підкреслення. Великі та малі літери не розрізняються.

Константи описуються зарезервованим словом Const, за яким записується список імен констант і кожній константі за допомогою символу = присвоюється значення. Значення констант можна також задавати виразом. Наприклад,

```
Const
  My_old     = 34 ;
  My_1Name = ' Oleg ' ;
  My_work    = My_old – 18 ;
```

Окрім звичайних констант часто використовуються типізовані константи, які можуть змінювати своє значення як змінні. При описанні таких констант додатково вказується тип:

```
Const
  Max_Ind    : word = 100 ;
  X          : integer = 50 ;
```

Для описання змінних використовується зарезервоване слово Var. Різні змінні одного типу відокремлюються комою, далі ставиться двокрапка і вказується певний тип цих змінних. Наприклад,

```
Var
    N, M      : integer ;
    Name      : string ;
```

У мові Pascal є такі *групи стандартних типів даних*: скалярні (цілі числа, дійсні числа, символи, булівський), структуровані, процедурні, покажчики. На відміну від інших мов, Pascal, крім стандартних типів, підтримує типи, які визначаються користувачем, що спрощує розв'язання деяких видів задач. Для визначення таких типів даних використовується зарезервоване слово Type.

Скалярні типи даних:

Вид	Назва	Діапазон
Цілі числа	byte	0 .. 255
	word	0 .. 65535
	integer	-32768 .. 32767
	shortint	-128 .. 127
	longint	-2147483648 .. 2147483647
Дійсні числа	real	2.9E-39 .. 1.7E38
	single	1.5E-45 .. 3.4E38
	double	5.0E-324 .. 1.7E308
	extended	3.4E-4932 .. 1.1E4932
	comp	-9.2E-18 .. 9.2E18
Символи	char	Кодова таблиця ЕОМ
Булівський	boolean	True, False

Скалярні типи даних, що визначаються користувачем:

- *перелік значень.* Задається безпосереднім переліком всіх значень, які може набувати змінна даного типу. Окремі значення вказуються через кому, а весь список заключається у круглі дужки. Наприклад, визначимо два типи даних: Animals, Names (перелік тварин, перелік імен) та опишемо тип змінних Animal, Name:

```
Type
   Animals  = ( Cat, Dog, Horse, Cow ) ;
   Names    = ( Oleg, Sasha, Inna ) ;
Var
   Animal   : Animals ;
   Name     : Names ;
```

- *інтервальний тип.* Задається з використанням двох констант, які визначають межі діапазону значень для даної змінної. Вказані константи повинні належати одному із стандартних типів (крім типу real). Значення першої константи має бути менше значення другої. Наприклад:

```
Const
   Min = 1 ;
   Max = 100 ;
Type
   Temperature = Min .. Max ;
```

Структуровані типи даних:

- *рядок.* Це послідовність символів кодової таблиці (від 0 до 255). При використанні у виразах рядок береться в одинарні лапки. Для визначення даних рядкового типу використовується зарезервоване

слово string, за яким у квадратних дужках вказується значення максимально допустимої довжини рядка даного типу (за замовчуванням – 255). Приклад описання даних рядкового типу:

```
Type
  NameStr   = string [ 30 ] ;
Var
  Name      : NameStr ;
```

До окремого символу рядка можна звернутися, якщо у квадратних дужках після імені вказати індекс рядкової змінної. Наприклад, до шостого символу змінної Name можна звернутися так: Name [6].

- *масив.* Складається з фіксованої кількості елементів, що мають один і той же тип. Для описання масивів використовується зарезервоване слово array.

Наприклад, опишемо одновимірний масив A, який може містити 10 елементів – дійсних чисел, та двовимірний масив B, який може містити 5x10 елементів – цілих чисел:

```
Type
   dataA    = array [ 1 .. 10 ]  of real ;
   masivB   = array [ 3 .. 7, 1 .. 10 ] of integer ;
Var
   A  : dataA ;
   B  : masivB ;
```

Доступ до окремого елемента масиву здійснюється шляхом індексування. Наприклад, A [1], B [3, 1] – перші елементи описаних масивів.

- *множина.* Складається з набору об'єктів, які взаємозв'язані за певною ознакою (групою ознак), і які можна розглядати як єдине ціле. Для описання множинного типу використовується словосполучення set of. Наприклад, опишемо дві множини:

```
Type
   Prost     = set of ( 3, 5, 7 ) ;
   Number    = set of 1 .. 31 ;
```

1.3. Вирази, операції. Числові функції

Зміні та константи використовуються у виразах. *Вираз* задає порядок виконання дій над елементами даних і складається з операндів (констант, змінних, звернень до функцій), круглих дужок, знаків операцій. *Операції* визначають дії, які виконуються над операндами.

Операції поділяються на арифметичні, відношення, логічні, рядкові тощо. Вирази відповідно називаються арифметичними, відношення, булівськими, рядковими тощо у залежності від того, якого типу операнди та операції у них використовуються. Розглянемо деякі основні операції.

Арифметичні операції

Арифметичні операції виконують арифметичні дії у виразах над значеннями операндів цілочисельних та дійсних типів.

Порівняльна таблиця деяких арифметичних операцій:

Дія	Операція		
	НАМ	Basic	Pascal
Піднесення до степеня	**	^	відсутня
Множення	*	*	*
Ділення	/	/	/
Цілочисельне ділення	відсутня	\	div
Остача від ділення націло	функція	MOD	mod
Додавання	+	+	+
Віднімання	–	–	–

Операції відношення

Операції відношення виконують порівняння двох операндів і визначають істинність або хибність значення виразу.

Порівняльна таблиця деяких операцій відношення:

Дія	Відношення		
	НАМ	Basic	Pascal
Дорівнює	=	=	=
Не дорівнює	< >	< >	< >
Більше	>	>	>
Менше	<	<	<
Більше або дорівнює	> =	> =	> =
Менше або дорівнює	< =	< =	< =
Належність списку	відсутня	відсутня	in

Логічні операції

Використовуючи прості вирази відношень, за допомогою логічних операцій одержують вирази відношень більш загального виду.

Порівняльна таблиця логічних операцій:

Дія	Операція		
	НАМ	Basic	Pascal
Логічне заперечення (не)	не	NOT	Not
Логічне І	и	AND	and
Логічне АБО	или	OR	or
Виключаюче АБО	відсутня	XOR	xor
Еквіваленція	відсутня	EQV	відсутня
Імплікація	відсутня	IMP	відсутня

Наведемо таблицю істинності для деяких виразів відношень загального виду. Позначатимемо I – істинно (True), X – хибно (False):

Прості вирази		Результат виконання дій виразів загального виду					
Y	Z	Не Y	Y І Z	Y АБО Z	Y викл. АБО Z	Y еквів. Z	Y імпл. Z
I	I	X	I	I	X	I	I
I	X	X	X	I	I	X	X
X	I	I	X	I	I	X	I
X	X	I	X	X	X	I	I

Пріоритет операцій

Порядок виконання операцій визначається їх пріоритетами. Якщо виникає потреба порушити вказаний порядок, то у виразах використовуються круглі дужки. Таблиця порядку виконання деяких операцій:

Пріоритет	Операція
Перший	Заперечення, піднесення до степеня
Другий	*, /, цілочисельне ділення, остача від ділення, логічне І
Третій	+, -, логічне АБО, виключаюче АБО
Четвертий	=, <>, <, >, <=, >=, належність списку

Числові функції

Аргументом будь-якої стандартної числової функції може служити довільний арифметичний вираз, який міститься у круглих дужках.

Таблиця основних числових функцій:

Дія, що виконуються	Функція		
	HAM	Basic	Pascal
Остача від ділення a націло на b	mod (a, b)	операція	операція
$y = x^2$	операція	операція	Sqr (x)
$y = \|x\|$	abs (x)	ABS (X)	Abs (x)
$y = \ln x$	ln (x)	LOG (X)	Ln (x)
$y = e^x$	exp (x)	EXP (X)	Exp (x)
$y = \sqrt{x}$	операція	SQR (X)	Sqrt (x)
$y = \sin x$	sin (x)	SIN (X)	Sin (x)
$y = \cos x$	cos (x)	COS (X)	Cos (x)
$y = \text{tg}\, x$	відсутня	TAN (X)	відсутня
$y = \text{arctg}\, x$	відсутня	ATN (X)	ArcTan (x)
Обчислення цілої частини	int (x)	INT (X)	Int (x)
Відкидання дробової частини	відсутня	FIX (X)	відсутня
Обчислення дробової частини	відсутня	відсутня	Frac (x)
Генерація випадкових чисел	відсутня	RND (x)	Random (x)

Зауважимо, що інші функції, не вказані у таблиці основних числових функцій, можуть окремо існувати як стандартні для певної мови (наприклад, у мові Pascal є стандартна функція Pi, яка вертає значення числа π). Разом з тим, інші функції можна виразити[1] через основні. Наприклад:

а) значення числа π: $\pi = 4 \cdot \mathrm{arctg}(1)$;

б) логарифм за основою a числа b: $\log_a b = \frac{\ln b}{\ln a}$;

в) піднесення числа m до степеня p: $m^p = e^{p \cdot \ln m}$

г) значення кута α з градусної міри у радіанну: $\alpha_{рад} = \frac{\pi}{180} \cdot \alpha_{град}$

Додатково опишемо *деякі скалярні функції та процедури, функції перетворення типів* для мови Pascal:

Dec (X {,n}) – процедура зменшує значення змінної X цілого типу на величину n. За відсутності необов'язкового параметра n значення X зменшується на одиницю;

Inc (X {,n}) – процедура збільшує значення змінної X цілого типу на величину n. За відсутності необов'язкового параметра n значення X збільшується на одиницю;

Odd (I) – повертає True, якщо ціле число I непарне, і False, якщо I парне.

Chr (I) – повертає символ стандартного коду обміну інформацією з номером, рівним значенню I;

Ord (S) – повертає порядковий номер значення S в множині, яка визначена типом S;

Round (X) – повертає значення X, яке округлене до найближчого цілого числа;

Trunc (X) – повертає найближче ціле число, менше або рівне X, якщо X>=0, і більше або рівне X, якщо X<=0.

---

[1] Див. довідники з математики. Наприклад, Литвиненко В.Н., Мордкович А.Г. Практикум по элементарной математике: Алгебра. Тригонометрия. – М.: Просвещение, 1991. – 352с.

У зразках та прикладах програм символами { } будемо обмежувати ті фрагменти, які можуть бути відсутні у тексті програм. Такими ж символами будемо обмежувати авторські коментарі, які не записуються до тексту програм. У символах < > поміщатимемо смислове описання параметрів (фрагментів), які необхідно мати у тексті програми для того, щоб вона виконувалася на ЕОМ.

2.1. Структура та синтаксис програм

Навчальна алгоритмічна мова.

Програма на НАМ це послідовність неіменованих вказівок, які виконуються у тому порядку, в якому вони записані. У загальному випадку програма повинна складатися з заголовку та тіла:

```
алг   <Ім'я> (<перелік аргументів і результатів та їх типи>)
  арг   <перелік аргументів> = < перелік значень>  {те, що відомо}
  рез   <перелік результатів>                      {те, що треба знайти}
нач {<перелік проміжних змінних та їх типи>}
    <тіло програми>
кон
```

Тут алг, арг, рез, нач та кон — службові слова. Елементи переліків відокремлюються між собою комами.

Особливості синтаксису для Е-практикуму.

Максимальна довжина програми у Е-практикумі становить 60 рядків, максимальне число змінних та елементів масивів – біля 360.

При запуску Е-практикуму для клавіатури встановлюється режим малих російських літер. Натиснувши та утримуючи клавіші Caps або Рус можна тимчасово змінити режим латиниці або кирилиці. Комбінація клавіш Ctrl + Caps (Ctrl + Рус) дозволяє включити постійний режим.

Імена змінних та алгоритмів можна записувати великими і малими, російськими і англійськими літерами, але без пропусків. Замість пропусків можна використовувати символ "підкреслення".

Службові слова пишуться тільки маленькими літерами, при цьому целтаб та вещтаб пишуться без пропуску, а литтаб відсутнє.

В рядках кон, кц, все забороняється будь-що записувати.

Описання проміжних змінних можуть міститися тільки у рядку нач, але не після нього.

Перевірка відповідності типу аргументу і вказаних для нього значень проводиться при виконанні програми, а не під час редагування тексту програми (не після переміщення курсору на інший рядок).

Основні команди Е-практикуму (клавіші та їх комбінації):

Ins / Del	- вставити /вилучити символ
Shift + Ins	- вставити порожній рядок
Shift + Del	- вилучити рядок або конструкцію (для вилучення конструкції необхідно курсор розташувати у першому її рядку)
BS	- вилучити символ зліва від курсору
ESC + А	- вставити конструкцію алгоритм
ESC + Е	- вставити конструкцію если
ESC + И	- вставити конструкцію иначе
ESC + В	- вставити конструкцію выбор
ESC + С	- вставити конструкцію при
ESC + Д	- вставити конструкцію цикл для
ESC + П	- вставити конструкцію цикл пока
F4	- виконати програму покроково
F9 (Shift + F4)	- виконати програму неперервно
Ctrl + Stop	- перервати виконання програми
Ctrl+S / Ctrl+Q	- призупинити / відновити виконання програми
Stop	- призупинити або відновити виконання програми
Cls	- очистити програму

Basic.

Програма на мові MSX-BASIC являє собою послідовність пронумерованих рядків (цілі числа від 0 до 65529) до яких записують команди (оператори) мови. Якщо певний рядок містить декілька операторів, то вони відокремлюються символом "двокрапка". Текст програми записується тільки літерами англійської мови, коментар або підказки можна записувати й кирилицею.

На початку програми як правило поміщають коментар, - після номера записують оператор REM або символ апострофу, після якого записують текст коментарю.

Текст програми, як правило, закінчується оператором END.

Для полегшення внесення змін у програму, рядки нумерують з деяким кроком, наприклад через 10 номерів: 10, 20, 30 і т.д. Кожен рядок повинен містити не більш як 225 символів – такий екранний рядок називається логічним.

Особливості роботи з MSX-BASIC на ЕОМ "YAMAHA".

Комп'ютер може працювати у двох режимах:

1. *Режим прямого виконання команд.* Команди (оператори) вводяться у новий рядок екрану без нумерації і виконуються безпосередньо після вводу в оперативну пам'ять – натисканням клавіші Enter (позначатимемо ⌟). Наприклад, набравши з клавіатури

```
A=5 : B=7 : PRINT "A="; A : PRINT "C="; A + B ⌟
```

одержимо на екрані результат:

```
A=5
C=12
```

2. *Режим виконання програми.* Програму можна виконати, якщо кожен її рядок занесено до оперативної пам'яті (ОП), причому кожен рядок команд, що має свій номер, в ОП заноситься окремо.

Щоб занести рядок у пам'ять, необхідно на екрані набрати його номер, записати його команди та, розмістивши курсор на символах даного рядка, натиснути клавішу Enter.

Перед виконанням програми, як правило, перевіряють її текст, що міститься у оперативній пам'яті (команда LIST).

Запустити програму на виконання можна, скориставшись командою RUN.

Команди Basic для роботи з програмою.

NEW	вилучає з ОП наявну програму
LIST	виводить записану в ОП програму або деякі її рядки на екран, наприклад: List (вся програма) List . (останній рядок програми або рядок, на якому припинилось виконання програми через помилку) List 20- (рядки з 20 до кінця програми) List -30 (рядки з початку програми до 30) List 50-80 (рядки з 50 по 80) List 60 (рядок 60)
RUN	починає виконання програми, наприклад: Run (виконання розпочинається з першого рядка програми) Run 100 (виконання програми розпочинається з рядка 100)
AUTO	забезпечує автоматичну нумерацію рядків при введенні тексту програми до ОП, наприклад: Auto , (нумерація з 0 кроком 10) Auto 20, 5 (нумерація з 20 кроком 5)
DELETE	вилучає рядки, які вже записані в ОП, наприклад: Delete . (останній рядок програми або рядок, на якому припинилось виконання програми через помилку) Delete -30 (рядки з початку програми до 30) Delete 5-10 (рядки з 5 по 10) Delete 60 (рядок 60)
RENUM	перенумеровує рядки програми, записаної в ОП, наприклад: Renum (всі рядки програми перенумеровуються з кроком 10) Renum 20, 5, 10 (починаючи з рядка 5 нумерація розпочнеться з номера 20 з кроком 10)
KEY OFF/ON	забирає з екрану/ виводить на екран значення функціональних клавіш
KEY LIST	виводить на екран 10 повідомлень про значення кожної функціональної клавіші
CONT	відновлює виконання програми з того місця, де воно було припинене оператором Stop або комбінацією клавіш Ctrl+Stop
TRON	включає режим трасування, за яким виконання програми супроводжується виводом на екран номерів послідовно виконуваних рядків
TROFF	відміняє введений раніше режим трасування
CLEAR	ініціалізує (виконує обнуління) всіх змінних

Pascal.

Структура програми включає заголовок, розділи опису та операторів:

```
Program <Ім'я_програми>;
   Uses      <перелік модулів>;
   Label     <перелік міток>;
   Const     <перелік констант>;
   Type      <перелік типів даних>;
   Var       <перелік змінних разом з описанням їх типів>;
 Procedure   <ім'я>;
     <тіло процедури>;
 Function <ім'я>;
     <тіло функції>;
Begin
   <оператори>
End.
```

Заголовок програми включає службове слово Program та ім'я програми (не більше одного слова), після якого ставиться символ "крапка з комою".

Розділ опису може містити:

- *розділ модулів* (USES). Модуль – це сукупність (бібліотека) описів констант, типів даних, змінних, процедур і функцій. Кожен модуль являє собою самостійну програму на Паскалі: він може містити кілька операторів, що викликаються перед запуском програми, і здійснювати всю необхідну ініціалізацію. Засоби, описані в модулі, можна використовувати у програмі. До переліку модулів включаються імена як стандартних модулів (Crt, Dos, Graph тощо), так і модулів, створених користувачами. Наприклад,

```
USES Crt, Dos, MyMod;
```

- *розділ міток* (LABEL). Мітки у переліку відокремлюються комою;
- *розділ констант* (CONST). Різні константи відокремлюються крапкою з комою. Наприклад,

```
CONST
  Vxid  = ' Блок ' ;
  Kod   = $124 ;
```

- *розділ типів* (TYPE). Якщо у розділі описується більш, ніж один тип даних, то вони відокремлюються крапкою з комою. Наприклад,

```
TYPE
  Matr  = array  [1 . . 5]  of real ;
  LatLit = ( 'a' . . 'z' ) ;
  Dni = 1 . . 31;
```

- *розділ змінних* (VAR). Між змінними різних типів ставиться крапка з комою. Наприклад,

```
VAR
  A, B, C    : integer ;
  Rez, S     : real ,
```

- *розділ процедур і функцій.* У цьому розділі розміщуються підпрограми, роль яких виконують процедури та функції. У загальному випадку така

підпрограма має структуру, аналогічну програмі.

Розділ операторів починається словом BEGIN і закінчується словом END (після якого обов'язково ставиться крапка), між цими словами розміщуються оператори програми. Оператори програми відокремлюються один від одного символом "крапка з комою".

Особливості синтаксису Pascal-програми.

Кожен елемент даних, що використовуються у програмі має бути описаний у розділах CONST або VAR. Виключення складають дані, які задані безпосередньо значенням. Наприклад,

```
Sum := X + 5          { 5 – елемент даних, заданий значенням }
```

Якщо у програмі використовуються нестандартні типи даних, то описання типу у розділі TYPE завжди передує описанню змінної.

Щоб у тілі програми прокоментувати деякі операції, можна у фігурних дужках { } записати текст коментаря. У середині коментарів не можна використовувати символи фігурних дужок.

Основні операції для роботи з програмою у середовищі програмування містить розділ "Додаткові можливості TURBO-PASCAL".

2.2. Введення, виведення даних. Лінійні програми

Під *лінійною програмою* будемо розуміти програму, алгоритм якої складається лише з структури слідування.

Навчальна алгоритмічна мова.

В НАМ введення даних реалізується у розділі арг: Після службового слова арг вказується перелік змінних, ставиться символ =, після якого записуються конкретні значення змінних у тому порядку, у якому слідують змінні. Наприклад:

```
арг A, B, C, D = 5, -7, 0, -3.26        { A=5, B= -7, C=0 , D= -3,26}
```

Для НАМ виведення даних передбачається у розділі рез: за закінченням виконання програми проти переліку змінних мають бути конкретні значення — результати роботи програми.

Але у середовищі Е-практикуму результати роботи програми виводяться на екран разом з проміжними результатами, - і не в розділі рез, а справа від тексту програми. Іноді важко визначити, які з даних на екрані є проміжними, а які результуючими. Тому рекомендується (там де це можливо) для змінних, які повинні містити результуючі значення, перед розділом кон використати операцію присвоєння. Наприклад, якщо про змінні Y та S відомо, що вони містять результати роботи програми (вказані у розділі рез), то потрібно записати:

```
  Y := Y
  S := S
кон
```

Для того, *щоб виконати програму у середовищі Е-практикуму* покроково, необхідно натиснути F4, неперервно – F9 (комбінація клавіш <Shift + F4>).

Basic.

У мові BASIC введення значень відбувається за допомогою оператора INPUT.

Наприклад, рядок програми, який забезпечує введення значення для змінної A буде мати вигляд (припустимо, хоча це і не суттєво, що рядок має номер 20):

```
20 INPUT A
```

Якщо потрібно ввести значення для кількох змінних, то можна скористатися декількома операторами INPUT. Але доцільніше використати один такий оператор, після якого записати змінні, відокремлені символом "кома".

Наприклад:

```
20 INPUT A, B
```

Оператор INPUT дозволяє виводити на екран коментарі, які полегшують введення даних. Коментарі містяться у лапках, після яких ставиться символ "крапка з комою", а далі записуються змінні. Текст коментарю може містити разом з англійськими й російські літери.

Наприклад:

```
20 INPUT "Введіть значення A і B"; A, B
```

При виконанні даного оператора на екрані комп'ютера з'являється знак питання і очікується введення даних. Якщо даних декілька, то при веденні вони відокремлюються комами. Введення закінчується натисканням клавіші Enter.

Виведення даних забезпечує оператор PRINT. Якщо необхідно вивести не одну, а декілька змінних або значень виразів, то після оператора записують їх перелік, відокремлюючи символами "кома" або "крапка з комою". Використання символу "крапка з комою" забезпечує пропуск на екрані однієї позиції після виведеного значення. Якщо ж після змінної або виразу використана кома, то на екрані значення займатиме зону шириною в 14 позицій. Аналогічно оператору INPUT, оператор PRINT теж може використовувати коментарі.

Наприклад,

```
50 PRINT "S= " ; S , "P= " ; P
```

Якщо у програмі використано декілька операторів PRINT, то для кожного оператора виведення починається з нового рядка.

Для того, *щоб виконати програму*, необхідно з нового рядка екрана записати команду RUN і натиснути клавішу Enter. Для виконання програми з рядка, що має певний номер N, потрібно виконати команду RUN N.

Pascal.

Введення даних на мові Pascal виконується за допомогою оператора READ.

Наприклад, фрагмент програми, який забезпечує введення значення для змінної A буде мати вигляд:

```
READ ( A ) ;
```

Якщо потрібно ввести значення для декількох змінних, то можна використати декілька операторів READ або для одного оператора READ записати перелік змінних, відокремлених комами.

Наприклад,

```
READ ( A , B ) ;
```

При виконанні оператора програма зупиняється і очікує на введення необхідної кількості значень для змінних. Якщо значень декілька, то вони відокремлюються пропуском. Закінчують введення завжди натисканням клавіші Enter.

Виведення даних на екран дисплея виконуються оператором WRITE. Після оператора в дужках вказується список виразів. Слідом за виразом, після двокрапки можна зазначити ширину поля екрана, в якому розміститься виведене значення. Наприклад, оператор WRITE (10:5, 7:5) виведе на екран числа 10 і 7 так[1]:

␣␣␣10␣␣␣␣7.

Для дійсних значень можна вказати, скільки цифр необхідно вивести на екран в дробовій частині числа.

Наприклад, якщо x = 31.23456098, а y = - 9032.45987, то оператор

```
WRITE ( X :6 :3 , Y :12 :2 ) ;
```

виведе на екран

31.234␣␣␣␣- 9032.45.

Якщо не використовувати такий спосіб виведення, то оператор WRITE дійсні значення виводить у режимі плаваючої крапки.

Щоб прокоментувати виведене значення, у список виведення можна поміщати рядки будь-яких символів, обмежених одинарними лапками. Наприклад,

```
Write ( ' S= ', S :10 :4, ' P= ', P :6 :3) ;
{у наведеному прикладі оператором Write на екран виведеться 4 значення:
  - рядок символів 'S='
  - поточне значення змінно S (у заданому форматі)
  - рядок символів 'P='
  - поточне значення змінно P (у заданому форматі)}
```

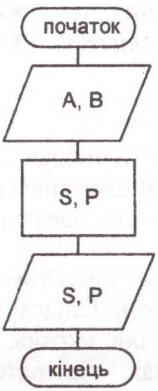
Існують допоміжні оператори введення та виводу, які при виконанні в програмі переводять курсор на новий екранний рядок — це оператори READLN і WRITELN.

---

[1] Нагадаємо, що символом ␣ ми позначаємо порожню позицію екрана

ПРАКТИЧНА РОБОТА №1

ПРИКЛАД 1. Обчислити площу та периметр прямокутника, якщо відомо його сторони.



мал. 7

І етап. Математична модель: маючи значення сторін a і b прямокутника, обчислимо його площу за формулою $S=ab$ та периметр за формулою $P=2(a+b)$.

ІІ етап. Складемо алгоритм розв'язку задачі у вигляді блок-схеми (мал. 7).

ІІІ етап. Запишемо текст програми.

```
алг Прямокутник (вещ a, b, S, P)
  арг a, b
  рез S, P
нач
    S := a * b
    P := 2 * ( a + b )
кон
```

НАМ-програма 1

```
10 REM Прямоугольник
20 INPUT "Введіть A і B" ; A, B
30 S = A * B
40 P = 2 * ( A + B )
50 PRINT "S= " ; S, "P= " ; P
60 END
```

Basic-програма 1

```
Program Rectangle; {прямокутник}
  Var
    A, B, S, P : Real;
Begin
      {вводимо данні}
  Write ( ' Введіть a= ' ) ; ReadLn (A) ;
  Write ( ' Введіть b= ' ) ; ReadLn (B) ;
      {виконуємо обчислення}
    S:= A * B;
    P:= 2 * ( A + B ) ;
      {виводимо результат на екран}
  WriteLn ( ' S= ', S :10 :4, ' P= ', P :6 :3)
 End.
```

Pascal-програма 1

IV етап. Тест. Вхідні дані: A=10, B=2. Результати: S=20, P=24. ■

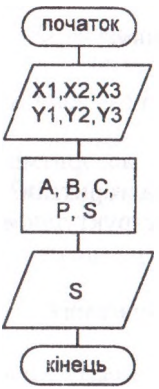
ПРИКЛАД 2. Обчислити площу трикутника, який задано координатами своїх вершин.

І етап. Математична модель: маючи координати вершин за формулою довжини відрізка ($L=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$) визначимо

довжини сторін. Для визначення площі трикутника скористаємося формулою Герона ($S=\sqrt{p(p-a)(p-b)(p-c)}$, де $p=\frac{a+b+c}{2}$).

II етап. Складемо алгоритм у вигляді блок-схеми (мал. 7).

III етап. Запишемо текст програми.



мал. 8

```
алг Герон (вещ x1, y1, x2, y2, x3, y3, S)
  арг x1, y1, x2, y2, x3, y3
  рез S
нач вещ a, b, c, p
    a := ((x2-x1)**2+(y2-y1)**2)**(1/2)
    b := ((x3-x2)**2+(y3-y2)**2)**(1/2)
    c := ((x3-x1)**2+(y3-y1)**2)**(1/2)
    p := (a+b+c)/2
    S := (p*(p-a)*(p-b)*(p-c))**(1/2)
    S := S
кон
```

НАМ-програма 2

```
10 REM Герон
20 INPUT "Введіть координати
точок" ; X1, Y1, X2, Y2, X3, Y3
30 A = SQR ((X2-X1)^2+(Y2-Y1)^2)
40 B = SQR ((X3-X2)^2+(Y3-Y2)^2)
50 C = SQR ((X3-X1)^2+(Y3-Y1)^2)
60 P = (A+B+C) / 2
70 S = SQR (P*(P-A)*(P-B)*(P-C))
80 PRINT "Площа дорівнює "; S
```

Basic-програма 2

```
Program Geron;
  Var
    A, B, C, P, S : Real;
    X1, Y1, X2, Y2, X3, Y3 : Real;
Begin
  Write ( ' Координати 1-ї точки ' ) ;
      ReadLn (X1, Y1) ;
  Write ( ' Координати 2-ї точки ' ) ;
      ReadLn (X2, Y2) ;
  Write ( ' Координати 3-ї точки ' ) ;
      ReadLn (X3, Y3) ;
  A := Sqrt ( Sqr (X2-X1)+Sqr (Y2-Y1) ) ;
  B := Sqrt ( Sqr (X3-X2)+Sqr (Y3-Y2) ) ;
  C := Sqrt ( Sqr (X3-X1)+Sqr (Y3-Y1) ) ;
  P := ( A + B +C ) / 2 ;
  S := Sqrt ( P*(P-A)*(P-B)*(P-C) ) ;
  WriteLn ( ' S= ', S :10 :4 )
End.
```

Pascal-програма 2

IV етап. Тест. Вхідні дані: X1=0 , Y1=0 , X2=3 , Y2=3 , X3=7 , Y3= -1 . Результат: S=12. ■

Лабораторна робота №1

Мета: Набути уміння та навички з введення та виведення даних, розробки та опису лінійних програм.

Матеріальне забезпечення: Е-практикум (MSX-Basic v.1.0, Pascal v.5.5)

Теоретична частина: завдання та контрольні питання.

І рівень.

1. На які характерні етапи можна розбити процес розв'язку задачі з використанням ЕОМ?
2. Означення алгоритму. Які основні властивості алгоритмів? Що означає "описати алгоритм"? Які є способи описання алгоритмів?
3. З яких базових фігур складають алгоритми при структурному підході?
4. Дати означення структурному програмуванню.
5. За якими ознаками можна поділяти мови програмування?
6. Яка програма називається лінійною?
7. Які оператори мови використовуються для введення та виведення даних? Описати синтаксис таких операторів.

ІІ рівень.

1. Описати етапи розв'язання задач: постановка задачі та описання алгоритму.
2. Описати етапи розв'язання задач: запис і трансляція програми, налагодження програми, експлуатація програми.
3. Описати кожну з основних властивостей алгоритмів.
4. У чому суть структурного підходу при розробці алгоритмів? Яких домовленостей дотримуються для складання блок-схем при структурному підході?
5. У чому полягає різниця між компіляцією та інтерпретацією при трансляції тексту програми у машинний код.
6. Яку структуру має програма? Особливості синтаксису програми та роботи з нею?

ІІІ рівень.

1. Як можна класифікувати мови програмування? Які тенденції їх розвитку?
2. Описати декомпозицію та абстракцію як методи проектування.
3. Вказати найбільш відомі методології проектування, орієнтовані на обробку. Описати методологію модульного програмування.
4. Вказати найбільш відомі методології проектування, орієнтовані на дані. Описати об'єктно-орієнтовану методологію проектування.
5. Описати синтаксис та пояснити роботу операторів DATA, READ, RESTORE (Basic).

Практична частина.

І рівень.

Знайти площі вписаного в трикутник і описаного навколо нього кругів. Трикутник задано координатами своїх вершин.

ІІ рівень.

1-3 варіанти. Знайти площу круга, вписаного у прямокутний трикутник з катетом a і прилеглим до нього гострим кутом. Передбачити переведення градусної міри кута у радіанну.

Вказівка: $S=\dfrac{\pi a^2\sin^2\alpha}{(\cos\alpha+\sin\alpha+1)^2}$.

4-6 варіанти. Знайти площу круга, вписаного у рівнобедрений трикутник з бічною стороною a і кутом α, протилежним основі. Передбачити переведення градусної міри кута у радіанну.

Вказівка: $S=\dfrac{\pi a^2\sin^2\alpha}{4(1+\sin\frac{\alpha}{2})^2}$.

7-9 варіанти. Знайти площу круга, вписаного у рівнобедрену трапецію з більшою основою a і гострим кутом α. Передбачити переведення градусної міри кута у радіанну.

Вказівка: $S=\dfrac{\pi a^2}{4}tg^2\dfrac{\alpha}{2}$.

10-12 варіанти. Суміжні сторони паралелограма дорівнюють a і b, а один з його кутів дорівнює α. Знайти косинус кута між діагоналями паралелограма. Передбачити переведення градусної міри кута у радіанну.

Вказівка: $\cos\varphi=\dfrac{a^2-b^2}{\sqrt{(a^2+b^2)^2-4a^2b^2\cos^2\alpha}}$.

13-15 варіанти. Одна з сторін паралелограма дорівнює a, більша діагональ d, кут між цією стороною і діагоналлю дорівнює β. Знайти синус кута між сторонами паралелограма. Передбачити переведення градусної міри кута у радіанну.

Вказівка: $\sin\alpha=\dfrac{d\cdot\sin\beta}{\sqrt{a^2+d^2-2\cdot a\cdot b\cdot\cos\beta}}$.

ІІІ рівень.

1 варіант. Визначити середню кінетичну енергію молекул ідеального газу при температурі t.

2 варіант. Обчислити площу повної поверхні зрізаного конуса, якщо відомі радіуси основ R, r і висота h.

3 варіант. Обчислити висоти трикутника зі сторонами a, b, c.

4 варіант. Визначити внутрішній об'єм циліндра двигуна, якщо його висота h, товщина стінок l, основа b, а зовнішній діаметр d.

5 варіант. Відрізок задано координатами його кінців M_1 (x_1, y_1), M_2 (x_2, y_2). Знайти координати точки $M(x, y)$, що ділить його у відношенні λ.

6 варіант. Трикутна піраміда задана координатами своїх вершин $A(x_1, y_1, z_1)$, $B(x_2, y_2, z_2)$, $C(x_3, y_3, z_3)$, $D(x_4, y_4, z_4)$. Визначити об'єм піраміди і площу повної поверхні.

7 варіант. Визначити довжини всіх сторін паралелепіпеда, у якого відомі діагоналі всіх граней.

8 варіант. Визначити площу опуклого чотирикутника з вершинами в точках $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$, $D(x_4, y_4)$.

9 варіант. Обчислити мінімальне значення гальмівного шляху автомобіля, що розпочав гальмування на горизонтальній ділянці шосе маючи швидкість v м/с. Коефіцієнт тертя дорівнює μ.

10 варіант. Обчислити площу повної позерхні правильної чотирикутної піраміди з основою a і висотою h.

11 варіант. Визначити механічну потужність крану, що підіймає вантаж масою m на висоту h за час t.

12 варіант. Знайти площу бічної поверхні правильної чотирикутної піраміди об'ємом V і висотою h.

13 варіант. Різниця між твірною конуса та його висотою a, а кут між ними α. Знайти об'єм конуса.

14 варіант. Визначити червону межу фотоефекту для металу з роботою виходу A.

15 варіант. Знайти повну поверхню конуса, як що відомо радіус основи R та довжина твірної l.

Вимоги до захисту лабораторної роботи.

Для певного рівня знати відповіді на питання та виконати завдання теоретичної частини. Протокол лабораторної роботи повинен містити назву теми, процес розробки математичної моделі (виведення формули) для 1-го та 3-го рівнів, блок-схеми алгоритмів, тексти програм, тести: вхідні дані та результати роботи програми.

2.3. Програми з розгалуженнями

У розглянутих раніше прикладах та задачах всі дії виконувались послідовно одна за одною, але насправді таких задач дуже мало.

Багато задач вимагає від виконавця змінити послідовність виконання дій *у залежності від певних умов*. Під вказаними умовами будемо розуміти логічні вирази, кожен з яких може набувати двох значень: "так" (true) – якщо умова істинна і "ні" (false) – якщо умова хибна.

Нагадаємо, що умови можуть бути складними, тобто використовувати логічні операції[1]. Прості умови обмежуються дужками, якщо вони є частинами складної умови. Істинність складних умов визначається за таблицею істинності (стор. 17).

Наприклад, на деякому етапі розв'язку певної задачі виконавцю потрібно послідовно виконати дві дії: обчислити значення виразу *x-1*, а потім і значення кореня квадратного цього виразу. У цьому випадку необхідно передбачити виконання таких дій:

1. обчислити значення виразу *x-1*;
2. у залежності від результатів перевірки умови змінити послідовність виконання дій, тобто, якщо умова $x-1 \geq 0$ є істинною, то обчислювати значення $\sqrt{x-1}$, інакше – умова є хибною – виконати інші дії (наприклад, запропонувати користувачу ввести правильні дані).

Алгоритми та відповідні програми, які вимагають від виконавця перевірки деяких умов і зміни послідовності виконання дій у залежності від результатів перевірки, будемо називати *алгоритмами (програмами) з розгалуженнями*.

Розв'язуючи задачі з розгалуженнями, при описанні алгоритмів обов'язково використовують базову структуру "розгалуження" (див. стор. 6), а у програмах – оператор (вказівку) розгалуження або, як його ще називають, оператор умовного переходу.

Для організації зміни послідовності виконання дій при розв'язуванні задачі іноді використовують (Basic, Pascal) *оператор безумовного переходу* GOTO. Зауважимо, що його використання може порушити структурний підхід до розв'язування задачі і призвести до логічних помилок, які важко шукати у тексті програми.

Навчальна алгоритмічна мова.

Вказівка розгалуження у НАМ має такий вигляд:

```
если <умова>
    то <1-ша послідовність дій>
    иначе <2-га послідовність дій>
все
```

Якщо умова істинна, то виконується 1-ша послідовність дій, інакше (умова хибна) то виконується 2-га послідовність дій.

Іноді вказівка розгалуження використовується у неповній формі:

```
если <умова>
    то <послідовність дій>
все
```

Узагальненням вказівки розгалуження є вказівка вибору, за допомогою якої ефективно описуються розгалуження, що мають більше як три напрями:

---

[1] див. порівняльну таблицю логічних операцій у п. 1.3.

```
выбор
    при <умова 1>
            <1-ша послідовність дій>
    при <умова 2>
            <2-га послідовність дій>
    ...
    при <умова N>
            <N-на послідовність дій>
    иначе <серія інших команд>
все
```

У неповній формі оператора вибору конструкція иначе відсутня.

Basic.

Оператор розгалуження на мові Basic можна подати в повній та неповній формах. Повна форма має вигляд:

```
IF <умова> THEN <1-ша серія операторів> ELSE <2-га серія операторів>
```

Неповна форма оператора розгалуження має наступний вигляд:

```
IF <умова> THEN <серія операторів>
```

Зауважимо, що кожна з вказаних форм має міститися в одному логічному рядку. Враховуючи те, що кількість символів (а значить і кількість операторів серії) логічного рядка обмежена, іноді оператор розгалуження записують у декількох логічних рядках. Такий запис використовують і для більш зручного візуального подання тексту програми, що може полегшити її перегляд та пошук можливих помилок. При цьому необхідно користуватися оператором безумовного переходу GOTO, що без будь-яких умов змінює послідовність виконання команд програми.

Розглянемо різні способи запису фрагмента програми, в якому спочатку у залежності від істинності умови змінюється послідовність виконання дій - визначається більше із двох чисел, а потім послідовність дій продовжується – на екран виводиться значення більшого числа.

1-й спосіб:

```
50  IF A>B THEN Y=A ELSE Y=B        {повна форма розгалуження}
60  PRINT Y
```

Спочатку виконується рядок 50: якщо умова істинна, то змінній Y присвоюється значення числа A, інакше змінній Y присвоюється значення числа B. Далі виконується наступний рядок 60.

2-й спосіб:

```
50  IF A>B THEN 60 ELSE 70          {повна форма розгалуження}
60  Y=A : GOTO 80
70  Y=B
80  PRINT Y
```

Спочатку виконується рядок 50:

якщо умова істинна,

то виконання програми перейде на рядок 60, де змінній Y присвоїться значення числа A і виконання програми перейде на рядок 80,

інакше (умова хибна) виконання перейде на рядок 70, змінній Y присвоїться значення B і робота продовжиться у наступному рядку.

Далі виконується рядок 80.

3-й спосіб:

```
50 IF A>B THEN Y=A : GOTO 70    {неповна форма розгалуження}
60 Y=B
70 PRINT Y
```

Спочатку виконується рядок 50:

якщо умова істинна, то змінній Y присвоїться значення числа A і виконання програми перейде на рядок 70,

інакше (умова хибна) виконання програми перейде на наступний рядок, де змінній Y присвоїться значення числа B.

Далі виконується рядок 70.

Якщо після THEN і перед ELSE (1-ша серія операторів) записується декілька операторів, то оператори один від одного відокремлюються двокрапкою. Аналогічно поступають з 2-гою серією операторів.

Оператор вибору реалізовано командою ON. За допомогою цієї команди можна реалізувати багаторазово розгалужений алгоритм[1].

Зауважимо, що доцільно оператор вибору реалізовувати за допомогою неповної форми оператора розгалуження, наприклад:

```
40 IF <умова1>   THEN <1-ша послідовність операторів> : GOTO 100
50 IF <умова2>   THEN <2-га послідовність операторів> : GOTO 100
...
90 IF <умоваN>   THEN <N-нна послідовність операторів> : GOTO 100
```

Pascal.

Оператор умовного переходу на мові Pascal можна подати в повній та неповній формах. Повна форма має вигляд:

```
If <умова> Then <оператор1>
           Else <оператор2> ;
```

Якщо умова істинна (True), виконується оператор1, інакше (умова хибна - False) виконується оператор2.

Неповна форма оператора умовного переходу має вигляд:

```
If <умова> Then <оператор1> ;
```

Якщо умова істинна, виконується оператор1, інакше – пропускається.

Для того, щоб була можливість виконати серію операторів, існує поняття та конструкція складового оператора.

Складовий оператор є групою із декількох операторів, відокремлених один від одного символами "крапка з комою", і обмежену оператор ними дужками Begin і End. Його схема така:

---

[1] Ця команда може бути також використана для обробки переривань, викликаних натисканням функціональної клавіші, "пропуску", одночасним натисканням клавіш Ctrl+Stop, вичерпанням інтервалу часу тощо. При цьому команда ON має інший синтаксис.

```
Begin
   < оператор1 ; >
   . . .
   < операторN >
End;
```

Складовий оператор сприймається як єдине ціле і може знаходитися у будь-якому місці програми, де можна використовувати простий оператор[1].

Змінити послідовність виконання дій дозволяє *оператор безумовного переходу* Goto. Він використовується тоді, коли після виконання деякого оператора без будь-якої умови потрібно виконати не наступний, а деякий інший, позначений міткою оператор. Формат оператора:

```
GOTO M ;
```

Цей оператор означає "перейти до" і перенаправляє виконання програми (здійснює перехід) до оператора з міткою M. Міткою можуть бути імена або цифри, які мають бути описані в розділі міток. Мітка ставиться зліва від оператора та відокремлюється від нього символом "двокрапка". Одна мітка не може використовуватись двічі, але до однієї мітки можна звертатися з різних ділянок програми.

Наведемо фрагмент програми з використанням оператора умовного переходу, складового оператора та оператора безумовного переходу:

```
M1 : Readln ( X ) ;
     If  ( X - 1) >= 0  Then Y := Sqrt ( X – 1 )
     Else Begin
       Writeln ( ' Неправильні дані. Повторіть ввід X. ' ) ;
       Goto M1
     End ;
```

Оператор вибору CASE дозволяє здійснити вибір із довільної кількості існуючих варіантів. Він складається з виразу (селектору) та списку параметрів (операторів), кожному з яких передує список констант вибору. Список констант може складатися з однієї константи). При цьому тип констант має бути сумісним з типом селектора. Для селектора заборонені типи real, string.

Формат оператора вибору:

```
CASE <вираз> OF
   <1-й список констант> : <оператор1 ;>
   . . .
   <N-й список констант> : <операторN >
  ELSE < оператор >
END ;
```

При виконанні оператора <вираз> обчислюється і його значення шукається в одному із списків констант. Після цього виконується оператор, що відповідає знайденому списку.

---

[1] Оператори, що не містять ніяких інших операторів, називають простими. До них відносяться оператори присвоєння, безумовного переходу, порожній оператор тощо.

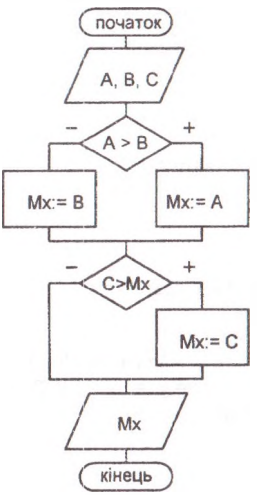
ПРАКТИЧНА РОБОТА №2

ПРИКЛАД 3. Знайти найбільше з трьох даних чисел.

І етап. Математична модель: спочатку знайдемо найбільше з двох перших чисел, а потім порівняємо знайдене число з третім. Таким чином знайдемо найбільше.

ІІ етап. Складемо алгоритм у вигляді блок-схеми (мал. 9).

ІІІ етап. Запишемо текст програми.



мал. 9

```
алг БЗТ (вещ a, b, c, mx)
  арг a, b, c
  рез mx
нач
    если a > b
      то mx := a
      иначе mx := b
    все
    если c > mx
      то mx := c
    все
mx := mx
кон
```

НАМ-програма 3

```
10 REM БЗТ
20 INPUT "Введіть три числа" ; A, B, C
30 IF A>B THEN MX=A ELSE MX=B
40 IF C>MX THEN MX=C
50 PRINT"Найбільшим є ";MX
60 END
```

Basic-програма 3

```
Program Max3;
  Var
    A, B, C, Mx : Real;
Begin
  Write ( 'Введіть три числа' ) ;
      ReadLn (A, B, C) ;
  If A > B Then Mx := A Else Mx := B;
  If C > Mx Then Mx := C ;
  WriteLn ( 'Найбільшим є', Mx :0 :4 )
End.
```

Pascal-програма 3

IV етап. Виконаємо повне тестування (за всіма лініями потоку).

Тест1. Вхідні дані : A=5, B=0, C= 7. Результат: 7.

Тест2. Вхідні дані: A=5, B=0, C= -1. Результат: 5.

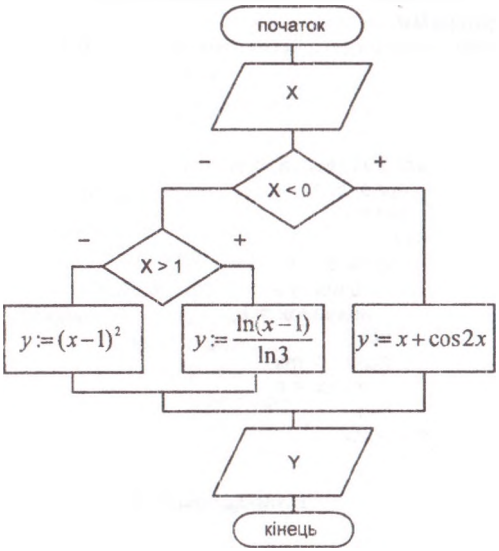
Тест3. Вхідні дані: A=0, B=6, C= -1. Результат: 6. ■

ПРИКЛАД 4. Обчислити значення функції $y=\begin{cases} x+\cos 2x, \text{ якщо } x<0 \\ (x-1)^2, \text{ якщо } 0\le x\le 1 \\ \log_3(x-1), \text{ якщо } x>1 \end{cases}$.

І етап. Математична модель: значення даної функції обчислюється за формулами як значення одного із трьох виразів у залежності від значення аргумента.

ІІ етап. Складемо алгоритм у вигляді блок-схеми (мал. 10).

ІІІ етап. Запишемо текст програми.



мал. 10

```
алг Функц (вещ x, y)
  арг x
  рез y
нач
  если x < 0
    то y := x + cos(2*x)
    иначе
        если x > 1
          то y := ln ( x -1) / ln (3)
          иначе y := ( x - 1 )**2
        все
  все
  y := y
кон
```

НАМ-програма 4

```
10 REM Функція
20 INPUT "Введіть значення для x=" ; X
30 IF X < 0 THEN Y=X + COS(2*X)  ELSE
IF X>1 THEN Y=LOG(X-1)/LOG(3) ELSE
Y=(X-1)^2
40 PRINT "При x=";X, "y=";Y
50 END
```

Basic-програма 4

```
Program FNF;
  Var
    X, Y : Real;
Begin
  Write ( 'Введіть значення для x=' ) ;
      ReadLn (X) ;
  If X<0 Then Y := X + Cos(2*X)
  Else
    If X>1 Then Y := Ln(X-1)/Ln(3)
    Else Y := Sqr(X-1);
  WriteLn ( 'При x=', X :10 :4, ' y=',Y:10:4 )
End.
```

Pascal-програма 4

ІV етап. Виконаємо повне тестування (за всіма лініями потоку). Тест1: X=0.5, Y=0.25. Тест2: X=10, Y=2. Тест3: X= - 3.14, Y= - 2.14. ■

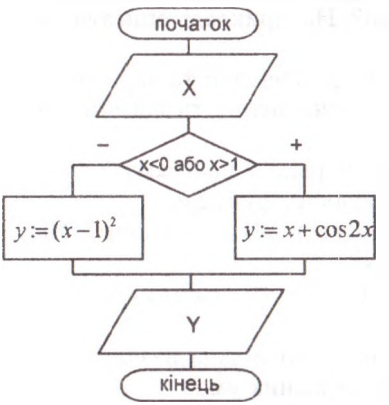
ПРИКЛАД 5. Обчислити значення функції: $y=\begin{cases} x+\cos 2x, \text{ якщо } x<0 \\ (x-1)^2, \text{ якщо } 0\le x\le 1 \\ x+\cos 2x, \text{ якщо } x>1 \end{cases}$.

І етап. Математична модель. Дана функція подібна раніше розглянутій (Приклад 4.): вираз $\log_3(x-1)$ замінено на $x+\cos 2x$.

Тоді дану функцію доцільно подати як: $y=\begin{cases} x+\cos 2x, \text{ якщо } x<0 \text{ або } x>1 \\ (x-1)^2, \text{ якщо } 0\le x\le 1 \end{cases}$

ІІ етап. Складемо алгоритм у вигляді блок-схеми (мал. 11).

ІІІ етап. Запишемо текст програми.



мал. 11

```
алг Функц (вещ x, y)
  арг x
  рез y
нач
    если x<0 или x>1
      то y := x + cos(2*x)
      иначе y := (x-1)**2
    все
кон
```

НАМ-програма 5

```
10 REM Функція
20 INPUT "Введіть значення для x=" ; X
30 IF X<0 OR X>1 THEN Y=X+COS(2*X)
ELSE Y=(X-1)^2
40 PRINT "При x=";X, "y=";Y
50 END
```

Basic-програма 5

```
Program FNF ;
  Var
    X, Y : Real ;
Begin
  Write ( 'Введіть значення для x=' ) ;
      ReadLn (X) ;
  If (X<0) Or (X>1) Then Y := X + Cos(2*X)
  Else Y := Sqr(X-1) ;
  WriteLn ( 'При x=', X :10 :4, ' y=',Y:10:4 )
End.
```

Pascal-програма 5

ІV етап. Виконаємо повне тестування.

Тест1: X=0.5, Y=0.25.

Тест2: X= - 3.14, Y= - 2.14. ■

Лабораторна робота №2

Мета: Набути уміння та навички розробки та описання програм з розгалуженнями.

Матеріальне забезпечення: Е-практикум (MSX-Basic v.1.0, Pascal v.5.5)

Теоретична частина: завдання та контрольні питання.

І рівень.

1. У чому полягає різниця між константами та змінними? Які типи констант та змінних використовують у програмах?
2. Поняття змінної. Який синтаксис має ім'я змінної? Як описується змінна для використання у програмі? На прикладі описати виконання вказівки присвоєння.
3. Поняття виразу та операції. Які є види операцій та виразів?
4. Які арифметичні операції, операції відношення та логічні операції можна реалізувати у програмі?
5. За яким пріоритетом виконуються операції у програмах?
6. Описати синтаксис стандартних числових функцій.
7. Яка програма називається програмою з розгалуженням? Яка структура блок-схеми описує таку програму?
8. Яка форма вказівки розгалуження (умовного переходу)?

ІІ рівень.

1. Які умови в програмах з розгалуженнями можна назвати простими? Які складними? Описати синтаксис складних умов.
2. Скласти таблицю істинності для умов, що використовують операції порівняння та операцію заперечення.
3. Навести приклади виразів, що дозволять у програмах: обчислити значення логарифма числа a за основою b, перевести значення кута з градусної міри у радіанну (HAM, Basic, Pascal), обчислити значення числа π (Basic), піднести число до степеня (Pascal).
4. За яких обставин у програмах (Basic, Pascal) використовують оператор безумовного переходу?
5. Які стандартні склярні типи даних використовуються у програмах (Pascal)?

ІІІ рівень.

1. Який тип можуть мати аргументи та значення стандартних числових функцій?
2. Описати значення істинності для виразів зі складними умовами.
3. Описати формат, навести приклади використання стандартного оператора вибору.
4. Описати скалярні типи даних, що визначаються користувачем (Pascal).

Практична частина.

І рівень.

Обчислити корені квадратного рівняння $ax^2+bx+c=0$. У програмі передбачити перевірку значення коефіцієнта a, який вводитиметься користувачем програми.

ІІ рівень.

Обчислити значення функції:

1-3 варіанти.
$$y=\begin{cases}\lg(10-x)+2, & x<10\\ \dfrac{\sin^2(\pi x)}{14+x}, & 10\le x\le 14\\ x+\sqrt{x-14}, & x>14\end{cases}$$

4-6 варіанти.
$$y=\begin{cases}\sin^2 4(x+3)+x, & x\le -3\\ \dfrac{x+2}{x^2+x-6}-2, & -3<x<2\\ e^{x-3}+x^3-2x-1, & x\ge 2\end{cases}$$

7-9 варіанти.
$$y=\begin{cases}\cos\sqrt{|x+1|}+3, & x\le 1.5\\ \dfrac{x+0.75}{\sqrt{x^2-2.25}}, & 1.5<x<7\\ \log_4(x+2)+x, & x\ge 7\end{cases}$$

10-12 варіанти.
$$y=\begin{cases}\sqrt[4]{8-x^3}+5x-10, & x\le 2\\ \dfrac{\sin(x-5)^2}{-x^2+10x-16}, & 2<x<8\\ \lg(x-7)+2x+5, & x\ge 8\end{cases}$$

13-15 варіанти.
$$y=\begin{cases}\log_7(8-x)+3x-1, & x<8\\ \dfrac{\sqrt{x-8}+x}{x^2-17x+70}, & 8\le x<10\\ \sin 2x-\cos 2x+2\sin^2 x, & x\ge 10\end{cases}$$

ІІІ рівень.

1 варіант. Визначити довжину найменшої з сторін трикутника, утвореного прямими $a_1x+b_1y+c_1=0$, $a_2x+b_2y+c_2=0$, $a_3x+b_3y+c_3=0$.

2 варіант. У середині квадрата з вершинами $O(0, 0)$, $A(0, a)$, $B(a, a)$,$C(a, 0)$ розміщена точка $M(x, y)$. Визначити найменшу площу трикутника утвореного точкою M і двома довільними вершинами квадрата.

3 варіант. Визначити найменшу довжину ламаної, що проходить через точки $M_1(x_1, y_1)$, $M_2(x_2, y_2)$, $M_3(x_3, y_3)$, $M_4(x_4, y_4)$. Ламана є не замкненою.

4 варіант. Перевірити, чи можна з векторів $a(a_1,a_2)$, $b(b_1,b_2)$, $c(c_1,c_2)$ утворити трикутник.

5 варіант. На площині задано чотирикутник координатами своїх вершин. Визначити найбільшу з його сторін (її довжину).

6 варіант. Визначити найбільшу з сторін трикутника, утвореного прямими $a_1x+b_1y+c_1=0$, $a_2x+b_2y+c_2=0$, $a_3x+b_3y+c_3=0$.

7 варіант. Скільки мають спільних точок три кола, що задано координатами своїх центрів та радіусами.

8 варіант. Розв'язати біквадратне рівняння $ax^4+bx^2+c=0$.

9 варіант. Дано три комплексних числа $\alpha_1+i\beta_1$, $\alpha_2+i\beta_2$, $\alpha_3+i\beta_3$. Визначити число, модуль якого найбільший.

10 варіант. Знайти найбільшу висоту трикутника зі сторонами *a, b, c*.

11 варіант. Знайти найменшу висоту трикутника зі сторонами *a, b, c*.

12 варіант. Знайти найбільшу медіану трикутника зі сторонами *a, b, c*.

13 варіант. Знайти найменшу бісектрису трикутника зі сторонами *a, b, c*.

14 варіант. Перевірити, чи є вектори $a(a_1,a_2,a_3)$, $b(b_1,b_2,b_3)$, $c(c_1,c_2,c_3)$ компланарними.

15 варіант. Скільки необхідно енергії, щоб нагріти воду (яка може бути і у вигляді льоду) від температури t_0 температури t_1 при нормальному тиску.

Вимоги до захисту лабораторної роботи.

Для певного рівня знати відповіді на питання та виконати завдання теоретичної частини. Протокол лабораторної роботи повинен містити назву теми, процес розробки математичної моделі (виведення формули) для 1-го та 3-го рівнів, блок-схеми алгоритмів, тексти програм, тести: вхідні дані та результати роботи програми.

2.4. Циклічні програми

При розв'язуванні багатьох задач іноді доводиться багаторазово виконувати певні дії. Такий процес назвемо циклічним, а відповідні алгоритми та програми – циклічними. Для графічного описання алгоритмів розв'язку таких задач використовується базова структура "цикл".

У математиці прикладом циклу є процес табулювання функції, де значення функції, заданої деяким виразом (формулою) обчислюється декілька разів при різних значеннях аргументу.

З погляду на програму, цикл - це така її конструкція, за допомогою якої деяка серія вказівок (операторів) програми повторюється скінчену кількість разів. Причому вказівки серії для кожного повтору, як правило, використовують різні значення величин.

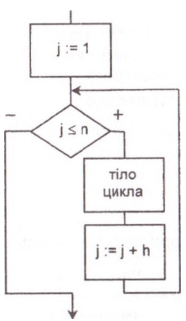
Серія повторюваних вказівок (операторів) програми складає *тіло циклу*. Умову, за якої відбуваються повтори тіла циклу, називатимемо

умовою циклу. У залежності від взаємного розташування тіла та умови розрізняють два види циклів: цикл-поки (цикл з передумовою) та цикл-до (цикл з післяумовою). З циклу-поки можна виділити окремий вид – цикл-для (цикл з параметром).

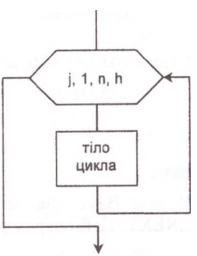
Цикл-для. У такому циклі вказується *для* яких послідовних значень параметра потрібно циклічно виконати серію вказівок. При цьому для параметра циклу (*i*) задається початкове значення, і в кожному оберті циклу параметр змінюється за законом арифметичної прогресії ($i = i + h$, де h – крок циклу) доти, доки поточне значення параметра не перевищить задане кінцеве значення. Зауважимо, що для деяких мов програмуванням h може бути нецілим числом.

Такий вид циклу використовується тоді, коли у програмі, перед виконанням тіла циклу, наперед відомо яку кількість повторів тіла циклу необхідно виконати.

При структурному підході до графічного описання алгоритмів для вказаного циклу використовується базова структура "цикл-поки". Можна також використати структуру з блоком послідовної зміни параметра – шестигранником (структура "для"). Наприклад, для одного і того ж циклу з параметром *j*, що послідовно змінюється від 1 до *n* з кроком *h*, наведемо базову структуру "цикл-поки" (мал. 12) та структуру "для" (мал. 13):



мал. 12



мал. 13

Цикл-поки (цикл з передумовою) призначений для повторного виконання деякої вказівки чи серії вказівок, поки виконується умова циклу. Цей цикл використовується у тих випадках, коли наперед невідома кількість повторень тіла циклу. Зауважимо, що при виконанні циклу-поки можлива ситуація, коли тіло циклу не виконається жодно-

го разу. Для графічного описання алгоритмів вказаного циклу використовується базова структура "цикл-поки".

Варто зазначити, що у програмах, які використовують цикл-поки, тіло циклу має містити команду, циклічне виконання якої приводить до зміни істинності умови циклу. Інакше цикл може виконувати нескінченну кількість повторів - програма "зациклюється".

Цикл-до (цикл з післяумовою) повторно виконує деяку вказівку чи серію вказівок доти, доки не виконається умова виходу з циклу. Як і цикл-поки цей цикл, як правило, використовують тоді, коли наперед невідома кількість повторень тіла циклу. Але, на відміну від циклу-поки, тіло циклу-до виконається хоча б один раз за будь-яких умов. Для графічного описання алгоритмів вказаного циклу використовується базова структура "цикл-до".

Аналогічно програмам з циклом-поки, програми, що використовують цикл-до, у тілі циклу мають містити команду, циклічне виконання якої приводить до зміни істинності умови циклу.

Навчальна алгоритмічна мова

Для реалізації циклічної конструкції у НАМ використовують вказівку пока або вказівку для. Вказівка пока у програмі забезпечує роботу циклу-поки (циклу з передумовою). Структура вказівки:

```
нц пока <умова>                          {початок тіла циклу}
  <серія команд>                         {тіло циклу}
кц                                       {кінець тіла циклу}
```

Структура вказівки для, яка реалізує цикл з параметром:

```
нц для <змінна> от <поч_знач.> до <кінц_знач.> шаг <значення>
  <серія команд>                         {тіло циклу}
кц
```

Basic

У мові Basic для реалізації циклу для використовують оператор FOR...NEXT. Цей оператор має таку структуру:

```
FOR < змінна > = < знач1> TO < знач2 > STEP < знач3 >
<серія операторів>                       {тіло циклу}
NEXT <змінна>
```

Тут змінна є параметром циклу, знач1 - початкове значення параметра, знач2 - кінцеве значення параметра, знач3 - крок циклу.

Після кожного виконання тіла циклу, при виконанні оператора NEXT, значення змінної збільшується на величину, що стоїть після вказівки STEP (знач3). Інакше крок циклу вважається рівний 1.

Циклічну конструкцію також можна реалізувати за допомогою операторів умовного, безумовного переходу (див. приклад на стор. 45).

Pascal

Конструкція "цикл-поки" (цикл з передумовою) у мові Pascal реалізована оператором, що має форму:

```
WHILE < логічний вираз > DO < тіло циклу > ;
```

Тіло циклу буде повторюватися доти, доки істинний логічний вираз. Нагадаємо, що для того, щоб програма не "зациклювалася", тіло циклу має містити оператори, які впливали б на істинність умови циклу. Якщо тіло циклу складається з декількох операторів, то їх об'єднують у складовий оператор.

Конструкція "цикл-до" (цикл з післяумовою) реалізована оператором, що має форму:

```
REPEAT < тіло циклу > UNTIL < логічний вираз > ;
```

Тіло циклу, розміщене між ключовими словами REPEAT (повторювати) і UTIL (до), повторюється доти, доки логічний вираз, що йде після слова UNTIL, не стане істинним. Якщо тіло циклу для вказаного оператора містить не один, а декілька операторів, то записувати їх в оператор них дужках не має потреби. На відміну від оператора WHILE обчислення логічного виразу відбувається не до, а після чергового повторення циклу.

Отже, цикл REPEAT обов'язково виконується хоча б один раз, а цикл WHILE може не виконуватися жодного разу.

Конструкцію "цикл-для" (цикл з передумовою) реалізує оператор, що має форму:

```
FOR  < змінна > := < вираз1 > TO < вираз2 > DO < тіло циклу > ;
```

де змінна - змінна порядкового типу, яка набуває послідовно зростаючих значень; вираз1 і вираз2 того ж типу, що і змінна. Виконання розпочинається з обчислення значень вираз1 і вираз2. Далі змінна одержує значення вираз1 і виконується перевірка, чи не перевищує одержане значення змінної вираз2: якщо не перевищує, то виконується тіло циклу. Тоді, коли значення змінної стає неменшим значення вираз2, оператор виконується останній раз.

Можливе використання оператора FOR для випадку, коли змінна приймає послідовно спадаючі значення:

```
FOR < змінна > := < вираз1 > DOWNTO < вираз2 > DO < тіло циклу > ;
```

Для того, щоб такий цикл виконався хоча б раз, значення вираз1 має бути не менше значення вираз2. Наприклад:

```
For C := ' Z ' Downto ' A ' Do WriteLn ( C ) ;
```

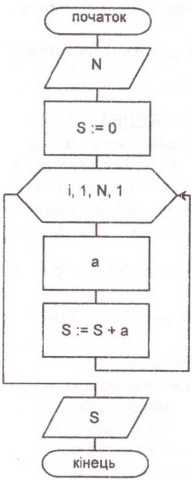
Зауважимо, що в обох випадках (зростання, спадання) параметр циклу може змінюватися тільки з кроком, рівним 1.

Оператор, повторюваний у циклі, сам може бути циклом. Така структура має назву вкладений цикл. У мові Pascal немає обмежень на кількість і глибину вкладення циклів.

ПРАКТИЧНА РОБОТА №3

ПРИКЛАД 6. Знайти суму перших n членів ряду $S = 1 + \frac{1}{2^2} + \frac{1}{3^2} + ... + \frac{1}{n^2}$

І етап. Подамо суму членів даного ряду так: $S = \sum_{i=1}^{n} a_i$, де $a_i = \frac{1}{i^2}$.



мал. 14

У змінній S будемо накопичувати шукану суму. Перед виконанням такого накопичення очистимо область пам'яті S – змінній S присвоїмо 0.

Організуємо цикл з параметром i, що змінюється від 1 до n. Для кожного обороту циклу (та для кожного нового значення i) повторюватимемо такі дії: обчислимо a_i, викличемо з пам'яті поточне значення S, додамо до S обчислене a_i, результат додавання знову присвоїмо S (значення S + a_i відправимо у область пам'яті S).

ІІ етап. Складемо блок-схему (мал. 14).

ІІІ етап. Запишемо текст програми.

```
алг Сума1 (вещ S, цел n)
  арг n
  рез S
нач цел i
    S :=0
    нц для i от 1 до n шаг 1
        S := S + 1 / ( i**2 )
    кц
кон
```

НАМ-програма 6

```
10 REM Сума1
20 INPUT " Введіть значення для n=" ; N
30 S=0
40 FOR I=1 TO N STEP 1
50 S=S+1/(I^2)
60 NEXT
70 PRINT "S="; S
80 END
```

Basic-програма 6

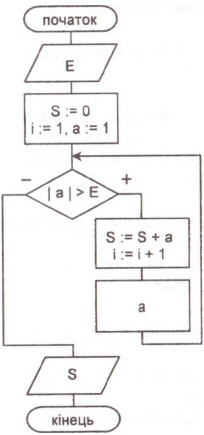
```
Program SUMA1;
  Var
    S : Real;     N, i : Integer;
Begin
  Write ( ' Введіть n= ' ) ; ReadLn  ( N ) ;
  S := 0;
  For i := 1 to N do
     S := S + 1 / Sqr ( i ) ;
  WriteLn ( ' S= ', S :10 :6 )
End.
```

Pascal-програма 6

IV етап. Для $n = 3$ $S = 1 + \frac{1}{4} + \frac{1}{9} = \frac{49}{36} = 1.361111$ ■

ПРИКЛАД 7. Знайти суму членів гармонійного ряду $S = 1 - \frac{1}{2^2} + \ldots + \frac{(-1)^{n+1}}{n^2}$ з точністю E.

I етап. Математична модель. Знайти суму ряду з точністю E означає сумувати члени ряду доти, доки модуль різниці між знайденою сумою S_i (для i членів) та попередньою сумою S_{i-1} (для i-1 члена) більший заданого числа E: $|S_i - S_{i-1}| > E$. Якщо у виразі, що стоїть під знаком модуля звести подібні доданки, то одержимо, що $|a_i| > E$, де a_i - i-й член ряду. Тому оцінюємо лише останній доданок.



мал. 15

II етап. Блок-схема (мал. 15).

III етап. Запишемо текст програми.

```
алг Сума2 (вещ S, E)
  арг E
  рез s
нач цел i, вещ a
    i := 1 ;  a := 1 ;   S := 0
    нц  пока abs (a) > E
        S := S + a
        i := i+1
.       a := ((-1)**(i+1)) / (i**2)
    кц
    S := S
кон
```

НАМ-програма 7

```
10 REM Сума2
20 INPUT "Введіть E = " ; E
30 S=0 : I=1 : A=1
40 IF ABS(A) > E THEN 50 ELSE 90
50 S=S + A
60 I= I + 1
70 A=((-1) ^ (I+1)) / (I^2)
80 GOTO 40
90 PRINT "S=" : S
100 END
```

Basic-програма 7

```
Program SUMA2 ;
  Var
    E, S, A : Real ;
     I, J     : Integer ;
Begin
  Write ( ' Введіть E = ' ) ;
  ReadLn ( E ) ;
  S := 0 ;  I :=1 ;  J:=1 ;  A:=1;
  While Abs(A) > E do
    Begin
      S := S + A;
      I := I+1;  J :=J*(-1) ;  A := J / Sqr(I)
    End;
  WriteLn ( ' S= ', S :10 :6 )
End.
```

Pascal-програма 7

IV етап. $E = 0.05$. Так як $|a_5| < E < |a_4|$, то $S = S_4 = 1 - \frac{1}{4} + \frac{1}{9} - \frac{1}{16} = \frac{115}{144} = 0.798611$ ■

Лабораторна робота №3

Мета: Набути уміння та навички роботи з циклічними програмами.

Матеріальне забезпечення: Е-практикум (MSX-Basic v.1.0, Pascal v.5.5)

Теоретична частина: завдання та контрольні питання.

І рівень.

1. Інформатика як наука. Поняття інформації та повідомлення.
2. Як вимірюється та кодується інформація?
3. Описати покоління ЕОМ.
4. Поняття системи числення. Яка система числення використовується в ЕОМ?
5. Як подається інформація в ЕОМ? У яких одиницях вимірюється пам'ять ЕОМ.
6. Яку логічну схему має ЕОМ?
7. Який алгоритм називається циклічним? Які базові конструкції блок-схем описують роботу циклічних алгоритмів?
8. Яка програма називається циклічною? Які вказівки (оператори) мови реалізують циклічні конструкції алгоритмів?

ІІ рівень.

1. Поняття кодування та декодування інформації. Як здійснюється обробка інформації?
2. Як здійснюється переведення числа з двійкової у десяткову систему числення? Як навпаки? Навести приклади.
3. Навести таблиці правил додавання і множення чисел у двійковій формі.
4. Яку базову конструкцію алгоритму використовують для циклу-поки? Як ще називають такий цикл? Яка вказівка (оператор) програми забезпечує реалізацію вказаного циклу?
5. Яку базову конструкцію алгоритму використовують для циклу-для? Як ще називають такий цикл? Яка вказівка (оператор) програми забезпечує реалізацію вказаного циклу?
6. Яку базову конструкцію алгоритму використовують для циклу-до? Як ще називають такий цикл? Чи є вказівка (оператор) програми, що забезпечує реалізацію вказаного циклу? Яка її форма?

ІІІ рівень.

1. Описати призначення компонентів логічної схеми ЕОМ. Які апаратні пристрої персональної ЕОМ відповідають кожному компоненту схеми?
2. Описати основний алгоритм роботи процесора.
3. Описати правила додавання і множення чисел у двійковій формі. Навести приклади.

Практична частина.

І рівень.

Обчислити суму перших n членів ряду:

$$S=\sum_{i=1}^{n}\frac{2\cos(i-\frac{\pi}{4})(\sin\frac{\pi}{4}\cos i-\sin\frac{\pi}{4}\sin i)}{\sqrt{n+i\cos 2i}}.$$

ІІ рівень.

Протабулювати дану функцію на проміжку від a до b з кроком h.

1-3 варіанти. $y=\begin{cases}\lg(10-x)+2, x<10\\ \dfrac{\sin^2(\pi x)}{14+x}, 10\le x\le 14\\ x+\sqrt{x-14}, x>14\end{cases}$, a=6, b=21, h=3.

4-6 варіанти. $y=\begin{cases}\sin^2 4(x+3)+x, x\le -3\\ \dfrac{x+2}{x^2+x-6}-2, -3<x<2\\ e^{x-3}+x^3-2x-1, x\ge 2\end{cases}$, a= - 6 , b=6, h=2.

7-9 варіанти. $y=\begin{cases}\cos\sqrt{|x+1|}+3, x\le 1.5\\ \dfrac{x+0.75}{\sqrt{x^2-2.25}}, 1.5<x<7\\ \log_4(x+2)+x, x\ge 7\end{cases}$, a= - 4, b=11, h=2,5.

10-12 варіанти. $y=\begin{cases}\sqrt[4]{8-x^3}+5x-10, x\le 2\\ \dfrac{\sin(x-5)^2}{-x^2+10x-16}, 2<x<8\\ \lg(x-7)+2x+5, x\ge 8\end{cases}$, a= - 5, b=13, h=4,5.

13-15 варіанти. $y=\begin{cases}\log_7(8-x)+3x-1, x<8\\ \dfrac{\sqrt{x-8}+x}{x^2-17x+70}, 8\le x<10\\ \sin 2x-\cos 2x+2\sin^2 x, x\ge 10\end{cases}$, a=2, b=14, h=1,5.

ІІІ рівень.

1 варіант. Скориставшись методом прямокутника обчислити інтеграл $I=\int_a^b\frac{dx}{\sqrt{4+x^3}}$. Вхідні дані: a=2, b=14, n=150.

2 варіант. Скориставшись методом трапецій обчислити інтеграл $I=\int_a^b\frac{\cos 2x dx}{\sqrt{|4+x\sin x|}}$. Вхідні дані: a=2, b=14, n=120.

3 варіант. Обчислити суму перших n членів ряду: $S=\sum_{i=1}^{n}\frac{(-1)^{i+1}}{i!}$.

4 варіант. Обчислити n-й член ряду Фібоначі.

5 варіант. На проміжку [-4; 5] знайти кількість коренів рівняння $x^3 \sin x + ax^2 + b\cos x = 0$

6 варіант. С класти програму розкладу числа на прості множники.

7 варіант. На заданому проміжку знайти всі натуральні числа, що дорівнюють кубу суми своїх цифр.

8 варіант. Знайти кількість "щасливих" автобусних квитків у рулоні (номером квитка є шестизначне ціле число).

9 варіант. Задано деяке число *N*. Скласти програму пошуку простих чисел менших за *N* (решето Ератосфена).

10 варіант. Задано деяке число *N*. Скласти програму пошуку "досконалих" чисел менших за *N*. "Досконалим" є число, яке дорівнює сумі своїх дільників, за винятком самого себе.

11 варіант. *N*-кутник задано координатами своїх вершин. Обчислити його площу, використовуючи векторний добуток.

12 варіант. Обчислити значення ірраціонального числа *e* з точністю *E*, використовуючи розклад у гармонійний ряд.

13 варіант. Обчислити значення ірраціонального числа π з точністю *E*, використовуючи периметр вписаного та описаного *n*-кутників.

14 варіант. Знайти корінь рівняння $x^2 - \cos x + 5 = 0$ на відрізку [0, 1] з точністю Eps використовуючи метод хорд.

15 варіант. Знайти корінь рівняння $x^2 - \cos x + 5 = 0$ на відрізку [0, 1] з точністю Eps використовуючи метод дотичних.

Вимоги до захисту лабораторної роботи.

Протокол повинен містити процес розробки математичної моделі (виведення формули) для 3-го рівня, блок-схеми алгоритмів, тексти програм, тести: вхідні дані та результати роботи програми.

2.5. Робота з масивами даних

Розв'язування на ЕОМ багатьох задач пов'язане з обробкою великої кількості числових даних. Ці данні подаються, як правило, в вигляді різноманітних таблиць — масивів, множин.

У програмуванні *під масивами* розуміють певним чином організовану сукупність однотипних елементів, яка може розглядається як єдине ціле. Кожен масив має найменування, а кожен елемент має свій номер (індекс). Тому будь-які дані можна знайти у масиві, знаючи лише ім'я масиву та індекси потрібних елементів.

Масиви відносяться до структурованих типів даних, але завдяки індексуванню з масиву можна виділити будь-який його елемент і використати у програмі як окреме значення раніше визначеного типу. Тобто над елементами масиву можна виконувати такі ж операції як і над змінними та константами.

Одновимірний масив — це лінійно впорядкована сукупність елементів одного і того ж типу. У такому масиві кожен елемент має номер, що складається з одного індексу. Такий масив ще називають вектором.

Двовимірний масив — це масив, що містить елементи, в яких номер складається з двох індексів. Перший індекс вказує номер рядка, а другий – номер стовпця, в якому знаходиться елемент. Двовимірний масив іноді ще називають матрицею. У програмуванні про двовимірний масив говорять як про структуру, що утворена з одновимірного масиву, у якому кожен елемент сам є одновимірним масивом. Подібно до цього задаються тривимірні, чотиривимірні і т.д. масиви.

Для обробки масиву даних часто виникає потреба відшукати певний елемент, впорядкувати елементи за певними ознаками тощо.

Пошук найбільшого елемента масиву.

Нехай до пам'яті ЕОМ введено деякий масив чисел, елементи якого мають імена A(1), A(2), ... , A(N). Серед елементів масиву знайдемо найбільше число та виведемо його на пристрій виводу (екран).

Пропонується така послідовність дій для розв'язання даної задачі:

1. Оберемо змінну для шуканого числа, наприклад, Max. Перед циклічним переглядом елементів масиву спочатку приймемо за шукане максимальне значення перший елемент масиву. Для цього змінній Max присвоїмо значення першого елемента масиву - Max := A(1).

2. Починаючи з наступного елементу масиву будемо порівнювати кожен такий елемент зі значенням Max: якщо елемент більший за Max, то змінній Max надати нового значення – значення даного елемента масиву; інакше значення змінної Max залишити без змін.

Для цього організуємо цикл з параметром *i*, що змінюється від 2 до *n*. У тіло циклу вмістимо конструкцію розгалуження: якщо умова A(*i*)>Max істинна, то Max := A(*i*).

За такими діями забезпечується те, що у змінній Max (у ділянці пам'яті з іменем Max), починаючи з першого присвоєння і до кожного наступного завжди буде міститися найбільше число.

3. Вивести значення змінної Max.

Подібні дії можна виконати і для того, щоб відшукати *найменший елемент масиву*.

Сортування елементів масиву.

Часто у задачах виникає потреба певним чином розташувати елементи масиву, впорядкувати їх – у порядку зростання або спадання. Така робота називається сортуванням масиву.

Алгоритм *сортування вибором* такий:

1. Встановити найменший елемент масиву та його номер.

2. Поміняти місцями знайдений найменший і перший елементи.

3. Виконати пункти 1 і 2 над залишком масиву (масивом без першого елемента). Пункт 3 повторювати доти, доки кількість елемен-

тів у залишку масиву не скоротиться до одного.

При виконанні п.2. для обміну місцями елементів масиву використовують проміжну змінну. Наприклад,

а) проміжній змінній присвоюють значення першого елемента;

б) першому присвоюють значення найменшого елемента;

в) найменшому елементу присвоюють значення змінної.

На прикладі масиву з п'ятьох елементів проілюструємо зміну їх порядку у міру повторення пункту 3 (сортування за зростанням):

40 50 10 20 30
10 50 40 20 30
10 20 40 50 30
10 20 30 50 40
10 20 30 40 50

При кожному повторі пункту 3 найменший елемент області пошуку переміщується у відсортовану частину масиву, за рахунок цього упорядкована частина масиву росте, а неупорядкована (підкреслена лінією) скорочується на 1 елемент.

Алгоритм *обмінного сортування (метод бульбашки)* теж складається з окремих кроків. На кожному кроці проходять масив від початку до кінця, порівнюючи пари сусідніх елементів. Якщо чергова пара порушує потрібний порядок, її елементи міняють місцями. Кроки повторюють доти, доки черговий прохід не викликає жодного обміну.

Розглянемо як змінюється значення масиву з п'ятьох елементів (30, 20, 10, 50, 40) на кожному кроці сортування за зростанням:

Вихідне значення: 40 50 10 20 30.

Після 1-го кроку : 40 10 20 30 50.

Після 2-го кроку : 10 20 30 40 50.

Навчальна алгоритмічна мова.

У НАМ елементи масивів даних задаються як елементи таблиць. Таблиці спочатку описуються: вказується їх тип, ім'я та розмірність.

В Е-практикумі для описання типу використовуються службові слова **вещтаб, целтаб**. Тип **литтаб** відсутній (не можна задавати таблиці, що складаються з рядків). Після службового слова записують ім'я таблиці, за яким у квадратних дужках вказують розмірність: номер першого елемента, символ двокрапка, номер останнього елемента.

Наприклад:

вещтаб A [1 : 10] – одновимірний масив (10 елементів);

целтаб B [3:7, 1:10] - двовимірний масив (50 елементів).

Доступ до окремого елемента масиву здійснюється шляхом індексування. Наприклад,

A [1], B [3, 1] – перші елементи описаних масивів.

Basic.

На мові Basic для опису масивів використовують оператор DIM, в якому вказують ім'я масиву, тип та в круглих дужках розмірність – максимальні значення індексів. Наприклад:

DIM A (20) – одновимірний масив дійсних чисел з 21 елементом;

DIM B% (4,10) – двовимірний масив цілих чисел з 55 елементами.

Доступ до окремого елемента масиву здійснюється шляхом індексування. Наприклад:

A(1) – другий елемент масиву A;

B (4, 10) – останній елемент масиву B.

Якщо ж наперед відомо, що кількість елементів масиву не буде перевищувати 10 (індекси від 0 до 9), то такий масив можна спеціально не описувати.

Якщо при виконанні програми на певному етапі деякий масив стає зайвим або не вистачає оперативної пам'яті для роботи програми, то можна звільнити зарезервовану під масив пам'ять за допомогою оператора ERASE, вказавши ім'я масиву. Наприклад:

ERASE A - вилучити масив A;

ERASE A, B% - вилучити масиви A і B.

На етапі налагодження програми з масивами даних зручно використовувати оператори DATA, READ, RESTORE.

Pascal.

Для описання масивів використовується зарезервоване слово array.

Наприклад, опишемо одновимірний масив A, який може містити 10 елементів – дійсних чисел, та двовимірний масив B, який може містити 5x10 елементів – цілих чисел:

```
Type
   dataA    = array [ 1 .. 10 ]  of real ;
   masiv    = array [ 3 .. 7, 1 .. 10 ] of integer ;
Var
   A  : dataA ;
   B  : masiv ;
```

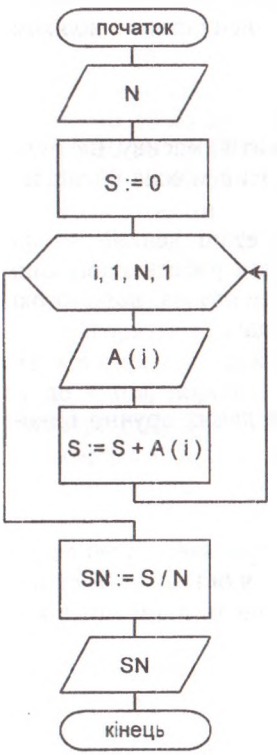
Доступ до окремого елемента масиву здійснюється шляхом індексування. Наприклад, A [1], B [3, 1] – перші елементи описаних масивів.

Якщо деякі масиви ідентичні за структурою та типом елементів, то над такими масивами як над послідовностями можна виконувати операції відношення "дорівнює", "не дорівнює" та операцію присвоєння (змінній-масиву можна присвоїти існуючий масив раніше введених елементів), тобто ці операції можна виконувати над масивами в цілому.

Арифметичні операції, операції введення та виведення елементів не можна виконувати над послідовностями, а отже не можна їх виконувати і над масивами в цілому – вказані операції можна виконати тільки над окремими елементами.

ПРАКТИЧНА РОБОТА № 4

ПРИКЛАД 8. Знайти середнє арифметичне лінійної таблиці, що містить десять елементів.



мал. 16

I етап. Дана лінійна таблиця – одновимірний масив певної розмірності. Спочатку потрібно вказати розмірність N масиву, а потім ввести елементи масиву. Значення елементів масиву вводяться у циклі (Basic, Pascal) або записуються у розділі рез (НАМ). Середнє арифметичне для масиву – це частка суми всіх його елементів та їх кількості. Щоб обчислити суму елементів масиву необхідно використати цикл та виконати дії, аналогічні прикладу на стор. 44. Тому операції з введення елементів масиву та сумування об'єднаємо в тілі одного циклу (Basic, Pascal).

II етап. Складемо блок-схему (мал. 16).

III етап. Запишемо програму.

```
алг An (цел n, вещтаб A [1:10], вещ Sn)
  арг n, An = 10, 0,1,2,3,4,6,7,8,9,10
  рез Sn
нач вещ S, цел i
    S :=0
    нц для i от 1 до n шаг 1
       S := S + A [ i ]
    кц
    Sn := S / n
кон
```

НАМ-програма 8

```
10 REM Сер. арифметичне
20 Input "Кількість елементів" ; N
30 DIM A (N) : S = 0
40 FOR i =1 TO N
50 INPUT A ( i )
60 S=S + A ( i )
70 NEXT
80 SN = S / N
90 PRINT "Сер. Ар. = " ; SN
100 END
```

Basic-програма 8

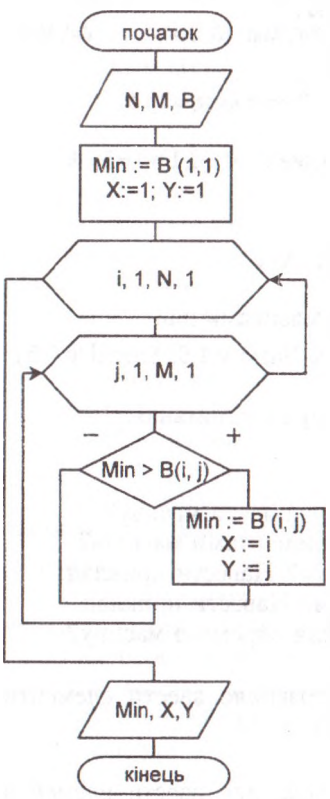
```
Program An ;
  Var
    S, Sn : Real ;    i, N      : Integer ;
    A      : Array [ 1..10 ] of Real ;
Begin
  ReadLn (N) ; S := 0 ;
  For i := 1 to N do
     Begin
       Read ( A [ i ] ) ; S := S + A [ i ]
     End ;
  Sn := S / N ;
  WriteLn ( ' Сер. ар. = ' , Sn :10 : 6 )
End.
```

Pascal-програма 8

IV етап. N=10, таблиця {0,1,2,3,4,6,7,8,9,10}. Середнє арифметичне- 5. ■

ПРИКЛАД 9. Для двовимірного масиву цілих чисел знайти мінімальний елемент та його індекси.

I етап. Задання розмірності масиву, введення елементів масиву здійснимо аналогічно до попереднього прикладу, тому описувати окремо послідовність таких дій не будемо, а відповідні записи внесемо до блоку введення. Пошук мінімального елементу масиву аналогічний пошуку максимального елемента, розглянутому вище (стор. 49).



мал. 17

Зауважимо лише, що у задачі дано двовимірний масив, тому спочатку перший елемент масиву приймаємо за мінімальний (Min:=B(1,1)), а потім для перегляду елементів масиву використовуємо два цикли: цикл з параметром *j*, що вкладений у цикл з параметром *i*. Це дозволяє для кожного значення *i* проглянути всі значення *j*. За параметром *j* забезпечується зміна другого індексу (номер стовпчика), а за параметром *i* – першого індексу (номер рядка).

II етап. Блок-схема (мал. 17).

III етап. Запишемо програму.

```
алг Min2 (целтаб B [1:2,1:3], цел min, x, y, N, M)
  арг N, M, B = 2, 3, 0, -5, 1, 4, 2, 1
  рез min, x, y
нач  цел i, j
  min := B [1,1]
  x :=1
  y :=1
  нц  для i от 1 до N шаг 1
  .    нц  для j от 1 до M шаг 1
  .    .  если Min > B [ i, j ]
  .    .  то  Min := B [ i, j ]
  .    .          x := i ;
  .    .          y := j
  .    .  все
  .    кц
  кц
  Min := Min
  x := x
  y := y
кон
```

НАМ-програма 9

```
10 REM Мінімальне
15 INPUT N, M
20 DIM B(N,M)
40 FOR I=1 TO N
50 FOR J=1 TO M
60 INPUT B(I, J)
70 NEXT J
80 NEXT I
90 MIN=B(1,1): X=1: Y=1
100 FOR I=1 TO N
110 FOR J=1 TO M
120 IF MIN > B (I, J) THEN MIN=B(I, J)
: X=I : Y=J
130 NEXT J
140 NEXT I
150 PRINT " Min= " ; MIN
160 PRINT "індекси"; X , Y
160 END
```

Basic-програма 9

```
Program Minimum;
  Const
    N = 2 ; M = 3 ;
  Var
    Min, I, J, X, Y    : integer ;
    B     : array [ 1..N, 1..M ] of integer ;
Begin
  For I := 1 to N do
     For J := 1 to M do
        Read ( B [ I, J ] ) ;
  Min := B [1,1] ;  X:=1 ; Y:=1 ;
  For I := 1 to N do
     For J := 1 to M do
        If Min > B [ I, J ] Then
           Begin
             Min := B [ I, J ] ; X:=i ; Y:=J
           End ;
  WriteLn('Min', Min :10, 'X=', X :3, 'Y=', Y :3 )
End.
```

Pascal-програма 9

IV етап. N=2, M=3, масив $\begin{Bmatrix} 0 & 1 & 2 \\ -5 & 4 & 1 \end{Bmatrix}$. Мін. елемент B (2,1)= −5. ■

Лабораторна робота №4

Мета: Набути уміння та навички роботи з масивами даних.

Матеріальне забезпечення: Е-практикум (MSX-Basic v.1.0, Pascal v.5.5)

Теоретична частина: завдання та контрольні питання.

І рівень.

1. Поняття масиву даних.
2. Як здійснюється доступ до окремого елемента масиву?
3. Які операції можна виконувати над елементами масивів?
4. Який масив називається одновимірним? Навести приклад.
5. Який масив називається двовимірним? Навести приклад.
6. Які операції найчастіше виконують для окремого масиву?
7. Як описуються масиви у програмі?
8. Навести фрагмент блок-схеми, що дозволяє ввести елементи одновимірного масиву з іменем A та розмірністю M.

ІІ рівень.

1. Навести фрагмент блок-схеми, що дозволяє ввести елементи двовимірного масиву з іменем A та розмірністю MxN.
2. Описати алгоритм пошуку найбільшого елемента масиву.
3. Описати алгоритм сортування вибором.
4. Описати алгоритм обмінного сортування.

III рівень.

1. Навести фрагмент блок-схеми, що дозволяє ввести елементи тривимірного масиву з іменем A та розмірністю MxNxP.
2. Які є види сортувань даних у залежності від того, який тип пам'яті використовується для зберігання елементів масивів? У чому полягають проблеми оптимізації для таких сортувань?
3. Описати алгоритм сортування елементів масиву за допомогою дерева.
4. Описати алгоритм пірамідального сортування.
5. У чому відмінність різних алгоритмів сортування?
6. Поняття пошуку. У чому суть задачі "комівояжера"?

Практична частина.

I рівень.

Знайти кількість додатних, від'ємних та нульових елементів лінійного цілочисельного масиву A(n).

II рівень.

Модифікувати програму першого рівня, забезпечивши виведення на екран даного масиву A(n) та масиву B(n), що утвориться у результаті виконання завдання, вказаного у варіанті.

1-3 варіанти. Впорядкувати масив за зростанням, використовуючи обмінне сортування.

4-6 варіанти. Впорядкувати масив за спаданням, використовуючи сортування вибором.

7-9 варіанти. Впорядкувати масив так: спочатку розмістити всі нульові елементи масиву, потім додатні і в кінці від'ємні.

10-12 варіанти. Розмістити елементи таблиці, змінивши порядок їх розташування на зворотній.

13-15 варіанти. Використовуючи сортування методом бульбашки, впорядкувати масив за неспаданням модулів його елементів.

III рівень.

Використовуючи завдання I та II рівнів утворити двовимірний масив та виконати завдання згідно з варіантом.

1 варіант. Утворити масив C(n x n), в якому перший рядок – це елементи масиву A, другий – елементи масиву B, а елементи, що знаходяться в кожному стовпчику утворюють геометричну прогресію. Вивести на екран масив C та його мінімальний елемент, що знаходиться над головною діагоналлю.

2 варіант. Утворити масив C(n x n), в якому перший рядок – це елементи масиву A, другий – елементи масиву B, а елементи що знаходяться в кожному стовпчику утворюють арифметична прогресію. Вивести на екран масив C та його максимальний елемент, що знаходиться під головною діагоналлю.

3 варіант. Утворити масив C(n x n), в якому перший рядок — це елементи масиву A, другий – елементи масиву B, а кожен *i*-й (*i*=3, 4, ... n) елемент стовпчика дорівнює сумі *i-1* та *i-2* елементів цього ж стовпчика. Вивести на екран утворений масив та середнє арифметичне елементів його головної діагоналі.

4 варіант. Утворити масив C(m x n), в якому перший рядок — це елементи масиву A, другий – елементи масиву B, а кожен *i*-й (*i*=3, 4, ... m) елемент стовпчика дорівнює добутку *i-1* та *i-2* елементів цього ж стовпчика. Вивести на екран утворений масив та його мінімальний додатній елемент.

5 варіант. Утворити масив C(m x n), в якому перший рядок — це елементи масиву A, другий – елементи масиву B, а кожен інший елемент стовпчика дорівнює середньому арифметичному всіх попередніх елементів, що знаходяться в одному і тому ж стовпчику. Вивести на екран утворений масив та його максимальний від'ємний елемент.

6 варіант. Утворити масив C(m x n), в якому перший рядок — це елементи масиву A, другий – елементи масиву B, а кожен інший елемент стовпчика дорівнює сумі попередніх додатних елементів, що знаходяться в одному і тому ж стовпчику. Вивести на екран утворений масив та його максимальний від'ємний елемент.

7 варіант. Прийнявши масив A за матрицю A(1 x n), а масив B за матрицю B(n x 1), утворити матрицю C=A x B. Вивести на екран утворену матрицю та елементи її головної діагоналі.

8 варіант. Вивести на екран масив C(n x n), елементи якого утворюються з елементів масивів A і B так: C(*i, j*) це перше просте число з інтервалу, що обмежений числами A(*i*) та B(*j*); якщо таких чисел на цьому інтервалі немає, тоді C(*i, j*) дорівнює 0. Підрахувати кількість нулів, що входить до складу утвореного масиву.

9 варіант. Вивести на екран масив C(n x n), елементи якого утворюються з елементів масивів A і B так: C(*i, j*) дорівнює 1, якщо A(*i*)=B(*j*); C(*i, j*) дорівнює -1, якщо A(*i*)=−B(*j*); і, якщо не виконується жодна з попередніх умов, C(*i, j*) дорівнює 0. Підрахувати кількість нулів, що містить головна діагональ утвореного масиву.

10 варіант. Утворити масив C(n x n), елементи якого обчислюються наступним чином: C(*i, j*) це A(*i*) в степені B(*j*). Вивести на екран утворений масив.

11 варіант. Вивести на екран масив C(n x n), де кожен елемент C(*i, j*) - це або число A(*i*) або число B(*j*), у залежності від того, модуль якого з даних двох чисел більший. Замінити всі додатні елементи над головною діагоналлю на нульові. Вивести на екран змінений масив.

12 варіант. Утворити масив C(n x n), елементи якого обчислюються з елементів масивів A і B так: C(*i, j*) це модуль комплексного числа, у якого A(*i*) – дійсна частина, B(*j*) – уявна частина. Вивести на екран утворений масив та максимальний елемент його головної діагоналі.

13 варіант. Вивести на екран масив C(n x n), елементи якого утворюються з елементів масивів A і B так: C(*i, j*) - менше з двох чисел A(*i*) та B(*j*). Модифікувати утворений масив за правилом: елементи *i*-ого рядка стають елементами *j*-ого стовпчика, і навпаки — елементи *j*-ого стовпчика стають елементами *i*-ого рядка. Вивести на екран модифікований масив.

14 варіант. Вивести на екран масив C(n x n), елементи якого утворюються з елементів масивів A і B так: C(*i, j*) це перше число кратне 5 з інтервалу, що обмежений числами A(*i*) та B(*j*); якщо таких чисел на цьому інтервалі немає, тоді C(*i, j*) дорівнює A(*i*)+B(*j*). Підрахувати кількість додатних елементів, що входить до складу утвореного масиву.

15 варіант. Утворити масив C(m x n), в якому перший рядок — це елементи масиву A, другий – елементи масиву B, а кожен *i*-й (*i*=3, 4, ... n) елемент стовпчика дорівнює залишку від ділення *i-1* на *i-2* елементів цього ж стовпчика. Вивести на екран утворений масив та його мінімальний елемент.

Вимоги до захисту лабораторної роботи.

Для певного рівня знати відповіді на питання та виконати завдання теоретичної частини. Протокол лабораторної роботи повинен містити назву теми, процес розробки математичної моделі (виведення формули) для 3-го рівня, блок-схеми алгоритмів, тексти програм, тести: вхідні дані та результати роботи програми.

2.6. Робота з рядковими та символьними величинами

Крім чисел, мови програмування можуть оперувати з окремими словами, наборами слів (реченнями, текстами). При цьому слова та їх набори необов'язково повинні мати смисл з точки зору людини – це можуть бути будь-які послідовності символів.

Під *рядковою величиною* розуміють значення константи (змінної), яке складається з будь-якої послідовності символів кодової таблиці ЕОМ. Для такої величини кількість символів може бути від 0 до 255.

Під *символьною величиною* розуміють значення константи (змінної), яке складається з будь-якого, але тільки одного символу кодової таблиці комп'ютера.

Над рядковими та символьними величинами можна виконувати такі операції як склеювання, визначення довжини величини, виділення (вирізка) частини символів з цілої послідовності тощо.

Навчальна алгоритмічна мова.

У НАМ рядкові та символьні величини не розрізняються, а використовується лише рядковий тип величин, який позначається службовим словом лит. Тому ще іноді про такі величин у НАМ говорять як про літерні. Рядкові величини беруться у лапки, наприклад: "так", "ні", "інформатика", "12йокCKEBWLKSM=+//$", "1", "10_A" тощо.

У тілі програми значення змінних та констант рядкового типу заключають у лапки. Наприклад, y := "вперед".

У Е-практикумі максимальна кількість символів (довжина), що входять до складу рядкової величини, рівна 16.

Основні операції:

- *додавання (склеювання)*:
  нехай x := "фіз", y := "мат", z := x + y, тоді z набуде значення "фізмат"
  (зауважимо, що на результат впливає порядок доданків);
- *визначення довжини (кількості символів)*:
  нехай y := длин ("університет"), тоді y набуде значення 11;
  нехай y := длин ("математика і фізика"), тоді y набуде значення 19;
- *вирізка*, за допомогою якої можна виділити частину рядка, вказавши номер першого та останнього символів:
  нехай y := "велосипед", z := y [7:9], k := y [4:7], тоді z набуде значення "пед", а k набуде значення "осип".

Basic.

У MSX-Basic рядкові та символьні величини не розрізняються, а використовується лише рядковий тип величин, який для змінної можна оголосити у два способи: дописати до імені змінної символ $ або використати оператор DEFSTR. Наприклад,

```
A$, B1$, C$ (10), WORD$
DEFSTR A, B1, C (10), WORD.
```

У програмі значення констант та змінних рядкового типу заключають у лапки. Наприклад, Y := "вперед".

Операції та деякі функції MSX-Basic, що використовуються для роботи з рядковими величинами:

- + - додавання (склеювання):
  нехай X = "фіз", Y = "мат", Z = X + Y, тоді Z набуде значення "фізмат";
- LEN (A$) - визначає довжину рядкової величини A$ (кількість символів):
  нехай Y = LEN ("університет"), тоді Y набуде значення 11;
  нехай X$ = "математика і фізика", Y = LEN (X$), тоді Y набуде значення 19;
  нехай Y = LEN (""), тоді Y набуде значення 0;
- MID$ (A$, M [, N]) – виділяє з A$ частину рядка довжиною N символів, починаючи з позиції M. За відсутності параметра N виділяється частина рядка, починаючи з позиції M і до кінця рядка:
  нехай Y$ = MID$ ("університет", 9, 3), тоді Y$ набуде значення "тет";

нехай X$ ="123456789", Y$ = "університет", MID$ (X$, 4, 3) = Y$, тоді X$ набуде значення "123уні789";

- LEFT$ (X$, N) – виділяє з X$ частину рядка довжиною N символів, починаючи з крайнього лівого символу;
- RIGHT$ (X$, N) – виділяє з X$ частину рядка довжиною N символів, починаючи з крайнього правого символу;
- STRING$ (N, X$) – формує рядок довжиною N, утворений першим символом X$. Другим аргументом даної функції може бути ціле число X (0≤X≤255) – код ASCII певного символу; тоді рядок формується з символу, код якого подано вказаним числом;
- SPACE$ (N, X$) – формує рядок символів-пропусків довжиною N;
- INSTR ([N,] X$, U$) – дає число, рівне номеру позиції, починаючи з якої частина рядка (U$) входить до всього рядка (X$) (при цьому відлік позицій ведеться з N-ої позиції). За відсутності параметра N відлік позицій починається з першої позиції;
- STR$ (A) – перетворює десяткове число A у рядок символів, який має форму десяткового числа;
- VAL (X$) – перетворює символьне подання X$ десяткового числа у числовий тип, тобто значенням цієї функції є десяткове число.

Pascal.

Символьні величини описуються як тип даних char. У програмі значення змінних і констант цього типу мають бути поміщені у апострофи, наприклад:

```
Var
  Simb : char ;
...
  Simb := 'A' ;
```

Рядкові величини описуються як тип даних string (див. стор. 14). Рядок по іншому можна розглянути як масив, компоненти якого мають тип char і тип індекса має нижню межу рівну 1, тобто тип string описується як array [1..255] of char. При використанні у програмі значення рядкових констант та змінних поміщаються у апострофи, наприклад

```
Var
  Name: string ;
...
  Name := ' Шевченко Тарас Григорович ' ;
```

Над рядковими даними допустимі операція зчеплення ("склеювання") та операції відношення. Наприклад,

нехай X := 'фіз', Y := 'мат', Z := X + Y, тоді Z набуде значення 'фізмат' ;

умова ' стіл ' <= ' стілець ' є істинна (true) ;

умова ' 12 ' > ' 2 ' є хибна (false) .

Серед усіх можливих значень рядків є порожній рядок. Він зображується двома одинарними лапками, між якими нічого немає.

Для того, щоб символ "апостроф" входив до складу рядка, у запису рядка цей символ повторюють двічі. Наприклад,

Write (' ім ' ' я ') виведе на екран рядок ім'я.

Стандартні функції та процедури для роботи з рядками:

- Concat (S1, S2, ... Sn : *string*) : *string* — виконує послідовне зчеплення рядків S1, S2, ... Sn. Результуючий рядок не повинен перевищувати 255 символів.
- Copy (St : *string*, Poz : *integer*, N : *integer*) : *string* - виділяє з рядка St фрагмент довжиною N символів, починаючи з позиції Poz;
- Length (St: *string*) : *integer* — повертає довжину рядка;
- Pos (Subst : *string*, St: *string*): *byte* - повертає позицію, з якої рядок Subst перший раз зустрічається в рядку St.
- Delete (St : *string*, Poz: *integer*, N : *integer*) — з рядка St вилучає N символів, починаючи з позиції Poz;
- Insert (SubSt : *string*, S: *string*, Poz: *integer*) - вставляє рядок SubSt у рядок S, починаючи з позиції Poz;

Для перетворення чисел у рядок і навпаки служать процедури:

- Str (X, St) - перетворює числове значення величини X у рядок St. Після X може записуватися формат, аналогічний формату виводу оператора Write;
- Val (St : string, V, Code : integer) - перетворює рядок St у значення числової змінної V. Рядок St повинний бути коректним записом числа. Якщо це не так, V = 0, Code <> 0. У випадку вдалого перетворення Code = 0.

До окремого символу рядка можна звернутися, якщо у квадратних дужках після імені вказати індекс. Наприклад, до п'ятого символу змінної Name можна звернутися так: Name [5]. Якщо звернутися до нульового індексу рядкової змінної, то одержимо кількість символів у рядку.

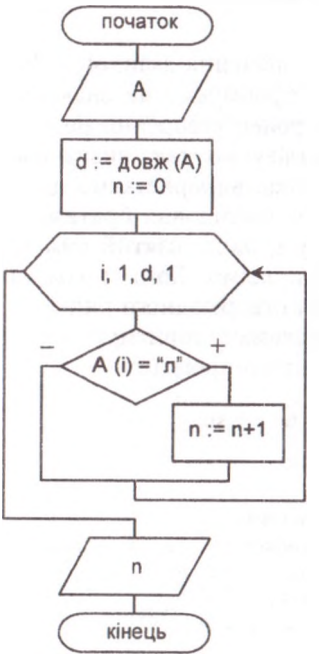
ПРАКТИЧНА РОБОТА № 5

ПРИКЛАД 10. Визначити кількість літер "л", які містить даний рядок символів.

І етап. Спочатку визначимо довжину (кількість символів) для даного рядка. Довжину рядка використаємо як кінцевий параметр циклу. У циклі послідовно, починаючи з першого, братимемо по одному символу і перевірятимемо: якщо вказаний символ рівний літері "л", то лічильник кількості літер "л" збільшуємо на одиницю. Зауважимо, що на початку роботи програми лічильнику потрібно присвоїти значення 0 (n := 0).

ІІ етап. Складемо блок-схему алгоритму (мал. 18).

III етап. Запишемо текст програми.



мал. 18

```
алг Буква ( цел n, лит A)
  арг A
  рез n
нач  цел d, i
     d := длин ( A )
     n := 0
     нц для i от 1 до d шаг 1
          если A [ i : i] = "л"
            то n := n+1
          все
     кц
  n := n
кон
```

НАМ-програма 10

```
10 REM Кількість літер "л"
20 INPUT "Введіть рядок символів" ; A$
30 N=0
40 D=LEN(A$)
50 FOR i = 1 TO D
60 IF MID$ ( A$, i, 1) = "л" THEN N=N+1
70 NEXT
80 PRINT "Результат" ; N
90 END
```

Basic-програма 10

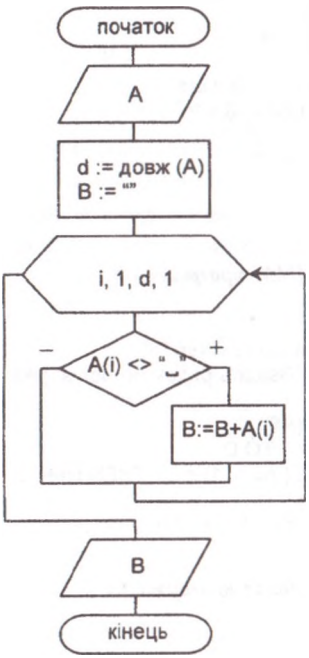
```
Program Kol_Liter ;
  Var
     i, n, d : integer ;
     A : string ;
Begin
  WriteLn ( ' Введіть рядок символів ' ) ;
  ReadLn ( A ) ;
  d := Length ( A ) ;
  n := 0 ;
  For i := 1 to d do
     If Copy (A, i, 1) = ' л ' Then n := n+1 ;
  WriteLn ( ' Результат ', n :10)
End.
```

Pascal-програма 10

IV етап. Тест1. Рядок: "лелека летить". Результат n=3.
Тест2. Рядок: "інформатика". Результат n=0. ■

ПРИКЛАД 11. У рядку символів видалити всі пропуски.

І етап. Вводимо даний рядок символів як значення змінної A. Результуючий рядок сформуємо як значення нової змінної B. Процес утворення результуючого рядка реалізуємо так: визначимо довжину даного рядка; використаємо цикл, за допомогою якого послідовно братимемо по одному символу i, якщо взятий символ не пропуск, то "приклеїмо" його справа до поточного значення створюваного рядка.



мал. 19

ІІ етап. Блок-схема алгоритму.

ІІІ етап. Запишемо програму.

```
алг Пропуск (лит A, B)
  арг A
  рез B
нач  цел d, i
    d := длин ( A )
    B := ""   {присвоїти порожній рядок}
    нц  для i от 1 до d шаг 1
        если A [ i : i ] <> "␣"
          то B := B +  A [ i : i ]
        все
    кц
  B := B
кон
```

НАМ-програма 11

```
10 REM Видалення пропусків
20 INPUT " Введіть рядок " ; A$
30 B$ = ""{присвоїти порожній рядок}
40 D=LEN ( A$ )
50 FOR i = 1 TO D
60 Y$=MID$( A$, i, 1)
70 IF Y$ <> "␣" THEN B$=B$+Y$
80 NEXT
90 PRINT "Результат" ; B$
100 END
```

Basic-програма 11

```
Program Del_Space ;
  Var
    i, D     : integer ;
    A, B, Y : string ;
Begin
  WriteLn (' Введіть рядок ') ; ReadLn (A) ;
  B := ' ' ;   {присвоїти порожній рядок}
  D := Length ( A ) ;
  For i := 1 to D do
      begin
        Y := Copy (A, i, 1);
        If Y <> '␣' Then B := Concat ( B, Y )
      end ;
  WriteLn ( ' Результат ', B )
End.
```

Pascal-програма 11

IV етап. Тест1. Рядок: "␣фіз␣мат␣". Результат: "фізмат".
Тест2. Рядок: "фіз.-мат.". Результат: "фіз.-мат.".■

Лабораторна робота №5

Мета: Набути уміння та навички роботи з рядковими та символьними величинами.

Матеріальне забезпечення: Е-практикум (MSX-Basic v.1.0, Pascal v.5.5)

Теоретична частина: завдання та контрольні питання.

І рівень.

1. Поняття рядкової та символьної величин.
2. Як описуються рядкові та символьні величини?
3. Яка вказівка програми дозволяє виконати операцію склеювання рядкових (символьних) величин? Навести приклад.
4. Яка вказівка програми дозволяє визначити кількість символів рядкової величин? Навести приклад.
5. Яка вказівка програми дозволяє виконати операцію вирізки (копіювання)? Навести приклад.
6. Описати алгоритм, що дозволяє підрахувати скільки разів певний символ зустрічається у даному тексті.

ІІ рівень.

1. Описати алгоритм, що дозволяє підрахувати кількість слів у даному тексті.
2. Описати алгоритм, за яким перевіряється, чи дане слово є словом-"паліндромом" (слово однаково читається зліва направо і справа наліво).
3. Описати оператори, що дозволяють визначити позицію, починаючи з якої певний символ (слово) зустрічається у даному рядку символів (Basic, Pascal).
4. Описати оператори, що дозволяють подати число як рядок символів та виконати обернену операцію – перетворити рядкове подання числа у число (Basic, Pascal).

ІІІ рівень.

1. Яке кодування символу називають ASCII кодом? Які існують інші способи кодування символів? За якими правилами символи можна перекодувати у десятковий код? Чи має мова програмування оператори, що дозволяють перекодувати символи?
2. Описати алгоритм підрахунку кількості слів-"паліндромів" у даному тексті.
3. Які оператори, крім вказаних у питаннях рівнів 1 та 2, має мова програмування? Описати їх призначення (Basic, Pascal).
4. Які є можливості для забезпечення роботи з текстом, кількість символів якого перевищує 255 (Basic, Pascal)?

Практична частина.

І рівень.

Підрахувати, скільки слів містить даний текст. Врахувати, що слова тексту можуть відокремлюватися одне від одного пропуском[1], комою та пропуском, двокрапкою та пропуском, крапкою з комою та пропуском, крапкою з пропуском, тире. Текст повинен закінчуватися крапкою.

ІІ рівень.

Модифікувати програму І рівня з урахуванням завдання, що вказане у варіанті.

1-3 варіанти. Який відсоток слів даного тексту починаються з певного символу.

4-6 варіанти. Який відсоток слів даного тексту закінчуються певним символом.

7-9 варіанти. Який відсоток слів даного тексту відокремлюється лише пропуском.

10-12 варіанти. Який відсоток слів даного тексту знаходяться в першій його половині.

13-15 варіанти. Який відсоток слів даного тексту знаходяться у його другій третині.

ІІІ рівень.

Модифікувати програму ІІ рівня з урахуванням завдання, що вказане у варіанті.

1 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 50%, знайти найдовше слово у даному тексті, інакше замінити всі літери "а" на "б" та "б" на "а".

2 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 60%, знайти кількість слів даного тексту, що починаються та закінчується одним і тим же символом, інакше замінити всі слова "кіт" на "котик".

3 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 35%, перевірити баланс відкритих та закритих дужок, інакше замінити всі подвійні пропуски одним пропуском.

4 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 20%, замінити всі слова "на" на "дай" та "дай" на "нема", інакше розташувати всі слова тексту в алфавітному порядку.

5 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 55%, знайти кількість чисел у даному тексті, інакше поміняти місцями перше та останнє слово тексту.

---

[1] Підряд може бути декілька пропусків.

6 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 40%, підрахувати кількість слів, що зустрічаються у тексті принаймні двічі, інакше розташувати всі слова даного тексту у порядку зростання їх довжини.

7 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 70%, підрахувати кількість різних символів, що входить до складу даного тексту, інакше замінити всі цифри так: 0 на 9, 1 на 8, ..., 8 на 1, 9 на 0.

8 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 25%, знайти найбільшу ділянку тексту, яка є "паліндром", інакше знищити всі круглі дужки.

9 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 35%, знайти символ, що найчастіше зустрічається у тексті, інакше розташувати всі слова даного тексту в зворотному порядку.

10 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 55%, знайти символ, яким найчастіше починаються слова у тексті, інакше розташувати всі слова даного тексту у порядку спадання їх довжини.

11 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 65%, замінити всі слова даного тексту на обернені, не порушивши їх порядок у тексті, інакше знайти найкоротше слово.

12 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 15%, розташувати всі слова даного тексту в порядку зростання їх довжини, інакше — замінити буквосполучення 'огова' на 'та'.

13 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 20%, поміняти місцями перший та останній символи кожного слова у даному тексті, інакше знайти всі слова-"паліндроми".

14 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 45%, замінити всі слова даного тексту на обернені, не порушивши їх порядок у тексті, інакше замінити всі цифри так: 0 на 9, 1 на 8, ..., 8 на 1, 9 на 0.

15 варіант. У залежності від отриманого відсотка виконати наступне: якщо отриманий відсоток менше 55%, знайти кількість слів даного тексту, що починаються та закінчується одним і тим же символом, інакше знищити всі круглі дужки.

Вимоги до захисту лабораторної роботи.

Протокол лабораторної роботи повинен містити назву теми, математичну модель, блок-схеми алгоритмів, тексти програм, тести: вхідні дані та результати роботи програми.

2.7. Використання підпрограм та функцій користувача

Під *основним алгоритмом* будемо розуміти алгоритм, використання якого забезпечує розв'язування даної (основної) задачі.

Іноді при повному розв'язування основної задачі можна виділити деякий її фрагмент – окрему допоміжну задачу, яка або раніше була розв'язана користувачем, або розв'язок якої є відомим (типова задача, чи задача розв'язана іншими користувачами). Врахуємо й те, що допоміжна задача може використовуватися декілька разів при розв'язуванні основної задачі. Для допоміжної задачі окремо від основного алгоритму може розроблятися новий алгоритм або використовуватися раніше описаний. Про *алгоритм розв'язку допоміжної задачі говорять як про допоміжний* по відношенню до алгоритму розв'язку основної задачі.

Таким чином, розроблені користувачем алгоритми можна використовувати для певних задачах як основні, а для інших – як допоміжні.

У мовах програмування розрізняють два види допоміжних алгоритмів: алгоритми – функції користувача та алгоритми – процедури користувача. *Функція користувача* – це незалежна іменована частина програми, яку можна викликати для виконання певних дій. Вона дозволяє знайти лише одне значення, передати його у точку виклику, і ім'я функції може бути частиною виразу (входити до виразу як операнд). *Процедура* аналогічна функції, але процедура не може бути частиною виразу (операндом), і після виконання процедури завжди є можливість використати у програмі не тільки один, а декілька результатів її роботи. Про програми, у залежності від того, який алгоритм вони подають, іноді говорять як про основні та допоміжні (підпрограми).

Розбиваючи задачу на частини і формуючи відокремлені модулі як процедури та функції, програміст реалізує основні принципи структурного програмування.

Навчальна алгоритмічна мова.

Для того, щоб у Е-практикумі скористатися допоміжним алгоритмом необхідно після конструкції основного алгоритму вставити нову конструкцію (комбінація клавіш <Esc + a>) Така конструкція розглядатиметься системою як допоміжний алгоритм тільки тоді, коли текст основної програми містить вказівку з назвою допоміжного алгоритму.

Якщо *допоміжний алгоритм є процедурою*, то у основному алгоритмі при посиланні на процедуру у дужках вказуються вхідні та вихідні дані (аргументи та результат). Наприклад, якщо допоміжний алгоритм має заголовок виду алг Min2 (цел K, N, целтаб B[1:K,1:N], цел min, v, w), то у основному алгоритмі має бути окрема вказівка, подібна до такої: Min2 (KA, NA, A, b1, x2, y2).

Якщо *допоміжний алгоритм є функцією користувача*, то у основному алгоритмі при посиланні на функцію у дужках записуються лише вхідні

дані (аргументи). При цьому у допоміжному алгоритмі використовується змінна знач, якій обов'язково присвоюється результат роботи і перед іменем заголовку такого алгоритму вказується тип вихідних даних – результатів функції. Наприклад, якщо заголовок допоміжного алгоритму алг вещ длина (вещ x, y, x0, y0), то у основному алгоритмі звернення до функції длина має бути подібне до такого: y := длина (x2, y2, x1, y1).

Basic.

Користувач має можливість використовувати не тільки стандартні функції мови Basic, а й нові. Перед використанням функції користувача її спочатку описують за допомогою команди DEF:

DEF FN<літера>(аргументи) = <вираз>

де <літера> – будь-яка літера латинського алфавіту. Ім'я функції користувача складається таким чином з трьох літер, причому перші дві фіксовані – FN. Після імені в круглих дужках можуть бути вказані вхідні дані – аргументи, які відокремлюються комами; після знаку = записується арифметичний вираз, яких у вигляді формули подає алгоритм обчислення значення нової функції. Наприклад, опишемо нову функцію $f = \sqrt{(x-x_0)^2+(y-y_0)^2}$:

```
20 DEF FNF ( X, Y, X0, Y0 ) = SQR ( ( X - X0 )^2 + ( Y - Y0 )^2 )
```

При зверненні до функції користувача замість формальних аргументів підставляються їх фактичні значення. Наприклад,

```
30 X1=2, Y1=2, X2=1, Y2=0
40 A = FNF ( X1, Y1, X2, Y2 )
```

Використання функції користувача доцільно при багаторазовому обчисленні за певною формулою і лише тоді, коли обчислювальний алгоритм достатньо простий та зводиться до однієї формули.

У тих випадках, коли для описання допоміжного алгоритму необхідно декілька операторів, використовують підпрограми. *Підпрограма* – це послідовність операторів, що виконуються з основної програми за оператором GOSUB, і яка закінчується оператором RETURN.

При використанні підпрограм у основній програмі необхідно:

- спочатку підготувати аргументи для передачі у підпрограму. Наприклад:

```
100 KK=K : NN=N
```

- викликати підпрограму - записати оператор GOSUB N (де N – номер рядка програми, починаючи з якого записана підпрограма). Наприклад:

```
110 GOSUB 250
```

- забезпечити повернення до основної програми результатів роботи підпрограми. Наприклад:

```
120 C1=MIN : X1=X : Y1=Y .
```

При цьому основна програма має закінчуватися оператором END.

Програма може мати декілька підпрограм і до окремої підпрограми можна звертатися з будь-якого місця програми неодноразово.

Pascal.

Функція, що визначається користувачем, складається з заголовка і тіла функції. Структура функції має вигляд:

```
FUNCTION <ім'я> (параметри) : <тип результату> ;
  Const ... ;
  Type  ... ;
  Var   ... ;
  BEGIN
    <оператори>
  END ;
```

Заголовок містить: зарезервоване слово FUNCTION; ім'я функції, задане користувачем; параметри - список змінних (кількість змінних необмежена і вказується користувачем), для кожної з яких вказується тип; тип результату, тобто тип значення, яке обчислює функція (такий тип має бути скалярним). Зауважимо, що типи заголовку функції можна позначати тільки іменами, тому, наприклад, тип масиву потребує попереднього опису в розділі Type основної програми.

Тіло функції є локальним блоком, за структурою аналогічним програмі, але закінчується не крапкою, а крапкою з комою. У тілі функції обов'язково має бути оператор присвоєння, у лівій частині якого записане ім'я функції. Тоді ім'я функції виступає у ролі змінної, якій присвоюється деяке значення, і таке значення є результатом роботи функції. При цьому у точку виклику функції повертається результат останнього присвоювання. Звернення до функції в основній програмі здійснюється за іменем функції з переліком списку аргументів, який є необов'язковим. Якщо аргументи вказані, то порядок запису і типи аргументів повинні бути такими ж, як у параметрів заголовку функції.

Після описання функції її можна використовувати у виразах разом зі стандартними функціями. Аргументами при звертанні можуть бути будь-які вирази.

Процедура користувача - це самостійна програмна одиниця, що виконується по команді з іншої програмної одиниці (програми, процедури або функції). Процедура використовується тоді, коли підзадача не схожа на функцію, тобто не повертає жодного або повертає багато значень. Процедура відрізняється від функції тільки заголовком і способом звертання до неї.

Схема заголовка процедури така:

```
PROCEDURE <ім'я> (параметри) ;
```

Зауважимо, що описання міток, констант, типів тощо дійсні тільки у межах даної процедури. В тілі процедури можна використовувати будь-які глобальні константи та змінні. Процедура може не тільки одержувати значення від програми, але і повертати у програму нові значення. Для цього існують параметри-змінні.

У той час, як параметр одержує значення аргументу шляхом присвоєння, параметр-змінна - це просто інше ім'я для аргументу. Під цим ім'ям значення аргументу стає доступним у процедурі. Всі перетворення, виконувані в процедурі над параметром-змінною, виконуються над аргументом.

У списку параметрів заголовка процедури перед параметрами-змінними ставлять слово VAR. Наприклад,

```
procedure dataOUT ( Var A, B, C : integer ) ;
```

Параметри-змінні можна використовувати не тільки в процедурах, але й у функціях.

У деяких питаннях розходження між процедурами і функціями не є суттєвим. Обговорюючи такі питання, будемо називати їх загальним іменем - блок.

У програмі може бути описано скільки завгодно блоків. Усередині цих блоків можуть бути описи інших блоків і т.д. без видимих обмежень. Важливо знати, звідки які блоки можуть бути викликані (тобто встановити видимість). Для відповіді на це питання уявіть, що блок - це будинок, у якого в вікнах одностороннє дзеркало. Зсередини через них видно усе, що знаходиться зовні, всередину ж заглянути не можна.

Видимість - це властивість всіх описів, а не тільки блоків. Знаходячись у блоці, можна користуватися всім, що видно з нього.

Об'єкти, описані в блоці, називаються внутрішніми, а ті, що описані в блоках, що охоплюють даний, - зовнішніми відносно блока. Зовнішні змінні дають додатковий канал зв'язку між блоками (головним каналом варто вважати параметри і параметри-змінні). Користуватися цим каналом треба обережно, тому що він посилює залежність блоків один від одного і цим ускладнює розробку програми.

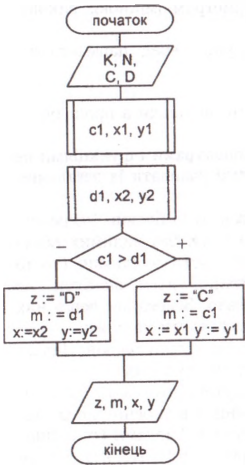
Може статися, що імена різних зовнішніх змінних співпадуть. Яка з однойменних змінних буде видна з блока? Буде видна та, що описана в найближчому з зовнішніх блоків. Якщо ж конкуренція виникне між зовнішньою і внутрішньою змінними, то видима буде внутрішня.

ПРАКТИЧНА РОБОТА № 6

ПРИКЛАД 12. Дано два цілочисельних масиви C (3x2) та D (3x2). Знайти більший серед мінімальних елементів даних масивів, індекси такого елемента та ім'я відповідного масиву.

І етап. Спочатку у кожному масиві визначимо мінімальний елемент та його індекси: для цього двічі скористаємося допоміжним алгоритмом, що забезпечує пошук мінімального елемента масиву. Знайдені мінімальні елементи порівняємо та визначимо більший.

II етап. Блок-схема (мал. 20).
III етап. Запишемо програму.



мал. 20

```
алг БЗМ (целтаб C[1:3,1:2], D[1:3,1:2], цел x,
         y, m, K, N, лит z)
  арг K, N, C, D = 3, 2,  0,-5,1,4,2,1,   0,-2,-3,4,3,1
  рез z, m, x, y
нач  цел x1, x2, y1, y2, c1, d1
     Min2 (K, N, C, c1, x1, y1)
     Min2 (K, N, D, d1, x2, y2)
     если c1>d1
       то z :="C"
          m := c1
          x := x1
          y := y1
       иначе  z := "D"
          m := d1
          x := x2
          y := y2
     все
  z := z
  m := m
  x := x
  y := y
кон
```

```
алг Min2(цел KK, NN, целтаб B [1:KK,1:NN], цел
min, v, w)
  арг KK, NN, B
  рез min, v, w
нач  цел i, j
     min :=B [1,1]
     v := 1
     w := 1
     нц для i от 1 до KK
        нц для j от 1 до NN
        . если min > B [ i , j ]
        .    то  min := B [ i , j ]
        .          v := I
        .          w := j
        .  все
        кц
     кц
кон
```

НАМ-програма 12

```
10  REM Більший серед менших
20  INPUT "Розмірн. масивів K, N" ; K, N
30  DIM C(K, N):DIM D(K, N):DIM B(K, N)
40  FOR i=1 TO K
50  FOR j=1 TO N
60  INPUT "Елемент масиву C" ; C(i, j)
70  B(i, j)=C(i, j)
80  NEXT j
90  NEXT i
100 KK=K : NN=N
110 GOSUB  250
120 C1=MIN : X1=X : Y1=Y
130 FOR i=1 TO K
140 FOR j=1 TO N
150 INPUT "Елемент масиву D" ;  D(i, j)
160 B(i, j)=D(i, j)
170 NEXT j
180 NEXT i
190 KK=K : NN=N
200 GOSUB 250
210 D1=MIN : X2=X : Y2=Y
220 IF C1>D1 THEN Z$="C":M=C1:X=X1:
Y=Y1 ELSE Z$="D":M=D1:X=X2 Y=Y2
230 PRINT Z$, M, X, Y
240 END
250 REM - - - - - - - Мінімальне - - - - - - -
260 MIN=B(1,1): X=1: Y=1
270 FOR i = 1 TO KK
280 FOR j = 1 TO NN
290 IF MIN > B( i, j) THEN MIN= B(i,j) :
X=i : Y=1
300 NEXT j
310 NEXT i
320 RETURN
```

Basic-програма 12

```
Program Big_small ;
  Const
    K=3 ; N=2 ;
  Var
    x1,y1,x2,y2,x,y, i, j, c1,d1,min,M: integer;
    c, d, b : array[1..K,1..N] of integer ;
    z : string ;
  Procedure Minimum ;
    Begin
      Min := B [1,1] ; X :=1 ; Y :=1 ;
      For i := 1 to KK do
        For j := 1 to NN do
          If Min > B [i, j] Then
            begin
              Min := B [i, j] ; X := i ; Y := j
            end
    End;
Begin
  For i := 1 to K do  {введення масиву C}
    For j := 1 to N do ReadLn ( C[i, j] ) ;
{пошук мін. елементу у масиві C}
  KK :=K ; NN := N ; B := C ;
  Minimum ;
  C1 := Min ; X1 :=X ; Y1 :=Y ;
For i := 1 to K do   {введення масиву D}
    For j := 1 to N do ReadLn ( D [i, j] ) ;
{пошук мін. елементу у масиві D}
  KK :=K ; NN := N ; B := D ;
  Minimum ;
  D1 := Min ; X2 := X ; Y2 := Y ;
  If C1 > D1 Then
    begin
      z := 'C' ; M := C1 ; X := X1 ; Y := Y1
    end
  Else
    begin
      z := 'D' ; M := D1 ; X := X2 ; Y := Y2
    end ;
  WriteLn ( z, ' M=', M :4, X :4, Y :4)
End.
```

Pascal-програма 12

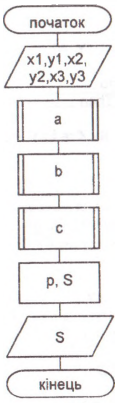
IV етап.

Тест1. K=3, N=2, масив C: $\begin{Bmatrix} 0 & 4 \\ -5 & 2 \\ 1 & 1 \end{Bmatrix}$, масив D: $\begin{Bmatrix} 0 & 4 \\ -2 & 3 \\ -3 & 1 \end{Bmatrix}$. Елемент D(3,1)= −3.

Тест2. K=3, N=2, масив C: $\begin{Bmatrix} 0 & 4 \\ -1 & 2 \\ 1 & 1 \end{Bmatrix}$, масив D: $\begin{Bmatrix} 0 & 4 \\ -2 & 3 \\ -3 & 1 \end{Bmatrix}$. Елемент C(2,1)= −1.

ПРИКЛАД 13. Обчислити площу трикутника, який задано координатами своїх вершин.

I етап. Визначимо довжину кожної із сторін трикутника: скористаємося функцією користувача, яка за координатами двох точок знаходить довжину відрізка. Площу визначимо за формулою Герона.



мал. 21

II етап. Блок-схема (мал. 21).

III етап. Запишемо текст програми.

```
алг Герон (вещ x1, y1, x2, y2, x3, y3, S)
  арг x1, y1, x2, y2, x3, y3
  рез S
нач вещ a, b, c, p
    a := длина (x2, y2, x1, y1)
    b := длина (x2, y2, x3, y3)
    c := длина (x3, y3, x1, y1)
    p := (a + b +c ) / 2
    S := (p*(p-a)*(p-b)*(p-c))**(1/2)
    S := S
кон
-----------------------------------
алг цел длина (вещ x, y, x0, y0)
  арг x, y, x0, y0
  рез
нач
    знач :=((x-x0)**2+(y-y0)**2)**(1/2)
кон
```

НАМ-програма 13

```
10 REM Герон
20 DEF FNF (X,Y,X0,Y0)=SQR((X-
X0)^2+(Y-Y0)^2)
30 INPUT "Введіть координати
точок" ; X1, Y1, X2, Y2, X3, Y3
40 A = FNF (X1,Y1,X2,Y2)
50 B = FNF (X2,Y2,X3,Y3)
60 C = FNF (X1,Y1,X3,Y3)
70 P = (A + B + C) / 2
80 S = SQR (P*(P-A)*(P-B)*(P-C))
90 PRINT "Площа дорівнює "; S
100 END
```

Basic-програма 13

```
Program Geron;
  Var
    a, b, c, p, s,   x1, y1, x2, y2, x3, y3 : real ;
  Function D (x, y, x0, y0 : real) : real ;
    begin
      d := Sqrt ( Sqr (x-x0)+Sqr (y-y0) ) ;
    end;
Begin
  Write ( ' Введіть координати точок ' ) ;
    ReadLn (x1, y1, x2, y2, x3, y3) ;
  a := d (x1, y1, x2, y2) ; b := d (x3, y3, x2, y2) ;
  c := d (x1, y1, x3, y3) ; p := ( a + b +c ) / 2 ;
  s := Sqrt ( p* (p-a) * (p-b) * (p-c) ) ;
  Writeln ( ' s= ', s :10 :4 )
End.
```

Pascal-програма 13

IV етап. Тест. Вхідні дані: X1=0 , Y1=0 , X2=3 , Y2=3 , X3=7 , Y3= -1 . Результат: S=12. ■

Лабораторна робота №6

Мета: Набути уміння та навички роботи з підпрограмами.

Матеріальне забезпечення: Е-практикум (MSX-Basic v.1.0, Pascal v.5.5)

Теоретична частина: завдання та контрольні питання.

І рівень.

1. Основний алгоритм. Допоміжний алгоритм та його види.
2. Що розуміється під терміном "підпрограма"?
3. Поняття функції користувача та процедури користувача.
4. За яких умов доцільно використовувати допоміжні алгоритми?

ІІ рівень.

1. У чому полягає різниця між допоміжними алгоритмами: функцією користувача та процедурою користувача?
2. Описати технологію подання у програмі (синтаксис) функції користувача. Навести приклад.
3. Описати синтаксис процедури користувача. Навести приклад.

ІІІ рівень.

1. Поняття рекурсії. Рекурсивно означити факторіал числа.
2. Поняття рекурсивної процедури. Навести приклад.
3. Поняття рекурсивної функції. Навести приклад.
4. Описати алгоритм Хоора.

Практична частина.

І рівень.

Модифікувати програму №13, забезпечивши знаходження повної поверхні трикутної піраміди, якщо відомо координати її вершин.

ІІ рівень.

Модифікувати програму першого рівня, залучивши підпрограму для знаходження цілої частини попереднього результату (***N=(S)***) та виконати наступні дії відповідно до варіанту.

1-3 варіанти. Перевірити для парного числа 2*N>2 гіпотезу Гольдбаха: будь-яке парне число, більше двох, можна подати у вигляді суми двох простих чисел.

4-6 варіанти. З'ясувати, чи є у послідовності N, N+1, N+2,..., 2N числа-близнята, тобто прості числа, різниця яких дорівнює числу 2.

7-9 варіанти. Розкласти на прості множники число N!.

10-12 варіанти. Вивести на екран N перших простих чисел.

13-15 варіанти. Вивести на екран N перших чисел ряду Фібоначі.

ІІІ рівень.

1 варіант. Обчислити кількість днів між двома датами. При необхідності можна ввести будь-який рік, що був високосним.

2 варіант. Вивести число N на екран прописом.

3 варіант. Дано набір назв міст, використовуючи правила гри в міста, утворити як можна більший ланцюг із даних слів.

4 варіант. Розмістити на шаховій дошці 8 ферзів так, щоб вони не загрожували один одному. Знайти всі можливі розміщення.

5 варіант. Написати програму, що реалізує алгоритм видачі здачі мінімальною кількістю монет в старій англійській грошовій системі (12 пенсів = 1 шилінг, 20 шилінгів = 1 фунт, 21 фунт = 1 гінея).

6 варіант. Магічним квадратом називається такий числовий квадрат, заповнений цілими числами від 1 до N^2, у якого суми чисел розташованих на довільній горизонталі, вертикалі або діагоналі рівні. Знайдіть магічний квадрат розмірність якого N.

7 варіант. Задано правила, за якими великі літери в словах замінюються на інші слова: A→aBb, A→abc, B→oh, B→zA. Знайдіть кількість слів, утворених з деякого початкового слова за даними правилами і що складені лише з малих літер, а також довжина цих слів повинна бути вдвічі більша від довжини початкового слова.

8 варіант. Є N предметів з відомою вагою і вартістю. Знайти такий набір предметів, щоб їх сумарна вага не перевищувала вантажність автомобіля M, а вартість, була найбільшої.

9 варіант. На площині дано координати M точок. Використовуючи ці точки, як вершини побудувати опуклий многокутник найбільший за площею.

10 варіант. Дано масив A (n x m), що складається з нулів та одиниць. Знайти найбільший за площею прямокутник, що складається з одних одиниць.

11 варіант. Дано координати N точок, що є кінцями відрізків деякої ламаної. Перевірити, чи має ламана точки самоперетину.

12 варіант. Є N населених пунктів і відома вартість проїзду між ними, якщо між ними є дорога, у противному випадку у таблиці стоїть 0. Знайти найдешевший замкнутий маршруг, що проходить через всі населені пункти.

13 варіант. Утворити новий рядок, що складається лише із слів-"паліндромів" даного рядка.

14 варіант. З певної клітинки на шаховій дошці конем необхідно перейти у задану клітинку за мінімальну кількість кроків (координати початкової та кінцевої клітинок задаються).

15 варіант. На нараду зібрались аташе з M країн, кожен з них вільно володіє двома мовами. Необхідно розсадити їх за круглим столом так, щоб вони могли вільно спілкуватися з сусідом.

Вимоги до захисту лабораторної роботи.

Протокол повинен містити процес розробки математичної моделі для 2-3 рівня, блок-схеми алгоритмів, тексти програм, тести: вхідні дані та результати роботи програми.

ДОДАТКОВІ МОЖЛИВОСТІ TURBO-PASCAL

1. СЕРЕДОВИЩЕ ПРОГРАМУВАННЯ TURBO PASCAL 5.5

Середовище програмування завантажується виконанням файлу turbo.exe. Активізація головного меню - F10. Головне меню має секції:

File	– команди для роботи з файлами;
Edit	– редагування тексту програми;
Run	– виконання програми;
Compile	– компіляція програми;
Options	– параметри конфігурування середовища;
Debug	– команди налагодження програми;
Break/watch	– команди переривання та перегляду.

Розглянемо особливості роботи з Turbo Pascal 5.5.

Завантаження середовища програмування Turbo Pascal:

- якщо ЕОМ працює під управлінням ОС Windows, то перед завантаженням середовища програмування бажано перейти до режиму DOS (Пуск→Завершение работы...→Перезагрузить комп'ютер в режиме MS-DOS→Ok);
- завантажити драйвер, що забезпечує набір символів кирилиці та відображає їх на екрані (наприклад, виконати файл cyr_vga.com);
- відкрити каталог, що містить файл turbo.exe, виконати файл.

Створення нового файлу (нової програми):

- у секції "File" головного меню виконати команду "New";
- новостворюваному файлу за замовчуванням надається ім'я noname.pas. Рекомендується відразу перейменувати такий файл, зберігши його під новим іменем. Для цього у секції "File" виконати команду "Write to" та у діалогове вікно "New name" за правилами DOS ввести шлях нового розташування файлу та його нове ім'я (з розширенням pas).

Відкриття існуючого файлу (тексту існуючої програми):

- у секції "File" виконати команду "Load" (або натиснути клавішу F3);
- у діалоговому вікні "Load file name" пересвідчитись у правильності запропонованого шаблону для відбору файлів – текстів програм мовою Pascal (має бути *.pas) або задати такий шаблон;
- використовуючи клавіши управління курсором відшукати каталог та потрібний файл, перемістити на нього курсор, натиснути клавішу Enter.

Збереження під новим іменем файлу з текстом програми:

- у секції "File" головного меню виконати команду "Write to";
- у діалогове вікно "New name" за правилами DOS ввести шлях нового розташування файлу та його нове ім'я (з розширенням pas).

Набір та редагування тексту програми.

Команда "Edit" головного меню викликає вбудований у Turbo Pascal екранний редактор та забезпечує перехід у режим редагування (набору) тексту програми. Знаходячись у вказаному режимі, можна викликати русифіковану допомогу (F1) з переліком команд редактора.

Компіляція тексту програми.

- перед виконанням компіляції необхідно пересвідчитись у потрібному виборі місця для запам'ятовування виконуваного коду програми (секція "Compile" головного меню, пункт Destination). Рекомендується спочатку встановити Destination Memory. Якщо встановлено Destination Disk, то під час компіляції буде створено і записано на диск виконуваний (.exe) файл у той же каталог, де міститься файл з текстом мовою Pascal. Переустановлення з Disk на Memory та навпаки виконується вибором пункту Destination та натисканням клавіші Enter;
- у секції "Compile" головного меню виконати команду "Compile" або натиснути клавішу F9. При цьому запускається транслятор, який перекладає текст програми з мови Паскаль у машинні коди, перевіряючи програму на наявність синтаксичних помилок. Про успішність компіляції свідчить повідомлення "Compilation successful";
- якщо знайдено синтаксичну помилку, компіляція переривається, у вікні редагування курсор встановлюється на рядок з помилкою та з'являється повідомлення про тип помилки. Якщо, маючи повідомлення, натиснути клавішу F1, то можна одержати розширену інформацію про помилку;
- після успішної компіляції рекомендується зберегти її текст (секція "File", команда "Save" або натиснути клавішу F2).

Виконання та перегляд результатів роботи програми:

- у секції "Run" виконати команду "Run" або натиснути Ctrl+F9.
- якщо програмою передбачено, ввести дані
- у разі потреби переривання роботи програми користувачем – Ctrl+Break;
- результати роботи можна проглянути, натиснувши Alt+F5.

Створення виконуваного файлу (розширення .exe):

- у секції "Compile" надати Destination значення Disk ;
- у цій же секції виконати команду "Make" або натиснути F9.

Завершення роботи у середовищі:

- зберегти зміни тексту програми, яка міститься у вікні редагування.
- виконати комбінацію клавіш Alt+X. Якщо до тексту програми вносилися зміни, які не були збережені, то під час виходу із середовища з'являється повідомлення "<ім'я файлу> not saved. Save? (Y/N)" (файл не збережено. Зберегти? (Так/Ні)). Далі потрібно натиснути одну з клавіш Y (зберегти зміни) або N (не зберігати зміни);
- повернутися в з режиму MS-DOS у Windows (набрати у командному рядку DOS команду Exit та натиснути Enter);
- коректно завершити роботу ОС Windows.

2.1. Поняття запису

Часто зустрічається така інформація, яка складається із даних різного типу. Це анкети, таблиці, картки каталогу тощо. Для подання їх у програмі використовують записи.

Запис – це складений тип даних, який об'єднує у собі різнотипні елементи (поля запису). Цей тип описує конструкція:

```
ім'я_типу =RECORD
    ім'я_поля: тип поля;
    ім'я_поля: тип поля;
    ...
    ім'я_поля: тип поля
End;
```

Наприклад, опишемо запис, що містить відомості про дату народження.

```
Type
  Day=1..31;
  Month=1..12;
     Date=Record
       Chislo : Day;
       Mesayc : Month,
       God : integer
     End;
```

Над записами допустимі операції присвоєння, перевірки на рівність і нерівність, вводу і виводу. Поле запису у програмі можна використовувати як змінну того ж типу, що і поле.

Звертаються до поля по складеному імені.

Наприклад, якщо змінна Born має тип - запис Date, то звернутися до поля Chislo можна так:

```
Born.Chislo
```

Тип поля може будь-яким, у тому числі і записом.

2.2. Оператор WITH

Цей оператор дозволяє скоротити складене ім'я поля під час звернення до полів запису. Форма:

```
WITH ім'я_запису DO оператор;
```

Скрізь у середині оператора можна не вказувати ім'я запису у складеному імені поля, воно буде додано автоматично. Наприклад, для описаного запису типу Date оператор With можна використати так:

```
Var
  Born : Date ;
  ...
  With Born do
    Begin
    Readln ( Chislo, Mesayc, God);
    Write (Chislo, Mesayc, God)
  End;
```

ПРИКЛАД 14. Записати відомості про N студентів. Відомості повинні мати структуру, що відображена у таблиці 1.

Таблиця 1. Відомості про студентів.

№ п/п	Прізвище	Ім'я	Дата народження			Стать	Оцінка диплому		
			Число	Місяць	Рік		Матем-ка	Педаг-ка	Фізика
1									
2									
...									
N									

Запишемо текст програми:

```
Program Spisok_Stud;
  Const
    N=5;
  Type
    Day=1..31;
    Bal=2..5;
    Month=1..12;
    Era=1900..2002;
      Ocen=Record
             Mat    : Bal;
             Ped    : Bal;
             Phiz   : Bal
        End;
      Date=Record
             Chislo : Day;
             Misac  : Month;
             Ric    : Era
        End;
      Student=Record
             Fam    : String[15];
             Name   : String[15];
             Born   : Date;
             Pol    : Char;
             Dipl   : Ocen
        End;
```

```
  Var
    I    : Integer;
    Body : Array [1..N] of Student;
BEGIN
    For I:=1 to N do
      Begin
        With Body[I] do
              Begin
                  WritreLn ('Номер п/п - ', I)
                  Write('Прізвище - '); ReadLn(Fam);
                  Write('Ім''я - '); ReadLn(Name);
                  Write('Число народж. - '); ReadLn(Born.Chislo);
                  Write('Місяц народж.(1-12) - '); ReadLn(Born.Misac);
                  Write('Рік народж. - '); ReadLn(Born.Ric);
                  Write('Стать (ч/ж) - '); ReadLn(Pol);
                      With Dipl do
                        Begin
                            Write('Математика - '); ReadLn(Mat);
                            Write('Педагогіка - '); ReadLn(Ped);
                            Write('Фізика - '); ReadLn(Phiz)
                        End
              End;
        WriteLn ('**************************************')
      End
END.
```

Pascal-програма 14

2.3. Послідовний пошук.

У записах можуть зберігатися великі обсяги інформації і часто виникає потреба відшукати певні відомості. Знайти відомості означає визначити номер запису у їх масиві або повідомити про відсутність такого запису.

Найпростіший алгоритм пошуку – послідовний. Суть його полягає у тому, що у для всіх записів послідовно, починаючи з першого, перевіряється значення ознаки, за якою ведеться пошук.

Наприклад, нехай для розглянутої задачі прикладу 14 необхідно відшукати студента, що має прізвище Хоменко. Це можна зробити, дописавши перед кінцем програми наступне:

```
i:=1;
While (i <= n) and (Body[i].Fam<>'Хоменко') do
  i:=i+1;
```

Очевидно, що якщо i<=n, то потрібного запису не має, інакше буде знайдено перший запис зі вказаною ознакою.

Якщо ж потрібно вивести ім'я особи з таким прізвищем, то можна дописати таке:

```
If i<=n  Then WriteLn ('Ім"я - ', Body[i].Name)
         Else WriteLn ('Запис відсутній') ;
```

2.4. Двійковий пошук.

Загальною властивістю алгоритмів послідовного пошуку є те, що час пошуку прямо пропорційний кількості записів. Але якщо масив записів відсортований за значенням шуканої ознаки, то можна шукати швидше.

Розглянемо алгоритм пошуку студента з певним прізвищем, для того випадку, коли всі записи у списку студентів відсортовані за прізвищами в алфавітному порядку:

1. Відкрити список посередині.
2. Порівняти шукане прізвище з тим, що у середині списку.
3. Якщо прізвища співпадають, пошук завершено.
4. Якщо прізвище у середині більше шуканого, продовжити пошук у першій половині списку. Інакше (прізвище у середині менше шуканого) продовжити пошук у другій половині списку.

"Продовжити пошук" означає застосування алгоритму в цілому до половини списку, тобто відкрити його посередині і т.д. У результаті або прізвище буде знайдено, або ділити навпіл буде нічого, що означає, що прізвища у списку немає.

Такий список можна зберігати у масиві записів, початок і кінець області пошуку запам'ятовувати в окремих змінних.

Порівняємо швидкість послідовного і двійкового пошуку. Перший скорочує область пошуку на один запис за кожен крок алгоритму. Другий за один крок зменшує область пошуку удвічі. Тому при подвоєнні числа записів час послідовного пошуку збільшується у два рази, а час двійкового пошуку збільшується лише на один крок алгоритму.

Лабораторна робота №7

Мета: Набутти вміння та навички роботи з записами.

Матеріальне забезпечення: Turbo-Pascal v.5.5-7.0.

Теоретична частина: завдання та контрольні питання.

1. Дати означення запису. Які операції допустимі над записами?
2. Як звернутися до поля запису? Навести приклад. Роль оператора With у зверненні до полів запису.
3. Чим відрізняються масиви від записів?

4. Як, використовуючи оператор With, модифікувати програму прикладу 14 так, щоб спросити складені імена при зверненні до полів запису Born?

5. Які переваги двійкового пошуку над послідовним?

Практична частина.

Скласти список навчальної групи, що налічує 15 осіб. Для кожного студента вказати прізвище, ім'я, стать, дату народження, курс, групу та оцінки з таких предметів: математичний аналіз, геометрія, алгебра, інформатика, педагогіка, психологія, фізика та історія. Інформацію про кожного студента оформити у вигляді запису. Сукупність записів об'єднати у масив. Забезпечити виведення на екран:

1 варіант. Прізвищ та дат народження студентів, які цього року вже відсвяткували свій день народження.

2 варіант. Прізвищ студентів, що навчаються на 4 і 5.

3 варіант. Прізвищ студентів, що мають одну оцінку 3.

4 варіант. Імен студентів, що отримали за весь час навчання не більше ніж три оцінки 4, а всі інші оцінки - 5.

5 варіант. Списку студентів, прізвища яких розпочинаються з літери А. Дані у списку: прізвище, ім'я студента, оцінки за весь час навчання.

6 варіант. Списку студентів, прізвища яких розпочинаються з літери Б. Дані у списку: прізвище, ім'я студента, дата народження.

7 варіант. Прізвищ і дат народження студентів, що не отримали жодної оцінки 3 за весь час навчання.

8 варіант. Прізвищ всіх студентів та середню оцінку за навчання.

9 варіант. Впорядкованого за віком списку прізвищ студентів.

10 варіант. Списку прізвищ студентів, згрупованого за курсами навчання.

11 варіант. Списку студентів, впорядкованого у такий спосіб: спочатку прізвища, імена студентів жіночої статі, а потім - чоловічої.

12 варіант. Списку прізвищ студентів, відсортований у зворотному порядку за алфавітом.

13 варіант. Списку студентів, що мають однакові прізвища.

14 варіант. Даних про студентів певного курсу (прізвище та ім'я), у яких середній бал вище 4.5.

15 варіант. Прізвищ та ініціалів студентів, що підлягають відрахуванню (мають хоча б одну двійку).

Вимоги до захисту лабораторної роботи.

Знати відповіді на питання та виконати завдання теоретичної частини. Протокол лабораторної роботи повинен містити назву теми та текст програми.

3. РОБОТА З ФАЙЛАМИ

Розрізняють два види файлів: послідовного і довільного доступу. Послідовні файли складаються з елементів різної довжини, між якими стоять розділювачі. Щоб знайти елемент послідовного файлу, потрібно переглянути все, що йому передує.

Файли довільного доступу складаються з однотипних елементів, як масиви. Знайти довільний елемент можна за порядковим номером.

Послідовні файли називають текстовими, а файли довільно доступу – типізованими. Програма може обробляти існуючий файл або створювати новий файл, але робота проходить такими етапами: відкриття файлу, зчитування або запис, закриття файлу.

3.1. Текстові файли

Текстові файли зберігають інформацію у вигляді послідовності символів. Символи утворюють рядки довільної довжини. В кінці кожного рядка знаходяться два особливих символи: #13 #10, які відокремлюють рядок від наступного.

У програмі на Паскалі текстовий файл подає файлова змінна типу TEXT. Наприклад, опишемо F як таку файлову змінну:

```
Var      F  : Text;
```

Перед тим, як із текстового файлу зчитувати (записувати у файл) інформацію, необхідно певним чином підготувати файл до такого використання. Для цього є стандартні оператори:

Assign (файлова_змінна, ім'я_файлу) – зв'язує файлову змінну з певним іменем зовнішнього текстового файлу;

Reset (файлова_змінна) – відкриває тестовий файл для читання;

ReWrite (файлова_змінна) – відкриває текстовий файл для запису;

Append (файлова_змінна) – відкриває текстовий файл для доповнення.

У файл , відкритий для запису чи доповнення, можна тільки записувати, із файлу, відкритого для читання, можна тільки читати.

Якщо необхідно прочитати із файлу, відкритого для запису, його спочатку потрібно закрити оператором **Close (файлова_змінна)**, а потім знову відкрити вже для читання. Цим оператором до завершення роботи програми рекомендується закрити всі файли, відкриті під час роботи програми.

Читання з текстового файлу

Для того, щоб прочитати із файлу використовують оператор **Read**. Якщо перед списком вводу у цьому операторі вказати ім'я файлової змінної, то оператор буде вводити дані не з клавіатури, а із файлу.

Наприклад,

```
Var
  F : Text ;
  A, b,c : integer;
Begin
  Assign (F, 'dani.dat');
  Reset (F);
  ...
  Read (A, B, C);          {введення з клавіатури}
  Read (F, A, B, C);       {введення з файлу "dani.dat"}
```

Текстовий файл для оператора Read таке ж джерело символів, як і клавіатура, але символи слідують один за одним не у часі, а у просторі файлу. Щоб зрозуміти, що прочитає із файлу оператор Read, існує поняття покажчика файлу. Фактично покажчик – це номер чергового символа файлу, але краще уявити його як стрілку, напрямлену у певну точку файлу. Одразу після відкриття стрілка вказує на перший символ файлу. Читання чергової порції даних завжди виконується, починаючи з символу, на який вказує стрілка. Після читання стрілка автоматично переміщується вперед на довжину прочитаної ділянки. Так продовжується доти, доки стрілка не досягне кінця файлу. Подальші спроби читання викличуть повідомлення про помилку.

Логічна функція Eof (файлова_змінна) дозволяє визначити, чи можна ще читати із файлу. Вона повертає True, якщо досягнуто кінець файлу, і False в іншому випадку.

Додатковий оператор читання ReadLn вводить все, що передбачено у його переліку вводу, і пересуває покажчик на початок наступного рядка.

ПРИКЛАД 15. Прочитати інформацію, що міститься у рядку текстового файлу "dani.dat", та вивести її на екран.

Текст програми:

```
Program Chtenie;
  Var
    F : Text ;
    s : string ;
Begin
  Assign (F, 'dani.dat');
  Reset (F);
    While not eof (F) do begin
      ReadLn (F,S);    {зчитує весь рядок із файлу "dani.dat" }
      WriteLn (S)
    end;
  Close (f)
End.
```

Pascal-програма 15

Запис у текстовий файл

Запис у тестовий файл виконується оператором Write, у якому перед списком вводу стоїть файлова змінна. Інформація, що виводиться приєднується до тієї, що була виведена у файл з моменту його відкриття.

Для створення нового файлу його відкривають оператором ReWrite. Якщо ж доповнюють існуючий файл, його відкриття виконують оператором Append. Відкриття існуючого файлу оператором ReWrite призводить до втрати інформації, яка у файлі містилася раніше.

Завершивши запис у файл, його необхідно закрити. Тільки після закриття новий файл буде остаточно сформовано.

Додатковий оператор запису WriteLn виводить все, що передбачено списком виводу та символи #13 #10 надодаток.

ПРИКЛАД 16. У доповнення до задачі прикладу 14 відомості про студентів, що вводяться з клавіатури, потрібно записати у новий файл Anketa.txt. Після чого у інший новий файл Gold_dip.txt записати прізвища тих студентів, що мають відмінні оцінки з усіх предметів.

```
Program Spisok_Stud_File;
  Const
    N=3;
  Type
    Day=1..31;
    Bal=2..5;
    Month=1..12;
    Era=1900..2001;
      Ocen=Record
                Mat : Bal;
                Ped : Bal;
                Phiz: Bal
        End;
      Date=Record
                Chislo  : Day;
                Misac   : Month;
                Ric     : Era
        End;
      Student=Record
                Fam     : String[15];
                Name    : String[15];
                Born    : Date;
                Pol     : Char;
                Dipl    : Ocen
        End;
```

```
  Var
    I    : Integer;
    F    : Text;
    Body : Array [1..N] of Student;
BEGIN
  Assign (F, 'Anketa.txt'); Rewrite(F);
  For I:=1 to N do
      With Body[I] do
        Begin
                WriteLn(F,I);
                Write('Прізвище - '); ReadLn(Fam);
                WriteLn(F,' Прізвище - ',Fam);
                Write('Ім"я - '); ReadLn(Name); WriteLn(F,' Ім"я - ',Name);
                Write('Число народж. - '); ReadLn(Born.Chislo);
                Write('Місяц народж.(1-12) - '); ReadLn(Born.Misac);
                Write('Рік народж. - '); ReadLn(Born.Ric);
                WriteLn(F,' Дата нар. - ',Born.Chislo, ,Born.Misac, ,Born.Ric);
                Write('Стать (ч/ж) - '); ReadLn(Pol); WriteLn(F,' Стать - ', Pol);
                    With Dipl do
                        Begin
                                Write('Математика - '); ReadLn(Mat);
                                Write(F,' Математика - ',Mat,' ');
                                Write('Педагогіка - '); ReadLn(Ped);
                                Write(F,' Педагогіка - ',Ped,' ');
                                Write('Фізика - '); ReadLn(Phiz);
                                WriteLn(F,' Фізика - ',Phiz);
                        End
      End;
  Close(F);
  Assign (F,'Gold_dip.txt'); ReWrite(F);
  For I:=1 to N Do
      With Body[I].Dipl do
        If (Mat=5) and (Ped=5) and (Phiz=5) Then WriteLn (F,I,Body[I].Fam);
  Close (F)
END.
```

Pascal-програма 16

3.2. Типізовані файли

Нагадаємо, що типізованими файлами є файли довільного доступу. Всі елементи такого файлу одного типу, а значить і розміру. Завдяки цьому можливий довільний доступ до елементів файлу — за порядковим номером елемента однозначно визначають його розташування. Нумеруються елементи файлу цілими числами, починаючи з нуля.

Типізована файлова змінна описується так:

VAR ім'я FILE OF базовий тип.

Зв'язування файлової змінної із зовнішнім файлом виконується так само, як і для текстових файлів, – оператором Assign.

У типізованому файлі дозволяється чергування операцій запису та зчитування. Оператором Reset відкривається існуючий файл, а оператором Rewrite — створюється новий. Закривається файл оператором Close. Оператори Read і Write використовуються аналогічно описаному вище для текстових файлів.

Довільний доступ до елементів файлу виконується оператором

SEEK (файлова змінна, номер елемента)

Для визначення розташування покажчика у файлі використовують функцію FilePos (файлова змінна), а для визначення кількості записів FileSize (файлова змінна). Робота з типізованим файлом – див. приклад [2;61-62].

Лабораторна робота №8

Мета: Набути вміння та навички роботі з даними у файлах.

Матеріальне забезпечення: Turbo Pascal v.5.5 – 7.0.

Теоретична частина: завдання та контрольні питання.

1. Що означає термін "текстовий файл"?
2. Які файли називаються типізованими? Чим вони відрізняються від текстових?
3. Які оператори і у якому порядку використовуються для зчитування інформації з текстового файлу?
4. Які оператори і у якому порядку використовуються для запису інформації у текстовий файл?
5. Які оператори забезпечують роботу з типізованим файлом?
6. Описати роботу фрагмента програми:

```
While Not Eof (F) Do
  Begin
   ReadLn (F, S);
   WriteLn (S)
  End;
```

Практична частина

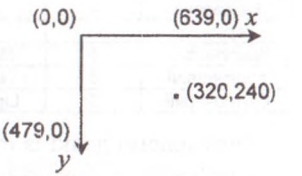
Модифікувати програми лабораторних робіт №4 та №5, забезпечивши введення даних з файлу DateN.txt і запис результату роботи програми в файл SolN.txt, де N – тризначне число, перша цифра якого 4 або 5 (відповідно до лабораторної), а дві наступні – номер варіанту.

Вимоги до захисту лабораторної роботи

Знати відповіді на питання та виконати завдання теоретичної частини. Протокол лабораторної роботи повинен містити назву теми, дані з файлів DateN.txt та SolN.txt.

Графічний режим

Графічний екран дисплея складається із точок, які можна відображати певним кольором. Ці точки мають назву піксель. Кількість пікселів на екрані може бути різноманітним і залежить він від якості монітора. Ми будемо розглядати екран, який має 640 точок по горизонталі та 480 по вертикалі. Початок відліку розташовано у верхньому лівому кутку екрана (мал. 22). Кожна точка характеризується двома координатами (x, y).



мал. 22

Ініціалізація графічного режиму

У розділі Uses програми, яка має виводити графічну інформацію, для підключення відповідного модуля потрібно записати:

```
Uses Graph;
```

Щоб забезпечити роботу з модулем Graph, програма має виконати процедуру InitGraph. Тому у тексті програми слід зробити таке:

- описати змінні цілого типу Driver і Mode;
- у розділі операторів присвоїти значення змінній Driver:=Detect;
- записати процедуру InitGraph (Driver, Mode, <шлях до драйвера – файлу egavga.bgi>);
- передбачити завершення роботи програми при неможливості ініціалізації графічного режиму , використавши оператор Graphresult;
- використати оператори створення та обробки графічних об'єктів;
- для візуалізації та затримки графічного зображення на екрані можна використати оператор ReadLn;
- закрити графічний режим – використати оператор CloseGraph.

Програма, що створює графічні зображення, має структуру:

```
Uses Graph;
Var  Driver, Mode: Integer;
Begin
  Driver:=Detect;   InitGraph (Driver, Mode, '');
  If graphresult<> 0 then halt;   {графічний режим не вдалося ініціалізувати}
    <оператори>;
  ReadLn;
  CloseGraph
End.
```

Створення графічних образів

Перед початком роботи необхідно встановити колір фону (екрану) та колір малюнка. Ці операції виконують процедури:

SetBkColor (C) – колір фону;

SetColor (C) – колір малюнка.

Параметр C може задаватися задається цілим числом (0..15) або назвою. Залежність між числом та назвою можна побачити в таблиці:

Колір	Число	Назва	Колір	Число	Назва
Чорний	0	Black	Темно-сірий	8	DarkGray
Синій	1	Blue	Світло-блакитний	9	LightBlue
Зелений	2	Green	Світло-зелений	10	LightGreen
Бірюзовий	3	Cyan	Світло-бірюзовий	11	LightCyan
Червоний	4	Red	Світло-червоний	12	LightRed
Малиновий	5	Magenta	Світло-малиновий	13	LightMagenta
Коричневий	6	Brown	Жовтий	14	Yellow
Світло-сірий	7	LightGray	Білий	15	White

Розглянемо деякі із стандартних графічних процедур та функцій:

Graphresult – повертає код помилки, якщо неможливо ініціалізувати графічний режим, і 0 в разі успіху;

GetMaxX - максимальне значення абсциси;

GetMaxY - максимальне значення ординати;

PutPixel(X, Y, C) - виводить точку з координатами (X, Y) та кольором C;

Line (X1,Y1, X2, Y2) - малює відрізок з кінцями в точках з координатами (X1,Y1) та (X2,Y2);

LineTo(X, Y) - малює відрізок з кінцями в поточній точці та точці з координатами (X, Y).

SetLineStyle (A, B, C) - задає стиль лінії; де:

A – назва лінії або цифра відповідно до таблиці:

B – лише для лінії користувача, інакше – 0 ;

C – задає товщину лінії і може приймати значення NormWidth (товщина 1 піксель) або ThickWidth (3 пікселі).

Вид лінії	Цифра	Назва лінії
Суцільна лінія	0	SolidLn
Лінія з крапок	1	DottedLn
Лінія з крапок тире	2	CenterLn
Пунктирна лінія	3	DashedLn
Лінія користувача	4	UserBitLn

Rectangle (X1,Y1,X2,Y2) - малює прямокутник у якого (X1,Y1) - координати лівого верхнього кута, а (X2,Y2) - координати правого нижнього кута.

Bar (X1,Y1,X2,Y2); - малює зафарбований прямокутник у якого (X1,Y1) - координати лівого верхнього кута, а (X2,Y2) - координати правого нижнього кута.

Circle(X, Y, R) - малює коло з центром в точці (X, Y) та радіусом R.

Arc (X, Y, A, B, R) - малює дугу або сектор з координатами центру (X, Y) від кута A до кута B радіусом R.

Ellipse (X, Y, A, B, RX, RY); - малює еліпс у якого (X, Y) – координати центра, A – початковий кут, B – кінцевий кут, RX, RY – радіуси.

Floodfill(X, Y, C) – заповнює замкнену область з межею кольору C за точкою з координатами (X, Y) в середині області;

Вирази та змінні, що є координатами точок мають бути цілого типу.

ПРИКЛАД 17. Побудувати графік функції $y=cos(x)$ на проміжку [-5; 5].

Зауважимо, що побудова графіків функцій має деякі особливості:

- більшість таких графічних зображень є множиною точок;
- необхідно врахувати особливості координатної системи екрану (де знаходиться початок відліку, певний напрям осі ординат тощо);
- у залежності від величини проміжку, для якого на екрані зображується графік функції, визначається масштабуючий коефіцієнт. Він дозволяє вивести зображення у зручному вигляді. Як правило, для проміжку $[a; b]$ масштабуючий коефіцієнт буде $Kof=getmaxx/(b-a)$[1].

```
Program fun_cos;
  Uses Graph;
  Const  Kof=64;                    {коефіцієнт масштабування }
  Var
    Gd, Gm, i, a, b : Integer;
    xk, yk: real;
BEGIN
  Gd := Detect;     InitGraph(Gd, Gm, '');
  If graphresult<> 0 then halt;
    a:=getmaxx div 2;                    {абсциса центра екрана}
    b:=getmaxy div 2;                    {ордината центра екрана}
    setcolor(5);
    line(5,b,2*a-5,b);                   {лінія – вісь абсцис}
    line(a,5,a,2*b-5);                   {лінія – вісь ординат}
  for i:=5 to 2*a-5 do
    begin
      xk:=(i-a)/Kof;
      yk:=b-Kof*cos(xk);
      putpixel (i,round(yk),2)
    end;
  ReadLn;
  CloseGraph;
END.
```

Pascal-програма 17

Виведення тексту у графічному режимі

Кожен символ тексту являє собою прямокутник розміром 8 на 8 пікселів. Символ може бути точковий або штриховий. Стиль символу та спосіб розміщення тексту задає процедура

---

[1] Для визначення коефіцієнта масштабування іноді потрібно враховувати проміжок, утворений мінімальним та максимальним значенням функції: якщо такий проміжок більший за проміжок аргументів, тоді його значення беруться для визначення знаменника у пропонованій формулі.

SetTextStyle (тип шрифту, спосіб розміщення, розмір)

Тип шрифту задається назвою або цифрою згідно таблиці:

Тип шрифту	Цифра	Назва
Стандартний	0	DefaultFont
Триплекс	1	TripleFontx
Зменшений	2	SmallFont
Прямий	3	SanserifFont
Готичний	4	GothicFont

Розміщення тексту може бути горизонтальним (HorizDir або 0) або вертикальним (VertDir або 1) – знизу догори. Розмір символу задається цифрою 0, 1, 2....

Процедура OutTextXY (X, Y, 'текст') виведе на екран текст в точці з координатами (X, Y).

Процедура OutText ('текст') виведе текст у поточній точці екрана.

Процедура MoveTo (X, Y) переміщує курсор в точку з координатами (X, Y).

Лабораторна робота №9

Мета: Набути вміння та навички побудови графічних образів та графіків функцій в декартовій і полярній системах координат.

Матеріальне забезпечення: Pascal v.5.5 – 7.0.

Теоретична частина: завдання та контрольні питання.

1. Як виконується ініціалізація графічного режиму?
2. Які оператори забезпечують побудову точки, відрізка, прямокутника, кола, еліпса, дуги?
3. Яким чином виводиться текст у графічному режимі?
4. Для чого необхідно використовувати коефіцієнти масштабування під час побудови графіків функцій?

Практична частина.

1. У заданій частині графічного екрану намалювати фігуру 1, у середині фігури 1 – фігуру 2, а в середині фігури 2 – текст: номер варіанту та прізвище виконавця. Всі елементи малюнка виконати різними кольорами. Замкнені ділянки зафарбувати.

№	Частина екрану	Фігура 1	Фігура 2
1	ліва половина	коло	восьмикутник
2	права половина	квадрат	сегмент
4	нижня половина	трикутник	п'ятикутник
5	верхня половина	еліпс	прямокутник
6	центр	прямокутник	еліпс
7	лівий верхній куток	ромб	шестикутник
8	правий нижній куток	трапеція	сектор
9	правий верхній куток	паралелограм	кардиоіда
10	лівий нижній куток	шестикутник	ромб
11	центр	пентограма	коло
12	нижня половина	восьмикутник	квадрат
13	верхня половина	сегмент	трикутник
14	центр	сектор	трапеція
15	лівий верхній куток	кільце	паралелограм

2. Відобразити рух Фігури1 по периметру Фігури2.

3. Скласти програму для побудови графіка функції

№	Функції задано:		
	у декартовій системі координат	у полярній системі координат	параметрично
1	$y = e^x \sin x$	$r = \varphi$	$x = 1 - t$, $y = 1 - t^2$
2	$y = e^{-x^2} \cos 2x$	$r = \pi / \varphi$	$x = t + 1/t$, $y = t + 1/t^2$
4	$y = 1 + x - x^3$	$r = 2^{\varphi/2\pi}$	$x = ch\,t$, $y = sh\,t$
5	$y = x \sin x$	$r = 2(1 + \cos\varphi)$	$x = 5\cos^2 t$, $y = 3\sin^2 t$
6	$y = x \cos x$	$r = 10 \sin 3\varphi$	$x = 2(t - \sin t)$, $y = 2(1 - \cos t)$
7	$y = \sin^{ctg\,x} x$	$r = 6\sqrt{\cos 2\varphi}$	$x = \sqrt[t+1]{t}$, $y = \sqrt[t]{t+1}$
8	$y = ctg\dfrac{\pi x}{1 + x^2}$	$r = a \sin^3 \dfrac{\varphi}{3}$	$x = t^2$, $y = t - \dfrac{t^3}{3}$
9	$y = \dfrac{1}{1 + x^2}$	$r = a\left(\dfrac{1}{\cos\varphi} - tg\varphi\right)$	$x = (t^2 - 2)\sin t + 2t\cos t$, $y = (2 - t^2)\cos t + 2t \sin t$
10	$y = x\left(2 + \sin\dfrac{1}{x}\right)$	$r = a\cos 2\varphi$	$x = actg^2 t$, $y = a\,ctg\,t$
11	$y = \sqrt{1 + x^2}\,\sin\dfrac{\pi}{x}$	$r = a(1 - \cos\varphi)$	$x = a\sin 2t$, $y = a\sin^2 t$
12	$y = x\sqrt{\lvert x\rvert}$	$\varphi = 2\pi \sin r$	$x = a\cos^2 t$, $y = a\sin 2t$
13	$y = \dfrac{\sin x}{x}$	$r = 2a\sin\varphi$	$x = 1 + 2\sec t$, $y = -1 + tg\,t$
14	$y = e^{\frac{1}{\sin x}}$	$r = \dfrac{a}{\cos\varphi} + a\ tg\varphi$	$x = \cos t + t \sin t$, $y = \sin t - t\cos t$
15	$y = x^2 \sin^2 x$	$\varphi = 4r - r^2$	$x = 2t - t^2$, $y = 2t^2 - t^3$

Вимоги до захисту лабораторної роботи.

Знати відповіді на питання та виконати завдання теоретичної частини. Протокол лабораторної роботи повинен містити назву теми, графічні зображення згідно завдання 3.

СПИСОК ЛІТЕРАТУРИ

1. Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн М.И. Задачи по програмированию – М.: Наука. Гл. ред. физ.-мат. лит., 1988. – 224 с.
2. Бондарев В.М., Рублинецкий В.И., Качко Е.Г. Основы программирования. – Харьков: Фолио; Ростов н/Д: Феникс, 1998. – 368 с.
3. Глинский Я.Н., Анохин В.Е., Ряжская В.А. Turbo Pascal 7.0 и Delphi: Учебное пособие. – СПб: ООО «ДиаСофтЮП», 2001. – 208 с.
4. Довгаль С.И., Литвинов Б.Ю., Сбитнев А.И. Персональные ЭВМ: ТурбоПаскаль V 6.0, Объектное программирование, Локальные сети: Учебное пособие. – К.: «Информсистема сервис», 1993. –440 с.
5. Есаян А.Р. и др. Информатика: Учеб. пособие для педагогических институтов. – М.: Просвещение, 1991. – 288с.
6. Жалдак М.І., Рамський Ю.С. Інформатика: Навч. посібник / за ред. М.І. Шкіля. – К.: Вища шк., 1991. – 319 с.: іл.
7. Заварыкин В.М., Житомирский В.Г., Лапчик М.П. Основы информатики и вычислительной техники. – М.: Просвещение, 1989. – 206 с.
8. Кривонос О.М. Основи програмування мовою Паскаль. - Житомир: ЖДПУ, 2000.- 69 с., іл.
9. Основы информатики и вычислительной техники: Пробное учебное пособие для средних учеб. заведений: В 2 ч. / Под ред. А.П. Ершова, В.М. Монахова. – М.: Просвещение, 1985-1986. – Ч. 1-2.
10. Прайс Д. Программирование на языке Паскаль: Практическое руководство. – М.: Мир, 1987
11. Программирование на языке MSX-Basic / Р.И.Зарогский, А.Н.Ломакович, В.Д. Будак и др. – К.: Вища шк. Головное изд-во, 1988. – 120 с.
12. Сердюченко В.Я. Розробка алгоритмів та програмування мовою Turbo Pascal. – Харків: "Парітет", 1995. – 352 с.
13. Турбо ПАСКАЛЬ 7.0. – К.: Издательская группа BHV, 1999. – 448 с.
14. Фролов Г.Д., Кузнецов Э.И., Элементы информатики: Учеб. пособие для пед. ин-тов. – М.: Высш. шк, 1989. – 304 с.: ил.

Навчально-методичне видання

Спірін Олег Михайлович
Кривонос Олександр Миколайович

ПОЧАТКИ АЛГОРИТМІЗАЦІЇ ТА ПРОЦЕДУРНОГО ПРОГРАМУВАННЯ

Рекомендовано вченою радою Житомирського державного педагогічного університету імені Івана Франка як методичний посібник

Житомир
Поліграфічний центр Житомирського педуніверситету

Надруковано з оригінал-макета авторів.

Підписано до друку 26.02.2002. Формат 60x90/16. Ум.друк.арк.5.75.
Обл.вид.арк.6.0. Друк різографічний. Зам.40. Наклад 300.

Поліграфічний центр ЖДПУ, вул. Велика Бердичівська, 40