Міністерство освіти і науки України

ДОНБАСЬКА ДЕРЖАВНА МАШИНОБУДІВНА АКАДЕМІЯ

А.Г.Фокін, І.А. Гетьман

ПРОГРАМУВАННЯ В СЕРЕДОВИЩІ PASCAL

Навчальний посібник

для студентів вищих навчальних закладів

Рекомендовано

Міністерством освіти і науки України як навчальний посібник

для студентів вищих навчальних закладів

Краматорськ 2006

УДК 004.43

ББК 22.18

Ф 74

Рецензенти:

С.О.Калоєров, д-р фізико-математичних наук, професор
(Донецький національний університет)

Гриф надано Міністерством освіти і науки України

Лист № від

Фокін А.Г., Гетьман І.А.

Ф 74 Програмування в середовищі PASCAL: Навчальний посібник для студентів вищих навчальних закладів. – Краматорськ: ДДМА, 2006. – с.

ISBN

Розроблені та сформульовані завдання для шести лабораторних робіт, що дозволяють освоїти прийоми і методи алгоритмізації задач, отримати навики складання і відладки програм для ЕОМ на мові PASCAL.

УДК 004.43

ББК 22.18

ISBN

ЗМІСТ

10 Питання для контролю

СПИСОК ЛІТЕРАТУРИ, ЩО РЕКОМЕНДУЄТЬСЯ

Вступ

Комп’ютер – це тільки пристрій для обробки даних. Він може **лише виконувати заздалегідь підготовлені команди.** Тому, щоб комп’ютер міг виконати що-небудь корисне, для нього необхідно розробити послідовність команд на тій мові, яку він розуміє. Така послідовність команд називається програмою. Процесор персонального комп’ютера орієнтований на виконання команд, поданих у вигляді двоїстих кодів. Такий код дуже зручний для комп’ютера, але занадто незручний для людини. Його використовували лише на початковому етапі розвитку комп’ютерної техніки. Більш ефективним є використування спеціальних мов програмування.

Мова програмування – це штучна мова. Від природної мови вона відрізняється обмеженою кількістю слів та дуже строгими правилами запису команд (операторів). Мови програмування можна поділити на два рівні: мови низького та високого рівня. Кожний оператор мови низького рівня (***Асемблера***) відповідає одній машинній команді. Тобто ***Асемблер*** повністю відображає систему команд комп’ютера і є машиннозалежною мовою. У мові високого рівня (алгоритмічній мові), навпаки, одному оператору мови відповідає декілька машинних команд, що дає можливість створювати машиннонезалежні мови програмування.

Перетворення програми на алгоритмічній мові в машинні команди здійснює спеціальна програма – компілятор.

Однією з найбільш поширених мов програмування високого рівня є ***Pascal***, яку створив у 1971 р. співробітник Інституту інформатики у Цюріху Ніколаус Вірт.

Для цієї мови розроблено декілька компіляторів. Найбільшого поширення знайшли компілятори фірми **Borland,** яку, між іншим, заснував колишній студент Н.Вірта. Ця фірма випустила декілька версій компіляторів ***Pascal***, найбільш досконалою з яких була версія ***Turbo Pascal 7.0***. (1992 р.). Далі на зміну операційної системи **MS DOS** прийшла **Windows**. Методи

програмування суттєво змінилися. З'явились принципово нові методи програмування. Це знайшло відображення у нових системах програмування. У 1993 р. фірма **Microsoft** представила перше візуальне середовище програмування ***Visual Basic***. У відповідь на це фірма **Borland** в 1995 році випустила першу версію системи візуального програмування ***Delphi***. Основу цієї системи складає мова програмування ***Pascal.*** Тому і при розробці програм для **Windows** в середовищі ***Delphi*** необхідне досконале знання мови ***Pascal.***

Основна мета даного навчального посібника – придбання студентами практичних навиків при взаємодії з персональним комп'ютером і розв'язання задач з використанням мови програмування ***Turbo Pascal 7.0***.

Вивчення основ програмування в середовищі ***Pascal*** є першим етапом при вивченні програмування в середовищі ***Delphi*** і виділено в окремий модуль. Матеріал даного посібника розрахований на модуль обсягом 2,5 кредити ECTS (90 годин загального навантаження, включаючи 45 годин аудиторного). Виділення в окремий модуль вивчення основ ***Pascal*** дозволяє студентам на першому етапі зосередити усю увагу на освоєнні методів алгоритмізації, створення та налагодження програм, не відволікаючись на великі можливості і прийоми візуального програмування.

У навчальному посібнику наведені короткі відомості про роботу в середовищі ***Turbo Pascal***, а також короткі теоретичні відомості про мову ***Pascal.*** Ці відомості не охоплюють всіх можливостей мови, проте вони достатні для придбання практичних навиків у складанні програм і виконання лабораторних і розрахунково-графічних робіт, передбачених посібником. Практична частина посібника подана у вигляді лабораторних робіт, кожна з яких присвячена певній темі. Кожній лабораторній роботі передує короткий теоретичний матеріал, що відноситься саме до цієї роботи. Наводяться приклади виконання роботи, що сприяє більш ефективному освоєнню матеріалу. Кількість завдань на кожну лабораторну роботу співпадає з

чисельністю академічної групи, що дозволяє видати кожному студенту індивідуальне завдання.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Порядок виконання лабораторних і розрахунково-графічних робіт

1 Вивчити короткі теоретичні відомості за темою роботи.

2 Ознайомитися із завданням.

3 Визначити вхідні, проміжні і вихідні дані.

4 Зазначити підпрограми (процедури або функції).

5 Розробити алгоритм всієї програми і кожної підпрограми окремо. Скласти блок-схеми.

6 Написати текст програми.

7 Ввести текст програми в ЕОМ.

8 Шляхом багаторазового прогону програми налагодити її.

9 Оформити звіт.

10 Подати викладачу програму на ПЕВМ і звіт про роботу.

11 Захистити роботу і отримати оцінку.

1.2 Вимоги до оформлення звіту

Звіт складається на стандартних аркушах споживацького формату А4 про кожну лабораторну роботу (ЛР) або розрахунково-графічну роботу (РГР) окремо. Якщо використовуються аркуші паперу з розмірами, що не відповідають стандарту (наприклад, розгорнені аркуші учнівського зошита), то вони мають бути приведені до стандартного розміру. Кожна робота захищається окремо. Після захисту робота залишається у студента. У кінці семестру, після захисту всіх передбачених планом робіт звіти про РГР збираються всі разом, зшиваються з додаванням загального титульного листа і подаються викладачу для відмітки про захист і розрахунку рейтингів за модуль. Звіти про ЛР також зшиваються. Проте, на розсуд викладача, допускається оформлення звітів про ЛР в окремому зошиті. Оформлені і

захищені роботи подаються викладачу, що приймає залік або екзамен з дисципліни.

1.3 Зміст звіту

Звіт повинен містити:

- заголовок;
- мету роботи;
- завдання;
- таблицю вхідних, вихідних і проміжних даних;
- блок-схеми програми (головної програми та всіх її підпрограм);
- текст програми. Допускається подання тексту надрукованого на комп'ютері. Кожна програма (підпрограма) повинна включати коментарі, що містять:

а) номер ЛР (або РГР), прізвище студента і позначення групи - тільки для основної програми;

б) коротке найменування програми або підпрограми (виконувані нею функції);

- короткі висновки з роботи.

У звіті можна навести результати розрахунків на комп'ютері (краще у вигляді роздруку).

1.4 Порядок виконання елементарних дій при виконанні будь-якої роботи на ЕОМ

У систему програмування ***Turbo Pascal*** фірми **Borland** вбудований простий, але достатньо зручний, текстовий редактор для створення текстів програм. Не виходячи з нього, можна компілювати програми, знаходити помилки і тут же їх виправляти, компонувати програми з окремих частин, запускати налагоджену програму. Для цього призначено системне меню, що

активізується натисненням клавіші ***F10.*** Деякі команди дублюються функціональними клавішами.

Нижче описані дії при виконанні основних функцій при розробці програм.

1.4.1 Завантаження системи *Turbo Pascal*

1 Увійти до папки (каталогу), що містить систему ***Turbo Pascal*** (..\tp7\BIN).

2 Активізувати файл **turbo.exe**, при цьому у верхній частині екрану з'явиться меню команд системи, яке активізується натисненням функціональної клавіші ***F10*** або клацанням миші на потрібному пункті (команді); зміна команд (вибір необхідного пункту меню) за допомогою клавіш-стрілок і клавіші ***Enter*** (рис. 1).

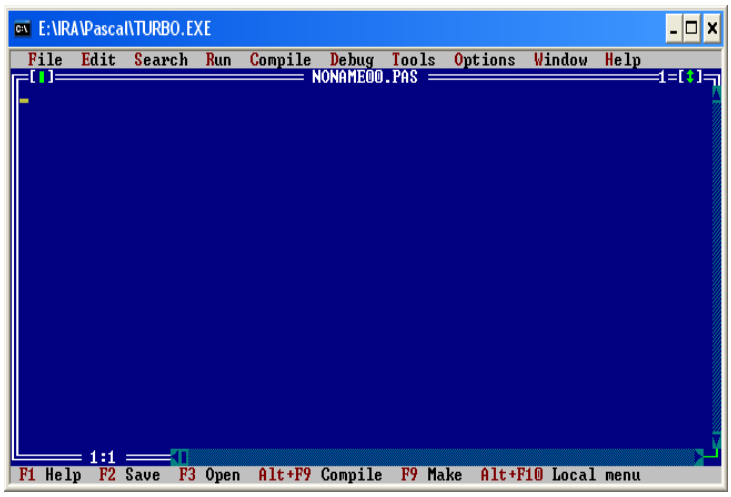


Рисунок 1 – Вікно ***Turbo Pascal***

1.4.2 Створення (введення) нової програми

1 Натиснути клавішу ***F10*** (вхід в меню).

2 Встановити курсор на команду **File** і активізувати її шляхом натиснення на клавішу ***Enter***.

3 У рамці з переліком команд, що з'явиться на екрані, встановити курсор на команду **NEW** і активізувати її (відкриється нове вікно).

4 Увести текст програми.

5 Після закінчення введення програми натискувати клавішу ***F10*** і активізувати знову команду меню **File,** в підменю вибрати команду **Save as...** і зберегти програму відповідно до пункту 4. Збереження програми можна виконати і раніше (на початковому етапі створення програми), час від часу зберігаючи проміжний стан програми за допомогою підпункту **Save** команду меню **File** або клавіші ***F2***. Це забезпечить можливість відновлення програми при її ненавмисному знищенні в процесі створення.

6 Для компіляції і виконання програми можна або за допомогою клавіші ***F10*** ввійти в меню системи ***Turbo Pascal*** і активізувати команду **Run** або натиснути комбінацію клавіш ***Ctrl+F9***; натиснення клавіші ***F9*** починає тільки компіляцію програми для виявлення помилок на ранній стадії створення програми.

7 Для спостереження результатів виконання програми натиснути комбінацію клавіш ***Alt+F5.***

1.4.3 Корегування (виправлення) програми

Якщо в програмі знайдені помилки, то для їх усунення слід виконати наступні операції:

1 Усунути помилки в програмі.

2 Новий варіант програми записати на диск, для чого натиснути клавішу ***F10,*** активізувати команду **File,** в підменю вибрати команду **Save**; або натиснути клавішу ***F2***.

3 Для компіляції, виконання і спостереження за результатами виконати раніше описані дії.

1.4.4 Збереження програми

Якщо необхідно записати на диск щойно створений програмний файл, то треба виконати наступні дії:

1 В опції **File** активізувати команду **Change dir...** і вибрати особистий каталог студента, що містить індивідуальні тексти програм студента; для переміщення деревом каталогів можна використовувати стрілки переміщення курсору або мишку; щоб розкрити вибраний каталог, слід натиснути клавішу **Enter** або зробити подвійне щиглик на ньому.

2 В опції **File** активізувати команду **Save as...** і у вікні діалогу, що з'явиться, в полі ***Save file as*** ввести ім'я файла, під яким буде збережено текст програми; при цьому є можливість за допомогою поля ***Files*** змінити поточний каталог, не вдаючись до команди **Change dir...**.

Цей же порядок дій використовується і у тому випадку, коли необхідно зберегти існуючий файл під іншим ім'ям.

Для збереження поточного програмного файла слід в опції **File** активізувати команду **Save** або натиснути клавішу ***F2***. Рекомендується при тривалій роботі періодично зберігати поточний файл.

1.4.5 Завантаження в редактор раніше збереженого тексту

1 Натиснувши клавішу ***F10,*** активізувати опцію **File** і за допомогою команди **Change dir...** вибрати особистий каталог студента, як вказано в пункті 4.

2 Натиснути клавішу ***F10*** і активізувати опцію **File,** в підменю якої вибрати команду **Open...** (***F3***) і активізувати її.

3 У полі ***Name*** вікна діалогу, що з'явиться, ввести ім'я файла, який слід завантажити, або вибрати це ім’я із запропонованого списку; при цьому є можливість змінити поточний каталог, не вдаючись до команди **Change dir...**.

Якщо після виходу з системи ***Turbo Pascal*** не було запису нового файла, то при повторному вході в неї попередній файл може бути збережений в редакторі, тоді його завантаження не потрібне.

1.4.6 Друкування програми і результатів її роботи

Після налагодження програми, тобто отримання результатів обчислень на екрані дисплея, необхідно вивести на принтер її текст і результати. Виведення тексту програми на принтер можна здійснити таким чином:

1 встановити курсор на верхній рядок тексту (зліва від тексту), що виводиться, і натиснути клавіші ***Ctrl+K-B*** (натискують одночасно клавіші ***Ctrl+K***, а потім – клавішу ***B***);

2 встановити курсор нижче за останній рядок тексту, що виводиться, на один рядок (зліва від тексту) і натиснути клавіші ***Ctrl+K-K***;

3 натиснути клавіші ***Ctrl+K-P***. Текст буде роздрукований.

Для виведення результатів рішення задачі на принтер необхідно в програму внести наступні зміни:

1 після заголовка програми ввести оператор **Uses printer**;

2 в операторах виведення даних перед списком виведення записати **lst;**

3 запустити програму на виконання (***Ctrl+F9)***. Результати будуть роздруковані.

1.4.7 Робота з фрагментами тексту

Система ***Turbo Pascal*** версії 7.0 допускає багатовіконну роботу, тобто одночасно може бути завантажено декілька програм, кожна в окреме вікно. Перемикання між вікнами здійснюється за допомогою клавіші ***F6***. Видалення поточного вікна – ***Alt+F3***. Для доступу до інших функцій роботи з вікнами слід за допомогою клавіші ***F10*** ввійти у меню системи і активізувати команду ***Windows***.

Редактор ***Turbo Pascal*** допускає роботу з фрагментами тексту. Будь-який фрагмент можна, використовуючи буфер обміну, скопіювати в будь-яке місце програми в будь-якому вікні:

- виділення фрагмента здійснюється клавішами-стрілками при натиснутій клавіші ***Shift*** (не відпускати, поки не буде завершено виділення);
- копіювання виділеного фрагмента в буфер обміну ***Ctrl+Ins*** або ***Shift+Del***. В останньому випадку виділений фрагмент буде видалений з програми;
- копіювання вмісту буфера обміну в програму в точку, де знаходиться курсор ***Shift+Ins***;
- видалення виділеного фрагмента – ***Ctrl+Del***.

Доступ до команд копіювання можливий через меню: натиснувши клавішу ***F10,*** вибрати команду **Edit,** з'явиться підменю команд редагування тексту.

Зауваження. Наведена вище послідовність операцій не оптимальна, але найбільш проста для першого знайомства з ПЕВМ.

1.4.8 Робота в середовищі Delphi у режимі консолі

Середовище програмування ***Delphi*** орієнтовано на створення додатків ***Windows***, що використовують графічний інтерфейс і мають практично необмежену складність. Однак, ця система надає також можливість створення і налагодження простих програм у стилі ***MS DOS***. Це так називані ***консольні додатки***. Зовні вони виглядають як програми з текстовим інтерфейсом, але здатні звертатися до більшості функцій ***Windows***. Консольні додатки можна розглядати як навчальні програми при вивченні мови програмування ***Pascal***.

Для створення консольного додатка необхідно ввійти в середовище ***Delphi*** і виконати команду **File⇒New⇒Other** чи клацнути кнопку **New** у панелі інструментів. У вікні, ***New Items*** (Створення програми), що з'явиться за цією командою**,** вибрати значок ***Console Application*** (Консольний додаток) і клацнути **OK** (рис. 2.).

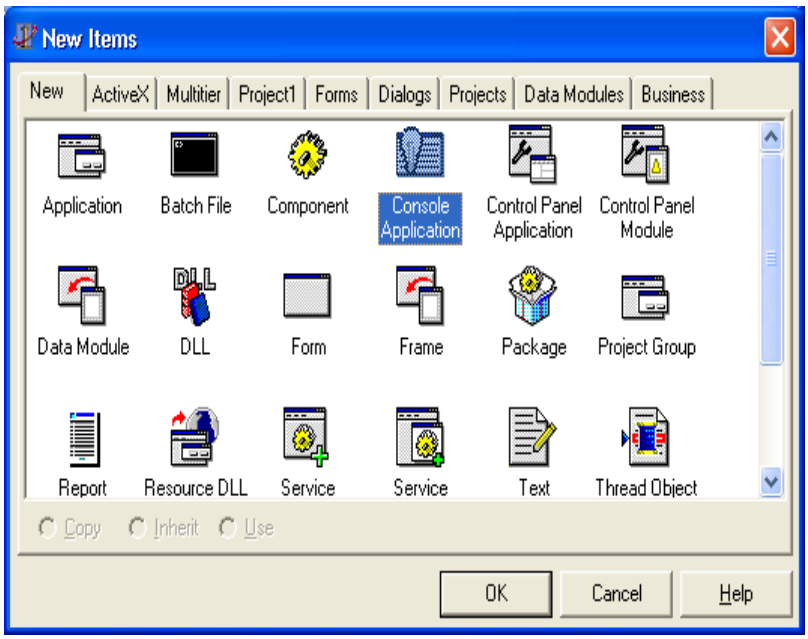


Рисунок 2 – Вибір категорії, до якої відноситься створювана програма

Відкриється вікно редактора коду, в якому **Delphi** помістить автоматично згенеровану заготовку майбутньої програми (рис. 3.).

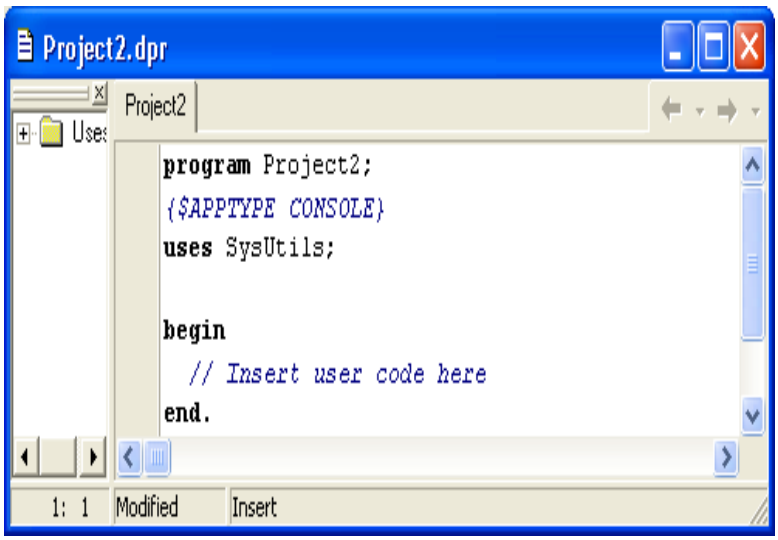


Рисунок 3 – Вікно редагування для вводу тексту програми

Згенеровані системою ***Delphi*** рядки рекомендується не змінювати. Перший рядок згенерованого коду – заголовок програми. Далі у вигляді коментарю йде директива компілятора. Вона відрізняється від звичайного коментарю тим, що відразу за символом-ознакою початку коментарю “{“ йде

символ “$”. За цим символом записані ключові слова, що задають настройки компілятора. У даному випадку директива {$APPTYPE CONSOLE} наказує компілятору створити консольний додаток. Наступний рядок за допомогою ключового слова **uses** задає підключення стандартного модуля **SysUtils**. Після цього рядка можна вводити описи змінних, констант, підпрограм і т.п. Далі йде власне програма – розділ операторів. ***Delphi*** генерує тільки границі цього розділу – рядки **begin** і **end**. Усередині розділу операторів ***Delphi*** розміщає коментар ***Insert user code here (вставте сюди свій вихідний код).*** Цей коментар можна видалити.

Перед введенням коду програми рекомендується зберегти текст програми в особистій папці студента. Для чого виконати команду **File⇒Save.** У вікні діалогу, що з'явилося**,** ***Save*** вибрати особисту папку студента і ввести ім'я майбутньої програми, наприклад**,** ***Lab1*****.** У результаті ***Delphi*** перейменує програму (в операторі **program** заготівки програми). Їй буде привласнене ім'я, під яким вона збережена на диску:

```
program Lab1;
{$APPTYPE CONSOLE}
uses SysUtils;
begin
// Insert user code here
end.
```

Для запуску програми на виконання виконати команду **Run⇒Run,** клацнути відповідну кнопку на панелі інструментів, чи натиснути клавішу **F9** на клавіатурі. Для виконання консольного додатка відкривається окреме вікно, що після завершення роботи додатка (програми) відразу ж закривається. Тому, щоб побачити результати роботи додатку, варто затримати його завершення. Це можна зробити, додавши наприкінці програми додатковий оператор введення, що не дасть програмі завершитися, поки не буде введене якесь дане. Інший спосіб – визначити точку контрольної зупинки на останньому рядку програми (рядку **end.**). Для цього досить клацнути на цьому рядку ліворуч у спеціальному полі маркера. На

місці щиглика з'явиться червоний кружок, а сам рядок виділиться червоним кольором. При виконанні програми її робота на цьому місці тимчасово припиняється. Можна переглянути результати роботи програми. Для продовження роботи слід натиснути клавішу **F9** чи відповідну кнопку (**Run**) на панелі інструментів.

До речі, точку контрольної зупинки можна створити на будь-якому виконуваному операторі. При зупинці програми у точці зупинки можна переглянути поточне значення будь-якої змінної, якщо вказати на неї курсором миші. Значення з'явиться в ярличку поруч з курсором.

2 ПОНЯТТЯ АЛГОРИТМУ. БЛОК-СХЕМИ

У програмуванні дуже важливим є поняття алгоритму. Алгоритм – це точно визначена послідовність процедур, яка гарантує одержання результату за скінчену кількість кроків. Програма, як зазначено вище, – це послідовність команд, зрозумілих комп'ютеру. Таким чином, алгоритм є більш абстрактним поняттям, ніж програма. Поняття алгоритму виникло задовго до появлення комп'ютерів.

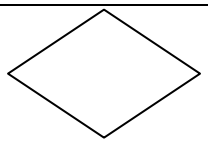
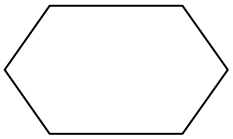
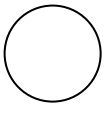
З точки зору програмування алгоритм однозначно визначає процес перетворення початкових даних у результат вирішення задачі. Розробка алгоритму передує складанню програми. При цьому в алгоритмі основна увага приділяється розв'язанню задачі, а не можливостям конкретної мови чи комп'ютера. Але сам алгоритм ще не забезпечує розв'язання задачі. Одержати результат можна лише тоді, коли алгоритм буде перетворено в конкретну програму на конкретній мові. Тому розробка алгоритму – це одна із стадій, обов'язкова при розробці програми.

Є різні способи запису алгоритму. Найбільш поширені з них:

- словесний запис;
- графічні схеми алгоритмів (блок-схеми);
- псевдокод (формальні алгоритмічні мови);
- структурограми.

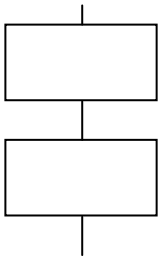
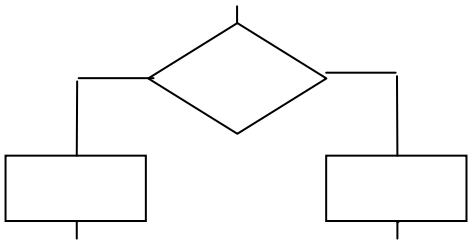
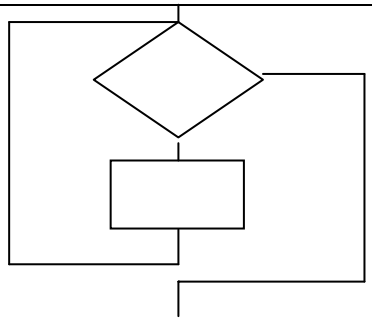
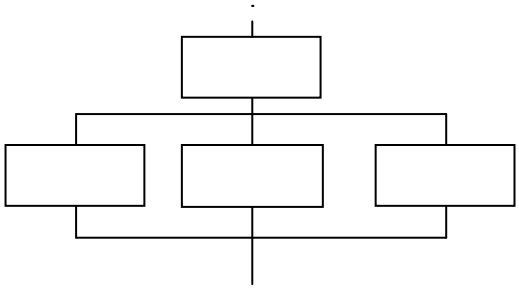
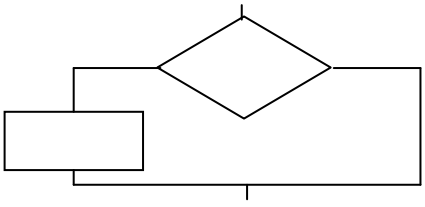
Блок-схема – це графічне зображення алгоритму, доповнене елементами словесного запису. На блок-схемі кожний пункт алгоритму зображено відповідною геометричною фігурою. У таблиці 1 наведено графічні елементи, на яких компонуються блок-схеми, їх назви та символи.

Таблиця 1 – Графічні елементи блок-схем

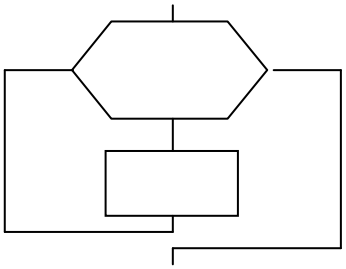
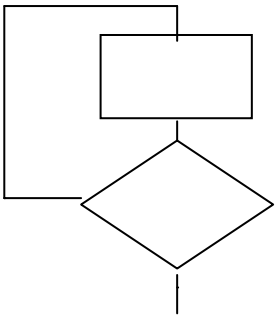
Назва блоку	**Блок**	**Функція, яка відображується**
Початок-кінець		Початок, кінець, вхід, вихід в підпрограмах
Блок введення-виведення		Уведення даних чи виведення результатів на екран
Блок виведення		Виведення даних на друк
Процес		Обчислення чи послідовність обчислень
Визначений процес		Виконання підпрограми
Альтернатива		Перевірка умов
Модифікація		Початок циклу
З’єднувач		Розрив ліній потоку інформації в межах однієї сторінки

У таблиці 2 наведені основні базові елементарні структури для складання блок-схем.

Таблиця 2 – Базові структури блок-схем

Назва типу структури	**Зображення**
Основні	
Послідовність	
Розгалуження (вибір)	
Цикл із передумовою	
Додаткові	
Вибір варіантів	
Скорочений запис розгалуження	

Продовження табл.2

Назва типу структури	**Зображення**
Цикл із параметром	
Цикл із постумовою	

3 ОСНОВНІ КОНСТРУКЦІЇ МОВИ PASCAL

3.1 Алфавіт мови

У програмі можна використовувати тільки символи, що входять до алфавіту мови:

1 26 латинських букв (не має значення, прописні чи рядкові);

2 символ підкреслення (_); цей символ використовується нарівні з латинськими буквами;

3 цифри від 0 до 9;

4 розділові знаки: крапка (**.**), крапка з комою (**;**), двокрапка (**:**), кома (,), апостроф (’);

5 знаки арифметичних операцій: плюс (**+**), мінус (**-**), множення (*****), ділення (**/**);

6 знак конкатенації (об’єднання) &;

7 знаки операцій відношення: більше (>),більше чи дорівнює (>=), менше (<), менше чи дорівнює (<=), рівно (=), не рівно (<>);

8 круглі дужки: **( )**, фігурні дужки **{ }**, квадратні дужки **[ ]** ;

9 пропуск (пробіл);

10 деякі спеціальні символи: ^,@,$,#;

11 пари спеціальних символів: “:=”, “(*”, “*)”, “..”, “//”.

Слід зауважити, що пари спеціальних символів, а також вищезазначені пари символів відношення “<=”, “>=” та “<>” розглядаються як один символ. Між ними не можна вставляти пропуск.

Особливе місце в мові ***Pascal*** займає пробіл. Цей символ розглядається як обмежувач елементів програми: ідентифікаторів, констант, зарезервованих слів, тощо (див. нижче). Декілька, стоячих підряд пробілів, сприймаються як один пробіл (це не відноситься до рядкових констант – сукупності різних символів, обмежених лапками. Символ ***Enter*** сприймається як пробіл. Таким чином, для поліпшення наочності програми між окремими її елементами

можна вставляти замість одного пробілу різну їх кількість. Це відноситься і до початку рядка: починати текст рядка можна з різного символу.

У тексті, який з двох сторін обрамляють апострофи або фігурні дужки, можна використовувати будь-які символи, що є на клавіатурі дисплея.

3.2 Елементи програми

Елементи програми – це найменші неподільні її частини, що мають значення для компілятора. До елементів відносяться:

- зарезервовані слова;
- ідентифікатори;
- типи;
- константи;
- змінні;
- мітки;
- підпрограми
- коментарі.

Зарезервовані слова – це англійські слова, що вказують компілятору на необхідність виконання певних дій. Зарезервовані слова не можуть використовуватися в програмі ні для яких інших цілей крім тих, для яких вони призначені.

Ідентифікатори – це слова, якими в програмі позначаються будь-які інші елементи програми, крім зарезервованих слів чи коментарів. Можуть складатися з латинських букв, цифр і знака підкреслення. Ніякі інші символи чи спеціальні знаки не допускаються. Починатися ідентифікатор повинен з букви. Великі та малі букви в ідентифікаторі не розрізняються: так, ідентифікатори **NAME, Name** і **name** є ідентичними, тобто будуть сприйматися компілятором як одне і те ж. Довжина ідентифікатора може бути будь-якою, але істотними є тільки перші 63 символи. Таким чином, ідентифікатори не можуть складатися з декількох слів (адже пробіли

неприпустимі) і не можуть містити символів кирилиці. Зарезервовані слова не можна використовувати в якості ідентифікаторів.

Типи – це спеціальні конструкції мови, що розглядаються компілятором як зразки для створення інших елементів програми, таких як змінні, константи, функції. Різний тип визначає дві важливі для компілятора речі: обсяг пам'яті, що виділяється для розміщення елемента (константи, перемінною чи результату, що повертається функцією), набір припустимих для елемента значень, і набір припустимих над елементом дій.

Константи визначають ділянки пам'яті, що не можуть змінюватися в процесі виконання програми.

Змінні зв'язані зі змінюваними ділянками пам'яті, тобто з такими її ділянками, уміст яких буде змінюватися під час роботи програми.

Мітки – це імена операторів програми. Використовуються дуже рідко і тільки для вказівки переходу до наступного оператору.

Підпрограми – це спеціальним чином оформлені фрагменти програми. Важливою особливістю підпрограм є їхня відносна незалежність від іншого тексту програми. Підпрограми – це засіб структурування програми, тобто, розчленовування її на декілька багато в чому незалежних фрагментів.

Коментарі використовуються для пояснення тих чи інших фрагментів програми. Компілятором вони ігноруються. Наявність коментарів у тексті програми робить її зрозумілішою і дозволяє пояснювати особливості реалізації програми чи окремих її фрагментів. Коментарем у ***Pascal*** вважається будь-який текст, що розміщений у фігурних дужках. У коментарях допускаються використання будь-яких символів, включаючи символи кирилиці.

Перелічені елементи програми нижче будуть розглянуті більш детально.

3.3 Дані

Для подання інформації в комп'ютері використовуються двоїсті цифри або біти, які можуть набувати тільки два значення: 0 чи 1. Для зручності прийнято об'єднувати біти в байти. Кожний байт – це 8 бітів. У свою чергу байти можуть об'єднуватись в слова (здебільшого два-вісім байтів). Усі біти слова обробляються одночасно. Інтерпретація слів заздалегідь не визначена: все залежить від того, що з ними буде робитись. Їх можна трактувати і як команди комп'ютеру, і як числові дані, і як деякий текст, тощо. Тому потрібно визначитись, яка інформація міститься в тому чи іншому слові. Тобто описати ділянки пам'яті комп'ютера, у які будуть поміщатися дані.

3.3.1 Види і типи даних

Програма здійснює обробку даних. Як уже відзначалось, дані бувають двох видів – константи і змінні.

Константи – це такі дані, значення яких, будучи визначеними перед виконанням програми, не змінюються на протязі її виконанні. У **Pascal** існує два виду констант: літерали та іменовані константи. Літералами називаються числа, окремі символи та рядки символів. Іменована константа – це фіксоване значення, якому при оголошенні на початку програми присвоюється ім'я (ідентифікатор). Літерали використовуються в програмі своїм значенням, іменовані константи – їх ідентифікаторами.

Змінні – це такі дані, значення яких можуть змінюватися у процесі виконання програми. При присвоєнні змінній нового значення старе значення знищується і втрачається.

Кожне дане (як константа, так і змінна) відноситься до якого-небудь типу. Під **типом даного** розуміється множина його допустимих значень. Тип дозволяє визначити, як буде інтерпретуватися той чи інший набір бітів (слово). Це дає можливість уникнути характерних для мов програмування низького рівня помилок зв'язаних зі спробою виконати арифметичні операції

над символами, інтерпретувати дані як команди, тощо: тип даного визначає також які дії над ним допустимі.

Типи в ***Pascal*** відіграють важливу роль. Це зв'язано з тим, що ця мова створювалась як засіб навчання студентів програмуванню. Оскільки починаючий програміст може легко припуститися помилки чи неточно описати свої дії, компілятор ***Pascal*** повинен був мати засоби контролю за діями програміста, щоб вчасно застерегти програміста від невірних дій. Спочатку типи саме і призначалися для того, , щоб програміст явно вказував компілятору, якого розміру пам'ять йому необхідна і що він збирається з цією пам'яттю робити. Практика застосування типів показала їхню високу ефективність для захисту програми від випадкових помилок, так що практично всі мови програмування високого рівня в тому чи іншому степені реалізують механізм типів.

Існують декілька груп типів. Ми розглядаємо тільки дві з них: прості та структуровані типи даних.

Прості типи не містять в собі інших типів. Дані цих типів можуть одночасно мати тільки одне значення.

З групи простих типів даних, у свою чергу, ми розглянемо тільки декілька основних типів (табл.3).

Таблиця 3 - Основні типи даних

Тип	**Ім'я типу**	**Зміст**
Цілий	**Integer**	Цілі числа в інтервалі -32768 до 32768
Дійсний	**Real**	Дійсні числа в інтервалі від 10^{-38} до
Логічний	**Boolean**	Логічні дані. Можуть приймати значення **True** (істина) або **False** (брехня)
Символьний (літерний)	**Char**	Дані символьного типу. Можуть приймати значення однієї літери з набору символів в ЕОМ
Перераховани		Набір значень, яких може набувати дане

Тип	Ім'я типу	Зміст
й		
Діапазон		Підмножина впорядкованих значень, визначуване початковим і кінцевим значеннями

Слід відмітити, що ***Pascal*** має значно більше простих типів. Це тому, що він передбачає по декілька подібних типів. Наприклад, цілі типи крім типу **Integer** можна описати також як **Byte**, **Longint**, тощо. Кожен з таких типів відрізняється внутрішнім представленням, тобто розміром пам’яті, яка відводиться для даного та діапазоном можливих значень: **Byte** – один байт, **Integer** – два і т.д. Усі прості типи, за винятком дійсного, називаються **порядковими** типами. Вони мають кінчену кількість допустимих значень, тобто значення цього типу можна певним способом впорядкувати. Це означає, що кожному значенню порядкового типу можна присвоїти якесь ціле число – порядковий номер. Таким чином встановлюється порядок слідування значень “наступний” – “попередній”. Значення порядкових типів в комп’ютері подаються абсолютно точно. На відміну від порядкових типів дані **дійсного типу** в комп’ютері подаються лише з деякою точністю, яка залежить від внутрішнього формату дійсного числа.

З групи **структурованих типів** ми розглянемо рядкові дані (типу **String**), дані типу множина, масиви, записи і файли.

Дане типу **String (рядок)** – це послідовність символів довільної довжини, але не більше 255 символів. Гранична кількість символів у рядку можна перевизначити, вказавши це число після імені типу в квадратних дужках: **String[n]**.

Множина – це набір логічно зв'язаних один з одним значень. Характер зв'язків між значеннями визначається програмою і ніяк не контролюється ***Delphi***. Кількість елементів, що входять до множини , може мінятися від 0 до 255 (множина, що не містить елементів, називається порожньою). Саме змінністю своїх елементів множини відрізняються від масивів і записів.

Дві множини вважаються еквівалентними тоді і тільки тоді, коли всі їхні елементи однакові, причому порядок входження елементів до множини не має значення. Якщо всі елементи однієї множини входять також і до іншої, говорять про включення (входження) першої множини до другої. Порожня множина включається до будь-якої іншої.

Над множинами визначені операції:

- *перетинання множин (*)* – результат містить елементи, загальні для обох множин;
- *об'єднання множин (+)* – результат містить елементи першої множини, доповнені відсутніми елементами з другої множини;
- *різниця множин (-)* – результат містить елементи з першої множини, що не належать другій.

Над множинами визначена також низка логічних операцій (операцій порівняння):

- *еквівалентність* (=); повертає ***true***, якщо обидві множини еквівалентні;
- *нееквівалентність* (<.); повертає ***true***, якщо обидві множини нееквівалентні;
- *входження* (<=) – повертає ***true***, якщо перша множина включена до другої;
- *входження* (>=) – повертає ***true***, якщо друга множина включена до першої;
- *приналежність* (***in***) – це бінарна (одномісна) операція. Повертає ***true***, якщо у множині маються задані значення.

Решта структурованих типів буде розглянута нижче.

Серед типів даних, що використовуються в мові **Pascal,** є стандартні (наперед визначенні) типи і типи, визначувані користувачем.

До **стандартних типів**, що не вимагають попереднього визначення, відносяться типи **Integer, Real, Boolean, Char** і **String**. Решта типів, що використовуються в програмі, має бути визначена або в розділі опису типів

(в цьому випадку кожному визначуваному типу буде привласнено ім'я), або безпосередньо в розділі опису змінних.

Розділ опису типу в цілому має такий вид:

TYPE <ім’я типу>=<опис типу>;

<ім’я типу – це ім’я, яке буде присвоєно новому типу;

<опис типу – визначення нового типу.

До типів, що потребують попереднього визначення належать перерахований тип та тип-діапазон.

Перерахований тип визначає множину ідентифікаторів, що задають усі можливі значення змінних цього типу. Уводиться, щоб спростити роботу з даними, зробити код програми більш зрозумілим, оскільки надає можливість мати справу не з абстрактними числами, а з осмисленими значеннями.

Опис перерахованого типу має вигляд:

TYPE T = (a1, a2 .,an);

де **T** – ім'я типу;

a1, a2 ., an – значення, які може приймати дане цього типу.

Тип-діапазон – це підмножина свого базового типу, в якості якого може виступати різний порядковий тип (крім типу-діапазону, звичайно). Тип-діапазон задається межами своїх значень усередині базового типу.

Опис типу «діапазон» має вигляд:

TYPE T = an .. ak;

де **T –** ім'я типу;

an, ak – початкове і кінцеве значення. Вони мають бути одного порядкового типу.

Приклади:

TYPE T1=(PLUS,MINUS,MULTY,DIVIDE);

TYPE T2=2..10;

TYPE T3=’b’..’z’;

Порядок опису решти типів буде наведений нижче.

3.3.2 Константи

Константи можуть бути цілого, дійсного, символьного типу та типу рядок. Тип константи однозначно визначається її значенням і явно не описується.

У зображенні **цілих констант** присутній лише знак (необов'язковий) і цифри: 121; +457; -2287.

Дійсні константи записуються в двох формах: основній і напівлогарифмічній (з порядком). Замість коми ставиться десяткова крапка. Замість основи степеня 10 ставиться буква Е, що дозволяє чітко відділити мантису від показника степеня і записати всі символи числа в одному рядку. Незначущий нуль перед десятковою крапкою може бути відкинутий. Знак "+" може бути опущений.

Приклади запису дійсних чисел:

Число	Запис в *Pascal*
2,87	2.87
-0,315	-0.315 або -.315
210000,0	2.1E+5, 2.1E5, 21E4, 210000.
-0,00045	-0.00045, -.00045, -45E-5, -.45E-3

Символьна константа – це будь-який символ, що взятий в апострофи, а **рядкова константа** – послідовність будь-яких символів (не більше 255), взятих в апострофи:

’a’; ’відповідь:’; ’значення=’;

Можливе двояке використовування констант:

- безпосереднє використовування значення константи (літерали);
- використовування ідентифікатора константи.

Визначення констант ідентифікаторами здійснюється в **розділі опису констант**:

CONST < Ім'я константи > = <Значення>;

Приклад:

CONST PI=31.592;

TT=’S’;

A=7; B=’відповідь’;

Тут визначені константи: дійсного (PT), символьного (TT), цілого (A) і рядкового (B) типів.

У **Pascal** за умовчанням визначені спеціальні константи:

PI=3.14159265.; MAXINT – найбільше ціле число, рівне 32767.

3.3.3 Змінні

Змінні описуються в **розділі опису змінних**, який починається з ключового слова **VAR**, після якого перелічуються імена ідентифікаторів змінних і їх типи. Загальний вигляд:

VAR < Ім’я змінної . < ім'я змінної > : < тип >;

< **Тип** > – є ідентифікатором (ім'я) типу – стандартного або визначеного раніше користувачем (в розділі **TYPE**). Допускається замість ідентифікатора типу наводити безпосередній опис типу.

Приклад:

VAR TOP: INTEGER; X, У : REAL;

A: ARRAY[1..10] INTEGER;

B : ARRAY[1..5,1..7] REAL;

У підпрограмах можуть використовуватися змінні, описані в заголовку підпрограми – **формальні параметри** (див. п. 3.7.3).

3.4 Стандартні функції

У процесі вирішення задач часто виникає необхідність в обчисленні елементарних функцій. Для звернення до функції необхідно у виразі записати ідентифікатор функції і в круглих дужках – аргумент. Аргументами функції можуть бути константи, змінні, функції або вирази. Для тригонометричних функцій кут слід задавати у радіанах. Деякі стандартні функції для роботи з числовими даними перелічені в таблиці 4.

Таблиця 4 - Стандартні функції

Функція	**Математичний запис**	**Запис в Pascal**
Синус	**sin(x)**	**SIN (X)**
Косинус	**cos(x)**	**COS (X)**
Арктангенс	**arctg(x)**	**ARCTAN (X)**
Абсолютне значення	**| x |**	**ABS (X)**
Корінь квадратний	$\sqrt{X}$	**SQRT (X)**
Обчислення експоненти	e^X	**EXP (X)**
Натуральний логарифм	**ln (x)**	**LN (X)**
Зведення в квадрат	x^2	**SQR (X)**
Привласнення знака	**sign (x)**	**SGN (X)**
Обчислення числа π	π	**PI**
Обчислення цілої частини	**[x]**	**TRUNC (X)**
Обчислення дробової частини		**FRAC (X)**
Округлення до цілого		**ROUND (X)**
Ініціалізація генератора псевдо випадкових чисел		**RANDOMIZE**
Псевдо випадкове число, рівномірно розподілене в діапазоні 0...1		**RANDOM**
Псевдо випадкове ціле число, рівномірно розподілене в діапазоні 0...x-1		**RANDOM(x)**

Тип результату функцій **ABS(X)** і **SQR(X)** визначається типом аргументу: цілий або дійсний. Результат функцій **RANDOM(X), TRUNC(X)** і **ROUND(X)** має цілий тип. Решта функцій дає дійсний результат.

Для обчислення інших функцій слід використовувати відповідну математичну залежність, наприклад:

$$x^a = e^{a\ln(x)}$$

$$\arcsin(x) = \operatorname{arctg}\left(\frac{x}{\sqrt{1-x^2}}\right)$$

$$\arccos(x) = \operatorname{arctg}\left(\frac{\sqrt{1-x^2}}{x}\right)$$

$$\operatorname{arcctg}(x) = \operatorname{arctg}\left(\frac{1}{x}\right)$$

$$\operatorname{tg}(x) = \frac{\sin(x)}{\cos(x)}$$

До **даних порядкового типу** можна також застосовувати такі функції:

Ord(x) – повертає порядковий номер значення x. Для цілих значень ця функція повертає саме значення **x**, тобто **Ord(x) = x**. Для даних логічного типу ця функція повертає 0 (***false***) або 1 (***true***), для символьних даних функція повертає значення від 0 до 255 – код символу.

Pred(x) – повертає попереднє значення, тобто значення, що відповідає порядковому номеру **Ord(x)-1.**

Succ(x)– повертає наступне значення, тобто значення, що відповідає порядковому номеру **Ord(x)+1.**

Для роботи з рядками (даними типу **String**) можна застосовувати функції, наведені в таблиці 5.

Таблиця 5 - Функції для роботи з рядками

Функція	**Призначення**
Concat(S1,S2, …, Sn)	Функція типу **String**. Повертає рядок, що являє собою зчеплення (конкатенацію) всіх рядків **S1,S2...Sn**: Якщо отриманий рядок містить більше 255 символів, вона усікається (обривається) після 255-го символу
Copy(St,IndX,k)	Функція типу **String.** Створює підрядок: копіює з рядка **St k** символів, починаючи із символу з

	номером **IndX**; якщо **IndX** більше довжини рядка **St**, повертається порожній рядок
Pos(Subst,St)	Функція типу **Integer**. Знаходить у рядку **St** перше входження підрядка **Subst** і повертає номер позиції, з якої вона починається: якщо підрядок не знайдений, повертається нульове значення
Delete(St,IndX,k)	Процедура. Видаляє з рядка **St** підрядок в **k** символів, починаючи із символу **IndX.**
Insert(Subst,St,IndX)	Процедура. Вставляє підрядок **Subst** у рядок **St**, починаючи із символу з номером **IndX**. Якщо отриманий рядок містить більше 255 символів, усі символи після 255-го відкидаються
Length(St)	Функція типу **Integer**. Повертає поточну довжину рядка **St.**
Str(x,St)	Процедура. Перетворить число **x** у строкову змінну, результат заноситься у рядок **St.**
Val(St,x,R)	Процедура. Перетворить символьну величину типу **String** з рядка в її чисельне подання й результат перешкодить у змінну **x.** Якщо рядок записаний з помилкою, то індекс (позиція) невірного символу заноситься до змінної **R**; у противному випадку **R** дорівнює нулю

3.5 Вирази

Вирази формуються відповідно до низки правил з констант, змінних, функцій (як стандартних, так і визначених в програмі), знаків операцій і круглих дужок. Дужки використовуються для зміни пріоритетів операцій.

Розрізняються вирази арифметичні, рядкові та логічні.

3.5.1 Арифметичний вираз

В арифметичному виразі можуть використовуватися операнди (змінні, константи і функції) тільки арифметичного типу: цілі або дійсні.

Знаки арифметичних операцій:

"+" (додавання)

"–" (віднімання)

"*" (множення)

"/" (ділення),

DIV (цілочислове ділення),

MOD (залишок від ділення).

Дії виконуються зліва направо з дотриманням наступного старшинства (пріоритету):

а) мультиплікативні операції (* /, DIV, MOD);

в) адитивні операції (+ , –).

Тип результату виразу залежить від типів операндів, а також операцій: якщо хоча б один операнд дійсний або використовується операція ділення (/), – вираз має дійсний тип (**Real)**, інакше – цілий **(Integer)**.

Результат операцій DIV і MOD завжди типу INTEGER (цілий). Аргументи цих операцій теж мають бути цілого типу. Перша з цих операцій повертає цілу частину від ділення, а друга – залишок. Так 7 div 3=2, а 7 mod 3=1.

Приклади.

Математичний запис:

а) $\sqrt{1+\ln(3x)+\cos(A-T)}$;

б) $2^x \cos(3x) - 3^x \sin(3x)$.

Запис на ***Pascal***:

а) SQRT(1+LN (1.3 * X) + COS (A - T));

б) EXP(X * LN (2))* COS(B * X) - EXP(X *LN (3))* SIN (B * X).

3.5.2 Рядковий вираз

Рядковий вираз утворюється з операндів рядкового (**STRING**) і символьного (**CHAR**) типів, над якими допускається тільки одна операція – конкатенація (зчеплення). Ця операція позначається символом ”+”. Результат завжди має тип **STRING** (рядок).

Приклад:

A:=’TURBO’;

B:=’PASCAL’;

K1:=A+’-’+B;

У результаті змінна K1 отримає значення: ’TURBO-PASCAL’

3.5.3 Логічний вираз

Логічний вираз часто називають **умовою**. Для його побудови використовуються операнди логічного типу: змінні або функції логічного (**BOOLEAN**) типа або відношення.

Відношення призначені для порівняння двох величин (вони мають бути порівнянних типів). Результат порівняння має логічний тип. Допустимі 6 операцій відношення:

"=" (дорівнює);

"<>" (не дорівнює);

"<" (менше);

">" (більше);

"<=" (менше або дорівнює);

">=" (більше або дорівнює).

У порівнянні можуть брати участь і вирази.

У логічному виразі використовуються наступні логічні операції: **NOT** (заперечення), **AND** (логічне **і**), **OR** (логічне **або**).

Обчислення логічного виразу здійснюється зліва направо з дотриманням наступного старшинства:

а) відношення (його слід укласти в круглі дужки);

б) заперечення (**NOT**);

в) логічне **і** (**AND**);

г) логічне **або** (**OR**).

Результат має логічний тип (**BOOLEAN**).

Приклад:

```
A:=3.5;
B:=4.1;
C:=2;
D:=’Дорога’;
L:=(A+B>C) and (D<>’Мост’);
```

У результаті змінна L отримає значення **TRUE** (Істина).

Слід зауважити, що відношення – це бінарна операція. Використовується для порівняння саме двох величин. Тому в ***Pascal*** немає подвійних порівнянь, як це має місце в математиці. Такі подвійні порівняння слід розбити на два відношення. Наприклад, математичній вираз: $a \le x \le b$

в ***Pascal*** необхідно записати так: (a<=x) and (x<=b)

3.6 Структура програми

Програма на мові ***Pascal*** складається з двох частин: описів (об’явлень) та інструкцій (операторів). За допомогою об’явлень програміст сповіщає компілятор, які дані будуть використані в програмі. Після об’явлень ідуть оператори. Вони потрібні для опису операцій, які виконуються над даними.

Структурно програма на мові ***Pascal*** складається із заголовка, блоку і закінчується крапкою. Блок, у свою чергу, містить розділи описів і розділ операторів:

```
PROGRAM  <ім'я>;
USES   <ім'я>,...,<ім'я>;
```

LABEL <Мітка>,...,<Мітка>;
CONST <ім'я константи> = <константа>;
⋮ <ім'я константи> = <константа>;
TYPE <ім'я типу> = <тип>;
⋮ <ім'я типу> = <тип>;
VAR <ім'я змінної>,...,<ім'я змінної>:<тип>;
⋮ <ім'я змінної>,...,<ім'я змінної>:<тип>;
PROCEDURE <заголовок процедури>;
<блок>;
FUNCTION <заголовок функції>;
<блок>;
BEGIN
<оператор>;
⋮ <оператор>
END.

Обов'язковим є тільки розділ операторів. Уся решта елементів програми може бути відсутньою.

Порядок розміщення розділів описів – довільний, єдине правило, яке необхідно витримати, – можна використовувати лише ті ідентифікатори, які перед цим були визначені. Можна визначати декілька однойменних розділів описів.

PROGRAM – заголовок програми. Носить чисто декоративний характер і компілятором ігнорується.

USES – визначає перелік зовнішніх модулів, що використовуються програмою (див. п.3.8).

LABEL – розділ опису міток. Мітка – ціле число без знака, що містить не більше чотирьох цифр або звичайний ідентифікатор. Використовується у розділі операторів для тог, щоб помітити оператор, на який здійснюється перехід. Мітка від оператора, що позначається, відділяється двокрапкою. Усі мітки, що використовуються в програмі, мають бути визначені в розділі опису міток.

CONST – розділ опису констант (див. п.3.3.2).

TYPE – розділ опису типів. Служить для визначення простих і структурованих типів даних, що задаються користувачем.

VAR – розділ опису змінних (див. п.3.3.3).

PROCEDURE, FUNCTION – розділи опису процедур і функцій. Ці розділи присутні в програмі, якщо крім стандартних процедур і функцій в програмі визначаються свої, що є самостійними програмними одиницями, до яких здійснюється звертання з основної програми (див. п.3.7).

РОЗДІЛ ОПЕРАТОРІВ – (тіло програми) є послідовністю виконуваних операторів, відокремлюваних один від одного крапкою з комою і обмежених операторними дужками **BEGIN** і **END**. Оператори описують деякі алгоритмічні дії, які необхідно виконати для вирішення задачі. В одному рядку можна записувати декількох операторів. І навпаки, один оператор може розміщуватися в декількох рядках.

У тих випадках, коли відповідно до правил побудови конструкцій мови допустиме використовування тільки одного оператора, а потрібно виконати декількох операторів, застосовується складовий оператор. **Складовий оператор** – це сукупність послідовно виконуваних операторів, укладених в операторні дужки **BEGIN** і **END**. Усередині операторних дужок оператори також відокремлюються один від одного крапкою з комою. **BEGIN** – це не оператор. Тому після нього крапка з комою не ставиться. Перед **END** крапка з комою допускається, але її наявність необов'язкова. Надалі скрізь, де буде сказано, що можна використовувати один оператор, цим оператором може бути складовий оператор.

У будь-яке місце програми можна вставляти **коментарі**. Це взята у фігурні дужки будь-яка послідовність символів, що не містить закриваючої фігурної дужки. Компілятор коментарі пропускає, не обробляючи. У процесі налагодження програми будь-яку її частину можна виключати тимчасово з розгляду, уклавши у фігурні дужки, тобто перетворивши на коментар.

3.7 Процедури і функції

У програмах часто можна виділити деякі фрагменти, що повторюються. Такі повторення подовжують програму, утрудняють роботу з нею. У мові **Pascal** передбачена можливість об'єднання будь-якої послідовності операторів (як фрагментів, що повторюються в різних місцях програми, так і просто логічно самостійних фрагментів) в окрему підпрограму, до якої можна багато разів звертатися (викликати) з будь-якого місця основної програми, звільняючи останню від винесених до підпрограми фрагментів.

Підпрограма – це спеціальним чином оформлений фрагмент програми. Особливістю підпрограм є їх значна незалежність від іншого тексту програми. Говорять, що властивості підпрограми реалізуються в її тілі. Це означає, що при змінах усередині підпрограми, як правило, немає необхідності що-небудь змінювати поза підпрограмою. Таким чином, підпрограми є засобом структурування програми, тобто розчленовування програми на деякі незалежні фрагменти (підпрограм).

Звернення до підпрограми здійснюється за її ім’ям. При цьому виконання основної програми буде тимчасово припинено, виконаються всі дії підпрограми, і виконання основної програми знову буде продовжено, починаючи з оператора, наступного за оператором виклику підпрограми.

Таким чином, за ім'ям підпрограми ховаються деякі винесені з програми дії, які ініціюються автоматично з появою в програмі звернення до цієї підпрограми.

Будь-яка підпрограма, у свою чергу, може звертатися до інших підпрограм. Рівень вкладення підпрограм практично не обмежений.

Структура підпрограми аналогічна структурі всієї програми, тобто в ній є заголовок, розділи описів і тіло підпрограми (розділ операторів).

Підпрограма має бути описана до того, як вона буде використана в програмі або іншій підпрограмі.

Усі змінні, що використовуються в підпрограмі, можна розділити на три категорії:

- *локальні* змінні – описуються і використовуються тільки усередині підпрограми;
- *глобальні* змінні – описуються в основній програмі, а можуть використовуватися як в програмі, так і в її підпрограмах;
- *формальні* параметри – це зарезервовані в підпрограмі змінні, до яких при зверненні до програми будуть занесені із зовнішньої програми конкретні дані або їх адреси.

Є два різновиди підпрограм – *процедури* і *функції*.

3.7.1 Процедури

У найпростішому випадку процедура може бути лише пойменованою групою операторів. Наприклад:

```
PROCEDURE SW1;
  begin
  r:=x;
  x:=y;
  y:=r;
end;
```

Така процедура не має ні параметрів, ні локальних змінних. У ній всі змінні, що використовуються (в нашому прикладі це **x, y** і **r**), є глобальними, тобто вони мають бути описані в зовнішній (що викликає цю процедуру) програмі. У прикладі процедура здійснює обмін значеннями змінних **x** і **y.** Ясно, що ці змінні мають бути доступними зовнішній програмі, тобто оголошення їх глобальними виправдано. А ось змінна **r** – це робоча змінна, що використовується для тимчасового зберігання даного при виконанні обміну. Для зовнішньої програми вона не становить ніякого інтересу. Тому її можна локалізувати:

```
PROCEDURE SW2
```

```
  var r:real;
begin
  r:=x;
  x:=y;
  y:=r;
end;
```

Тепер зовнішній програмі "невидима" змінна **r**.

Помітимо далі, що отримана процедура "жорстко" прив'язана до глобальних змінних **x** і **y**: до неї можна звернутися тільки для обміну значеннями змінних **x** і **y**. Її не можна застосувати для інших змінних (наприклад, **A** і **B**). Для того, щоб мати таку нагоду, дані в процедуру (і з процедури) потрібно передавати як параметри:

```
PROCEDURE SW3 (var x:real; var y:real);
  var r:real;
begin
  r:=x;
  x:=y;
  y:=r;
end;
```

Тепер цю процедуру можна використати, наприклад, для обміну значеннями змінних **A** та **B** (але вони мають бути описаними як **real**):

```
SW3(A,B);
```

При такому виклику процедури формальні параметри **x** та **y** будуть заміщені відповідними їм фактичними параметрами **A** та **B** і всі дії, закодовані в процедурі будуть виконуватись саме над змінними **A** та **B.**

Будь-яка процедура починається із заголовка. Він складається з ключового слова **PROCEDURE**, за яким йде ідентифікатор (ім'я) процедури, а далі, в круглих дужках, - список формальних параметрів (див. нижче). Потім можуть йти такі ж розділи, що і в основній програмі. На відміну від основної програми процедура завершується не крапкою, а крапкою з комою.

Приклад. Процедура введення N цілих чисел

Хай в зовнішній програмі визначений тип:

```
TYPE MASSIV=ARRAY [1..100] integer;
```

Процедура може мати вигляд:

```
PROCEDYRE INP4(VAR MAS:MASSIV; N:integer);
{заголовок процедури}
  var I:integer;
{локальна змінна - параметр циклу}
begin
  WRITELN ('Введіть',N:3,' цілих чисел');
  for I:=1 to N do READ(MAS[I]);
end;
```

Для виклику процедури в основній програмі або іншій підпрограмі слід записати оператор, що складається з імені процедури, і укладеного в дужки списку фактичних параметрів, які повинні співпадати за кількістю, типом і смислом з формальними параметрами процедури. Наприклад INP4 (M,K);

означає, що викликається вищенаведена процедура INP4 для введення k цілих чисел в масив M. У цьому випадку параметр k – типу **Integer**, а M – масив типу MASSIV, тобто ці змінні описані в зовнішній програмі так:

```
var k:inreger; M:MASSIV;
```

Природно, до звернення до процедури змінній k має бути привласнено певне значення.

3.7.2 Функції

Функція призначена для обчислення якого-небудь значення. У цієї підпрограми дві основні відмінності від процедури.

Перша відмінність функції в її заголовку. Він складається із слова **FUNCTION**, за яким йде ідентифікатор (ім'я) функції і в дужках – список формальних параметрів (див. нижче), потім через двокрапку записується тип функції – тип значення, яке обчислює функція та повертає в зовнішню програму через своє ім’я.

Друга відмінність полягає в тому, що в тілі функції хоча б один раз імені функції має бути привласнено деяке значення.

Приклад. Функція обчислення факторіала числа N

```
FUNCTION FAKTORIAL (N:Integer):Integer;
   var fakt:Integer; I:Integer;
Begin
   fakt:=1;
   for I:=2 to N do fakt:=fakt*I;
   faktorial:=fakt;
end;
```

Виклик функції також відрізняється від виклику процедури. Для виклику функції в основній програмі або іншій підпрограмі слід у виразі, в якому необхідно використати значення функції, вказати ім'я цієї функції із списком фактичних параметрів, які повинні співпадати за кількістю, типом і значенням з формальними параметрами функції. Наприклад:

```
P:=SQR(T)/FAKTORIAL(I);
```

У цьому операторі SQR(T) – виклик стандартної функції піднесення до квадрату з фактичним параметром T; FAKTORIAL(I) – виклик функції обчислення факторіалу з фактичним параметром I, він має бути типу **Integer**.

3.7.3 Формальні і фактичні параметри

Формальні параметри підпрограми указують, з якими параметрами слід звертатися до цієї підпрограми, і використовуються для побудови алгоритму їх обробки в розділі операторів підпрограми як змінні.

При зверненні до підпрограми формальні параметри замінюються на еквівалентні їм фактичні параметри головної (зовнішньої) програми (підпрограми). Типи відповідних фактичних і формальних параметрів мають бути ідентичними.

Усі формальні параметри можна розбити на дві категорії:

- параметри-значення (ці параметри в зовнішній програмі підпрограма не може змінити);

– параметри-змінні (ці параметри підпрограма може змінити в зовнішній програмі).

Параметр-значення передається зовнішньою програмою в підпрограму у вигляді копії свого значення, і тому підпрограмою змінитися не може. Точніше, змінювати його можна, але всі зміни параметра-значення, виконані в підпрограмі, при виході з неї втрачаються і в зовнішньою програму не повертаються, тому, що вони виконуються не над фактичним даним, а над його копією. Параметр-значення указується в заготівці підпрограми своїм ім'ям (ідентифікатором) і через двокрапку – типом. Якщо параметрів-значень одного типу декілька, їх можна перелічити через кому, а потім вже вказати загальний тип. Окремі групи параметрів відокремлюються один від одного крапкою з комою. Як фактичний параметр на місці параметра-значення при виклику підпрограми може виступати будь-який вираз сумісного для привласнення типу, у тому числі і константа. Цей вираз обчислюється, і отримане значення передається підпрограмі.

Параметри-змінні передаються зовнішньою програмою в підпрограму своїми адресами. Отже, підпрограма має безпосередній доступ до цих параметрів і може їх змінювати. Таким чином, при виході з підпрограми нове значення параметра-змінної повертається зовнішній програмі. Параметри-змінні указуються в заголовку підпрограми аналогічно параметрам-значенням, але тільки перед ім'ям параметра записується зарезервоване слово **VAR**. Дія слова **VAR** розповсюджується до найближчої крапки з комою. При звертанні до підпрограми на місці параметра-змінної в якості фактичного параметра повинна використовуватися тільки змінна ідентичного типу. Використовування виразу тут неприпустимо.

Наприклад, в описаній вище процедурі INP4 параметр MAS є параметром-змінною, а параметр N – параметром-значенням. Тому до цієї процедури допустимі такі звернення:

```
INP4(M,20)
INP4(M,k+2)
```

INP4(A,k+SQR(m))

Змінні **k** і **m** повинні мати тип **Integer**, а **M** і **A** - тип **MASSIV**, тобто в основній програмі ці змінні мають бути описані, наприклад, так:

var k,m: integer; M,A: MASSIV;

3.8 Зовнішні модулі

3.8.1 Призначення модулів

Наявність модулів в **Turbo Pascal** дозволяє створювати і налагоджувати програму по частинах, створювати бібліотеки процедур і функцій.

Модуль **(UNIT**) є сукупністю констант, типів, змінних, процедур і функцій, відповідним чином оформлених і записаних на магнітний диск у вигляді окремого файла.

За допомогою ключового слова **USES**, за яким йде ім'я модуля, цей модуль (точніше, його текст) можна зробити доступним в будь-якій програмі або іншому модулі.

Таким чином, модуль фактично є деякою бібліотекою описів (констант, типів, змінних, підпрограм), які можна використовувати в будь-яких призначених для користувача програмах (у тому числі й інших модулях).

3.8.2 Структура модуля

Структура модуля нагадує структуру основної програми, у якої немає тіла. Вірніше, тіло може бути присутнім, але воно служить виключно для цілей ініціалізації модуля. На відміну від основної програми в модулі заголовок обов'язковий, а сам модуль складається з двох частин: інтерфейсу і виконавчої частини.

Заголовок модуля складається з ключового слова **UNIT**, за яким йде унікальний ідентифікатор (ім'я) модуля:

UNIT <ім'я модуля>;

Через **інтерфейс модуля** здійснюється взаємодія основної програми з модулем (або модуля з модулем). В інтерфейсі указуються константи, типи, змінні, процедури і функції, які можуть бути використаними програмою (модулем), що викликала цей модуль. Інтерфейс починається ключовим словом **INTERFACE**. Далі може указуватися ключове слово **USES** та імена модулів, які використовуються даним модулем (необов'язкова частина). Після цього можуть бути: розділи оголошення констант, типів, змінних, розділи оголошення процедур, функцій. В останніх двох розділах оголошуються лише заголовки підпрограм. Самі підпрограми наводяться у виконавчій частині.

Виконавча частина модуля включає всі процедури і функції, що об’явлені в інтерфейсній частині модуля. Вона може включати також локальні мітки, константи, типи, змінні та підпрограми. Виконавча частина починається ключовим словом **IMPLEMENTATION**. Далі після слова **USES** указуються імена модулів, які використовуються підпрограмами даної виконавчої частини (необов'язково). Після цього можуть бути: розділ оголошення міток, розділ оголошення локальних констант, типів і змінних. Потім слідують описи підпрограм модуля звичайної структури (див. п.3.7).

Завершується модуль ключовим словом **END.** (з крапкою). Якщо необхідно включити оператори ініціалізації, вони поміщаються в кінці виконавчої частини і заключаються в операторні дужки **BEGIN** і **END.**

Таким чином, як випливає з вищесказаного, з секції інтерфейсних оголошень описуються програмні елементи (типи, константи, змінні процедури та функції), “видимі” (доступні) для інших програмних модулів, а в секції реалізацій розкривається механізм роботи цих елементів. Тобто, програма одержує доступ тільки до інтерфейсної части, в якій містяться оголошення. Деталі реалізації оголошених процедур, функцій, сховані в секції реалізацій й недоступні іншим модулям.

3.8.3 Використовування модуля

Як вже наголошувалося, описи модуля можна використовувати в будь-якій програмі, заявивши про це за допомогою ключового слова **USES:**

USES <ім'я 1> ,...,<ім'я n> ;

Усі описи констант, типів, змінних, а також усі підпрограми, що містяться в інтерфейсній частині модулів **<ім'я і>,** стають доступними основній програмі так, якби вони були визначені в самій програмі.

3.8.4 Стандартні модулі

У системі **Turbo Pascal** є вісім стандартних модулів. Два з них – **Turbo3** і **Graph3** – призначені для сумісності з більш ранніми версіями **Pascal** (3.0) і практично не використовується. Інші шість забезпечують:

1 **System** – виконання операцій введення-виведення даних, операцій над рядками і числами з плаваючою крапкою, управління динамічною пам'яттю. Цей модуль автоматично додається до будь-якої програми, тобто його ім'я не потрібно указувати в **USES.**

2 **Crt** – управління введенням з клавіатури, виводом на екран дисплея, генератором звуку.

3 **Dos** – виконання деяких, найбільш вживаних, функцій операційної системи.

4 **Overlay** – побудову величезних програм, що не поміщаються повністю в пам'яті.

5 **Graph** – роботу екрану в графічному режимі, виконання графічного виводу.

6 **Printer** – оголошення змінної текстового файла **lst** і зв’язування її з драйвером принтера. Після цього можна виводити інформацію на принтер за допомогою оператора **WRITE** . Наприклад:

WRITE(lst,’ Друкування на принтері’);

Модуль **Graph** знаходиться у файлі **Graph.tpu** . Решта п'ять модулів – у файлі **Turbo.tpl** .

3.8.5 Стандартний модуль Crt

Звичайно екран дисплея використовується в текстовому режимі, в якому екран розглядається як таблиця, що складається з 25 рядків і 80 колонок (можливі і інші розміри таблиці). У кожну позицію цієї таблиці (клітину) можна помістити будь-який з 256 символів, що є в ЕОМ.

Для ідентифікації символів на екрані використовується система координат, початок якої розташований в лівому верхньому кутку, а вісь **У** направлена вниз. Координати вимірюються цілими числами. Відлік починається з 1. Фактично, перша координата – це номер стовпця, а друга – номер рядка. Розглянемо деякі підпрограми стандартного модуля **Crt.**

Функція KeyPressed

Повертає **TRUE**, якщо на клавіатурі натиснута яка-небудь клавіша, інакше – **FALSE**. Символ залишається в буфері. Ця функція не обробляє допоміжні клавіші **Shift, Alt, Ctrl** і т.д. Наприклад, затримку виконання програми, поки не буде натиснута яка-небудь клавіша, можна зробити так:

```
While not keypressed do;
```

Функція ReadKey

Прочитує з клавіатури символ. Повертає значення прийнятого символу. При цьому символ на екран не виводиться. Якщо ніяка клавіша не натиснута, програма чекає, поки це не буде зроблено, тобто чекає введення з клавіатури.

Процедура Window(x1,y1,x2,y2)

Визначає розміри текстового вікна: **x1, y1** – координати лівого верхнього кута; **x2, y2** – нижнього правого.

Текстове вікно – це обмежена ділянка екрану, яка виконує ті ж функції, що і повний екран. Максимальний розмір вікна – весь екран (приймається за умовчанням), тобто (1, 1, 80, 25), мінімальний – два стовпці в одному рядку. Надалі всі координати задаються щодо поточного вікна.

Процедура TextColor(c)

Встановлює колір символів (табл.6). Мерехтіння можна задати, додавши до коду кольору 128 (константа **BLINK**).

Таблиця 6 – Кольори

Темні кольори			**Яскраві кольори**		
Ім'я	*Зн.*	*Призначення*	*Ім'я*	*Зн.*	*Призначення*
Black	0	Чорний	DarkGray	8	Темно-сірий
Blue	1	Синій	LightBlue	9	Світло-синій
Green	2	Зелений	LightGreen	10	Ясно-зелений
Cyan	3	Голубий	LightCyan	11	Ясно-голубий
Red	4	Червоний	LightRed	12	Рожевий
Magenta	5	Фіолетовий	LightMagenta	13	Світло-фіолетовий
Brown	6	Коричневий	Yellow	14	Жовтий
LightGray	7	Світло-сірий	White	15	Білий

Процедура TextBackground(c)

Встановлює колір фону. Можна визначити тільки перші 8 кольорів (**TextColor**, значення **c** від 0 до 7 відповідно до табл.3). Мигання фону неможливе.

Приклади:

```
TextColor(White);          { білий колір символів }
TextColor(15);
TextColor(Red+Blink);    {червоні мерехтливі символи }
TextColor(4+128);
TextColor(132);
TextBackGround(Green); { зелений фон }
TextBackGround(2);
```

Процедура ClrScr

Очищає поточне вікно кольором фону. Усі символи замінюються пропусками.

Процедура GotoXY(x,y)

Переміщає курсор в задану точку (**x,y**) щодо поточного вікна.

Функція WhereX

Повертає поточне значення координати **X** (номер стовпця) курсору щодо поточного вікна.

Функція WhereY

Повертає поточне значення координати **У** (номер рядка) курсору щодо поточного вікна.

Процедура Delay(Ms)

Задає затримку виконання програми на вказане число мілісекунд.

Процедура Sound(Hz)

Запускає внутрішній звуковий пристрій. **Hz** – частота.

Процедура NoSound

Вимикає внутрішній звуковий пристрій.

Приклади

1 Формування вікна з тінню:

```
TextBackground(7);        {світло-сірий фон}
ClrScr;
Window(31,6,51,11);
TextBackground(3);        {голубий фон}
ClrScr;
Window(30,5,50,10);
TextBackground(0);        {чорний (за умовчанням) фон}
ClrScr;
Write( . );               {виведення в створене вікно}
Write( . );
```

2 Видача сигналу про настання якоїсь події:

```
Sound(350);
Delay(200);
Nosound;
Delay(200);
Sound(400);
```

```
Delay(200);
Nosound;
```

4. ЛАБОРАТОРНА РОБОТА 1. НАЙПРОСТІША ПРОГРАМА. ОБЧИСЛЕННЯ ЗНАЧЕННЯ ФУНКЦІЇ, ЗАДАНОЇ УМОВНО

4.1 Теоретичні відомості

1 Оператор привласнювання – це один із самих важливих операторів. Використовується для обчислення значення виразу і привласнення його змінній. Формат оператора:

<ім'я змінної>:=<вираз>;

Зліва стоїть змінна, справа – вираз, значення якого має бути обчислено. Розділяє ці частини складений символ “:=”, який інтерпретується як один елемент.

Приклад:

У:= 5 + SIN(X);

B:= 2 * X/2 + 3 * X - 5;

D:= (A>0) AND (B>0);

C:=’Всього’;

Обчислюється значення виразу, записаного справа, а потім це значення привласнюється змінній, ідентифікатор якої записано зліва.

Змінна і значення виразу повинні належати одному типу. Виключення може складати випадок, коли вираз має значення цілого типу, а змінна – дійсного типу. Так, у вищенаведеному прикладі змінні мають бути описані таким чином:

Var Y,B: Real;

D: Boolean;

C: String[6];

Комп’ютер можна використовувати тільки в тому випадку, коли він має засоби для введення вхідних даних та виведення результатів розрахунків. Причому, результати повинні подаватися у формі, зручній для сприйняття людиною. Для організації вводу та виводу даних через відеотермінал (дисплей) користувача використовуються наступні оператори.

2 Оператор введення – дозволяє ввести з клавіатури значення заданих змінних. Модифікації оператора:

READ(b1,b2 ... ,bn);

READLN(b1,b2 ... ,bn);

де **b1, b2 ..., bn** - список введення, тобто перелік ідентифікаторів змінних, значення яких вводяться.

Виконання програми припиняється і очікується введення з клавіатури рядка даних. В одному рядку одночасно можна набрати значення декількох змінних, відділяючи їх один від одного пропусками (пробілами). Введення рядка завершується при натисканні клавіші <**Enter**>. Типи значень, що вводяться, повинні відповідати типам змінних, перерахованих в списку введення, інакше буде зафіксована помилка.

Якщо у введеному рядку значень менше ніж змінних в списку введення, очікується введення наступного рядка і т.д., поки не будуть заповнені всі змінні із списку введення.

Якщо в рядку значень виявиться більше, ніж змінних в списку введення, то при застосуванні модифікації **READ** незатребувані дані будуть використані в наступному операторі введення. При застосуванні модифікації **READLN** незатребувані дані ігноруються, і здійснюється перехід на новий рядок: наступний оператор введення чекатиме введення нового рядка. Порожній (без списку введення) оператор **READLN** забезпечує перехід до нового рядка.

3 Оператор виведення здійснює виведення інформації на екран дисплея. Модифікації оператора:

WRITE(b1,b2 ..., bn);

WRITELN(b1,b2 ..., bn);

де **b1, b2 ..., bn** – список виведення, тобто вирази, значення яких обчислюються, перетворюються в текстовий вигляд і виводяться на екран. Після кожного виразу через двокрапку може бути заданий шаблон перетворення у вигляді:

bi:m

або:

bi:m:n

Тут **m** означає загальну кількість позицій (символів), що відводяться під значення, **n** – кількість позицій, що відводяться під дробову частину числа (задається тільки для дійсних чисел).

Якщо реальна довжина значення, що виводиться, менше заданої параметром **m**, то значення доповнюється зліва пропусками до заданої довжини **m**. Якщо, навпаки, позицій, що відводяться, недостатньо для розміщення даного (реальне значення, що виводиться, містить більше символів, ніж задано параметром **m** шаблону), поле для виведення автоматично розширяється.

Виведення починається з поточної точки екрану (в якій знаходиться курсор).

Після виконання модифікації **WRITE** курсор залишається за останнім виведеним символом. Тому наступний оператор виведення продовжить виведення в той же рядок. Після виконання модифікації **WRITELN** здійснюється перехід до нового рядка: курсор встановлюється на першу позицію наступного рядка. Порожній (без списку виведення) оператор **WRITELN** виконує тільки переведення курсору в початок нового рядка.

Ці ж оператори часто використовуються і для виведення інформації на принтер. Для цього в операторах виведення перед списком виведення слід вказати ключове слово **lst**, а до програми необхідно приєднати стандартний модуль **PRINTER**:

USES PRINTER;

За допомогою розглянутих конструкцій можна створювати лише прості програми зі строго лінійною структурою. Б

4 Оператор безумовного переходу GOTO використовується для зміни природного порядку виконання операторів і переходу до виконання програми, починаючи із заданого оператора. Оператор, на який здійснюється

перехід, повинен бути помічений міткою. Ця мітка указується в операторі **GOTO**:

GOTO n;

Усі мітки, що використовуються в програмі, мають бути описані в розділі опису міток **LABEL.**

5 Оператор умовного переходу дозволяє змінити порядок виконання операторів у програмі залежно від певних умов, будувати розгалуження в програмі. Загальний вид оператора умовного переходу:

IF <умова> THEN
<оператор 1>
ELSE
<оператор 2>;

Якщо умова, що задана в операторі **IF,** істинна (має значення **TRUE**), то виконується **THEN-гілка** – оператор (простий або складовий), що стоїть після **THEN** (тобто **<оператор 1>**). Інакше (якщо умова не виконується, має значення **FALSE**) виконується **ELSE-гілка** – оператор (він теж може бути складовим), що стоїть після **ELSE** (тобто **<оператор 2>**). Після виконання однієї з гілок оператора робота програми продовжується з оператора, наступного за **IF**. Використовується також усічений формат оператора умовного переходу:

IF <умова> THEN <оператор>;

При його використовуванні **THEN-гілка** виконується при істинній умові, якщо ж умова має значення **FALSE** – оператор ігнорується, і відразу виконується оператор, що стоїть за оператором **IF.**

Слід звернути увагу, що після оператора **<оператор 1>** ставити ознаку кінця оператора (символ крапка з комою) не можна, якщо слідом за ним іде зарезервоване слово **ELSE**, оскільки це не кінець оператора.

Нагадаємо, що якщо в якійсь гілці вимагається виконати декілька (більше одного) операторів, з них необхідно утворити складовий оператор, тобто укласти ці оператори в операторні дужки **BEGIN** і **END.**

6 Оператор вибору варіанта є узагальненням умовного оператора: він дає можливість виконати один з декількох операторів залежно від значення деякого виразу, що називається селектором або ключем вибору. Вираз має бути порядкового типу. У загальному випадку оператор має вигляд

CASE < селектор>
<список вибору 1>: < Оператор 1>;
<список вибору 2>: < Оператор 2>;
⋮<список вибору n>: < Оператор n>;
[ELSE
<оператор>]
END;

<Селектор> – вираз будь-якого порядкового типу, тобто дійсний тип не допустимий.

<список вибору> – список розділених комами констант – значень виразу <селектор> або одного його значення. Можна задавати діапазон. Значення мають бути того ж типу, що й селектор.

<оператор> – будь-який оператор мови, у тому числі і складовий.

Оператор варіанту вибирає для виконання той оператор, одна з констант списку вибору якого співпадає з поточним значенням виразу <селектор>. Якщо значення виразу <селектор> не співпадає ні з однією з міток, тоді виконується оператор, що відповідає **ELSE**. Гілка **ELSE** необов'язкова. По закінченню виконання вибраного оператора виконання оператора вибору завершується і управління передається в кінець оператора **CASE**.

4.2 Приклад виконання лабораторної роботи 1

Приклад 1а. Обчислити дійсні корені квадратного рівняння. $ax^2 + bx + c = 0$

Блок-схема (рис. 4)

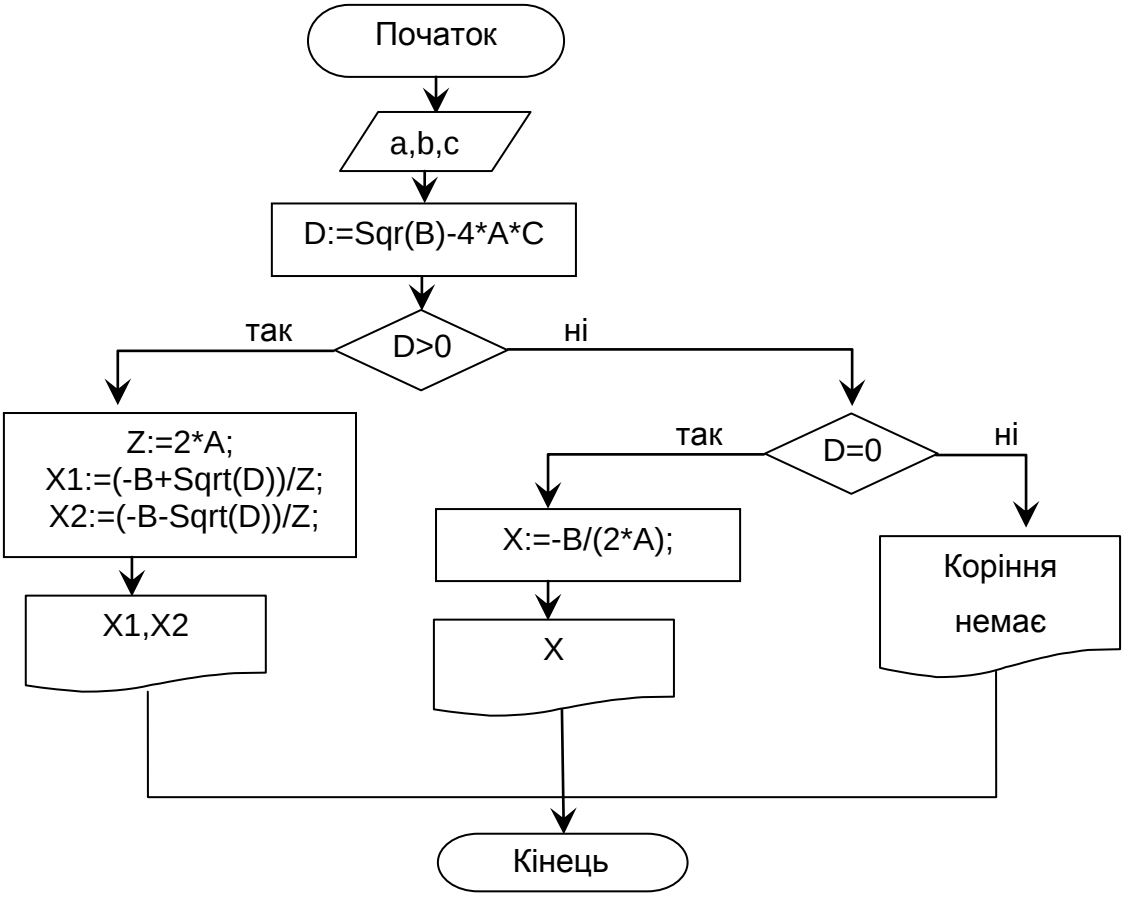


Рисунок 4 – Блок-схема алгоритму

Текст програми

```
Program  Lab1a;
{ Обчислення Коренів Рівняння }
Uses Crt;
Var A,B,C,D,X,X1,X2,Z: Real;
Begin
Clrscr;
Writeln('Введіть коефіцієнти квадратного рівняння');
Write(' A=  '); Readln(A);
Write(' B=  '); Readln(B);
Write(' C=  '); Readln(C);
Writeln('A= ',A:5:2,'  B= ',B:5:2,'  C= ',C:5:2);
D:=Sqr(B)-4*A*C;
If  D>0 Then
Begin
Z:=2*A;
```

```
X1:=(-B+Sqrt(D))/Z;  X2:=(-B-Sqrt(D))/Z;
Writeln(' Рівняння  має  два  корені ');
Writeln(' X1= ',X1:6:2,'  X2= ',X2:6:2)
End
Else If D=0  Then
Begin
X:=-B/(2*A);
Writeln(' Рівняння має один корінь ');
Writeln(' X=',X:6:2)
End
Else   Writeln(' Коріння  Немає ');
Readkey;
End.
```

Приклад 1b. Обчислити значення функції **У** для будь-кого х, с.

$$Y=\begin{cases}\sin x, & x>a\\ \cos x, & x<0\\ tgx, & 0\le x\le a\end{cases}\quad ,\quad a=2c+1$$

Блок-схеми (рис. 5,6)

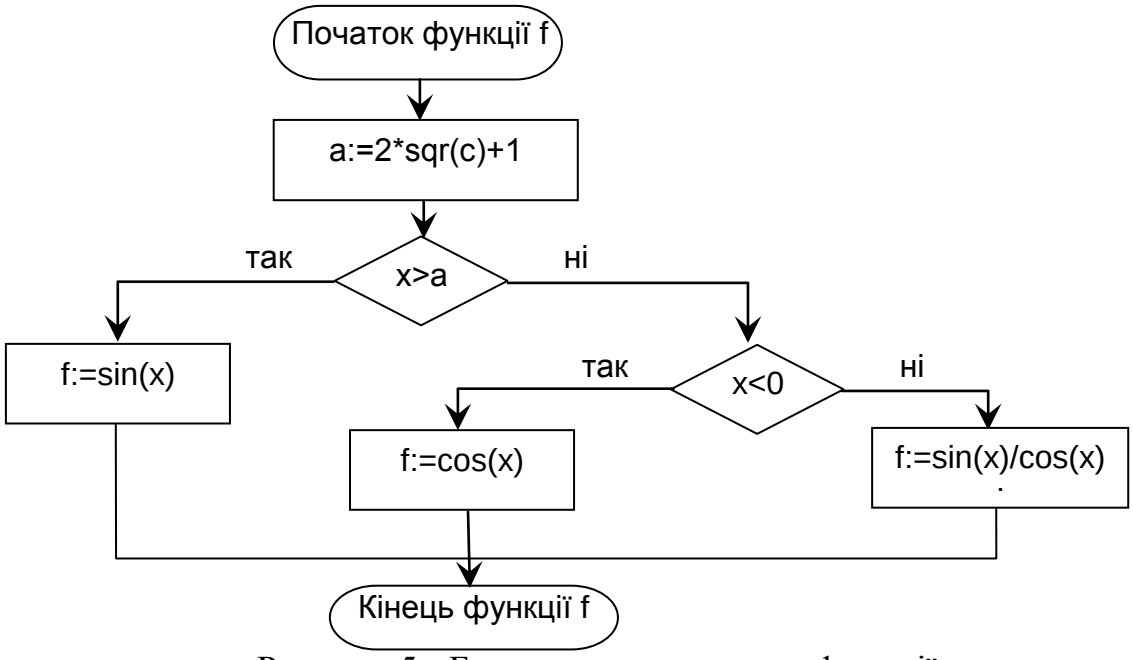


Рисунок 5 - Блок-схема алгоритму функції

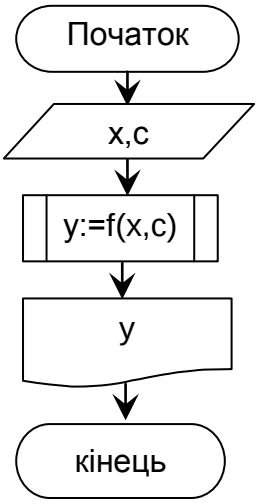


Рисунок 6 - Блок-схема алгоритму основної програми

Текст програми

```
Program lab1b;
{Лабораторна робота 1. Обчислення значення функції}
Uses crt;
var  c,x,y : real;
{-------------обчислення функції---------}
Function f(x,c:real):real;
var a : real;
Begin
a:=2*sqr(c)+1;
if x>a then f:=sin(x)
else if x<0 then f:=cos(x)
else f:=sin(x)/cos(x);
End;
{--------------основна програма-----------}
Begin
clrscr;
writeln(' введіть значення x,c');
write(' x= ');  readln(x);
write(' c= ');  readln(c);
y:=f(x,c);
writeln('якщо x=',x:5:2,' c=',c:5:2,' y=',y:6:2);
readkey;
```

End.

4.3 Варіанти завдання до лабораторної роботи 1

Обчислити значення функції

$$Y=\begin{cases} f1(x), & x<0 \\ f2(x), & 0\le x\le 1 \\ f3(x), & x>1 \end{cases}$$

Види функцій **f1, f2, f3** задані в табл. 7. Значення **x** запрошувати у діалозі.

Таблиця 7 – Варіанти завдання до лабораторної роботи 1

Варіант	**f1(x)**	**f2(x)**	**f3(x)**
1	$tg(x)$	$\sqrt[3]{x}-1$	$\cos(x-2)$
2	$\sin(e^x)$	$\sin(3x^2+x)$	$2\sin(3x^2)/5$
3	$\sqrt[3]{x}-1$	$x^4/7$	$\sin^3(2x)$
4	$\sqrt[3]{\sin^2(x)+\cos^4(x)}$	$\sin(x^2)$	$\ln(2x+5)$
5	$x^3-\ln\lvert x\rvert$	$\ln^3(x+4)$	x^4-x^2-x
6	$\sin(x)^2$	$e^{-X}+\sqrt[4]{x}$	$\ln(x^3+x^2)$
7	$(3x-1)/x^5$	$\ln^2\lvert\sqrt{x+5}\rvert$	$\sqrt{1+x^2}$
8	$\lvert x\rvert^x\cos(x)$	$1/(tg(2x)+1)$	x^2e^{-x}
9	$\lvert x\rvert^2\sin(3x)$	$x^2\cos(x)$	$\sin(x^2)+x$
10	$\lvert x\rvert^2$	$\sin(x^2)$	$\ln^2(x)+\sqrt{x}$

11	$\sin^2(x^3)$	$\ln(x^3+3)$	$2\sin(x)-e^{-x}$
12	$2xe^{-x}$	$\cos(2x)$	$x^x-\cos(x)$
13	$\ln(2x+5)$	$\sin(e^x)$	$tg(1/x)$
14	$\sin(2x+1)$	$(x+1)^2\cos(x^3)$	$\sqrt{x^3-1}+\sin(x)^2$
15	$\cos(3x^2)$	$\sqrt{x^3}\sin(x)$	$x^2+\ln(5x)$
16	$\sin(x^3+5)$	$\ln(4x+1)^2$	$\ln\sqrt[5]{x+x^2}$
17	x^4+2x^3-x	$\sin^2(x^3)$	$x^{x+1}\sin(x)$
18	$x^5ctg(2x^3)$	$\ln(x+1)$	$e^{-2X}-\sqrt[3]{x}$
19	$\sin(4x^3)$	$\sqrt[5]{6x-x^2+1}$	$\mathrm{Sin}(2x+1)^3$
20	$ctg(3x-1)^2$	$2+xe^{-x}$	$\sin^3(x^2)$
21	$x-\sin(x^3+1)$	$(x-1)^3+\cos(2x^3)$	$\sqrt{x^3}\sin(x^3)$
22	$(2x+1)/x^5$	$e^{x+1}+\cos(x)$	$3\ln\sqrt[5]{\sin(x)+x^2}$
23	$3x^5-ctg(x^3)$	$\ln(\sin(4x)+1)^2$	$\ln\sqrt[3]{2x+x^3}$
24	$1{,}3\sqrt{4+x^2}$	3^{x+3}	$5^{x+1}\sin(2x)$
25	e^{-3x}	$\sin^3(x^4)$	$e^{-X}+\sqrt[3]{3x}$

5 ЛАБОРАТОРНА РОБОТА 2. ЦИКЛІЧНИЙ АЛГОРИТМ. ТАБУЛЯЦІЯ ФУНКЦІЇ І ПОШУК ЕКСТРЕМУМІВ

5.1 Теоретичні відомості

Алгоритм циклічної структури **–** це обчислювальний процес, що містить багатократні обчислення за однією і тією ж математичною залежністю, але для різних значень деяких змінних, що до неї входять.

Змінні, що змінюються в циклі, називаються параметрами циклу.

Кожний алгоритм циклічної структури містить такі елементи:

а) підготовка циклу – визначення початкових значень параметрів циклу;

б) тіло циклу – дії, що повторюються багато разів для різних значень параметрів циклу;

в) модифікація циклу – зміна значень параметрів циклу;

г) управління циклом – перевірка умови виходу з циклу.

Оператор циклу з лічильником має дві модифікації наступного формату:

For I := M1 To M2 Do <Оператор>;

For I := M1 To M2 Downto <Оператор>;

де **i** - змінна порядкового типу, яка змінюється при повторенні циклу (параметр циклу, або лічильник);

m1 – вираз, що задає початкове значення лічильника;

m2 – вираз, що задає кінцеве значення лічильника;

DO вказує на те, що зміна параметра циклу здійснюється в прямому порядку: для чергового витка циклу береться наступне значення. Зарезервоване слово **DOWNTO** вказує на зміну в зворотному порядку береться попереднє значення. Найчастіше в якості параметра циклу береться змінна цілого типу. А для цілих типів наступне та попереднє значення

відрізняються на 1. Тому при використанні в якості параметра циклу змінної цілого типу ця змінна змінюється з шагом 1.

При виконанні цього оператора спочатку обчислюється вираз **m1** і одержане значення присвоюється лічильнику (параметру циклу). Потім циклічно виконується наступна послідовність дій:

- Перевірка умови, чи не перевищує поточне значення лічильника кінцевого значення, заданого виразом **m2** (цей вираз обчислюється). Якщо поточне значення перевищує кінцеве значення (становиться меншим у випадку модифікації **DOWNTO**) цикл завершується. Управління одержує оператор, який іде слідом за оператором циклу.
- Виконання тіла циклу (оператора **<Оператор**>). Це може бути складовий оператор, тобто оператор, складений з декількох операторів, заключних в операторні дужки **BEGIN … END.**
- Нарощування змінної циклу (береться наступне або попереднє значення у залежності від напряму його зміни).

Приклад:

```
a:=5;
for  i:= -1  to 1  do
begin
a:=a*i;
writeln(a:3,’   ’, i:2)
end;
```

Оператор циклу з попередньою перевіркою умови (WHILE) має наступний формат:

WHILE <умова> DO <Оператор>;

Доти поки дотримується умова, послідовно виконується тіло циклу (**<оператор>**). Якщо умова не дотримується, то виконання програми продовжується, починаючи з оператора, наступного за циклом.

Якщо в циклі необхідно виконати більше одного оператора, то їх слід укласти в операторні дужки **BEGIN** і **END**, тобто утворити з них складовий оператор.

Безумовно, для коректного виходу із циклу умова виходу повинна змінюватися усередині тіла циклу.

Приклад:

```
while  z > 0  do
begin
y := y + sqr(z);
z := z - 1
end;
```

Оператор циклу з наступною перевіркою умови (REPEAT) має такий формат:

REPEAT <Оператори> UNTIL <умова>;

Спочатку виконується тіло циклу. Потім перевіряється умова (логічний вираз). Якщо вона не виконується (тобто має значення **FALSE**), виконання циклу продовжується, інакше – вихід з циклу, тобто перехід до оператора, наступного за оператором циклу. У будь-якому випадку тіло циклу виконається хоча б один раз.

Слід звернути увагу, що в цьому операторі пара слів **Repeat…Until** грають роль операторних дужок, тому перед **Until** ставити ознаку кінця оператора (символ крапка з комою) необов'язково.

Як і в попередньому операторі для коректного виходу із циклу умова виходу повинна змінюватися усередині тіла циклу.

Приклад:

```
repeat
y := y + sqr(z);
z := z - 1
until z < 0;
```

Для гнучкого керування циклами ***Pascal*** в своєму складі має дві процедури без параметрів:

Break – реалізує негайний вихід з циклу; дія цієї процедури полягає в передачі управління оператору, який стоїть зразу за оператором циклу.

Continue – забезпечує дострокове завершення чергового проходу циклу; еквівалент передачі управління в самий кінець оператора циклу.

Табуляція функцій **–** це формування і виведення (на екран або принтер) таблиці значень функції для значень аргументу (**x**), що змінюється від деякого початкового значення (**xn**) до кінцевого (**xk**) із заданим кроком (**h**). Для цього використовується цикл. При підготовці циклу аргумент спочатку приймає початкове значення (**x:=xn**), в тілі циклу обчислюються і виводяться значення функції для поточного значення аргументу. Модифікація циклу полягає в зміні значення аргументу на величину крока (**x:=x+h**). Цикл завершується, коли після чергової зміни значення аргументу воно перевищить кінцеве значення (**x>xk**).

Пошук екстремумів функції на заданому відрізку методом перебору виконується в циклі, аналогічно табуляції. Замість виведення на екран чи принтер значення функції в кожній точці порівнюється з найбільшим (найменшим) із значень у всіх попередніх точках. Якщо поточне значення більше (менше) найбільшого (найменшого) з попередніх, то його треба вважати новим найбільшим (найменшим) значенням. Інакше найбільше (найменше) значення зберігає старе значення. При підготовці циклу найбільшому (найменшому) значенню привласнюється значення функції в початковій точці (**xn**).

5.2 Приклад виконання лабораторної роботи 2

Обчислити таблицю значень і знайти екстремуми функції

$$Y = \begin{cases} \sqrt{\sin x + 1}, & якщо \quad x > 0 \\ x^2 + 2x + 3, & якщо \quad x \le 0 \end{cases}$$

Блок-схеми (рис. 7,8)

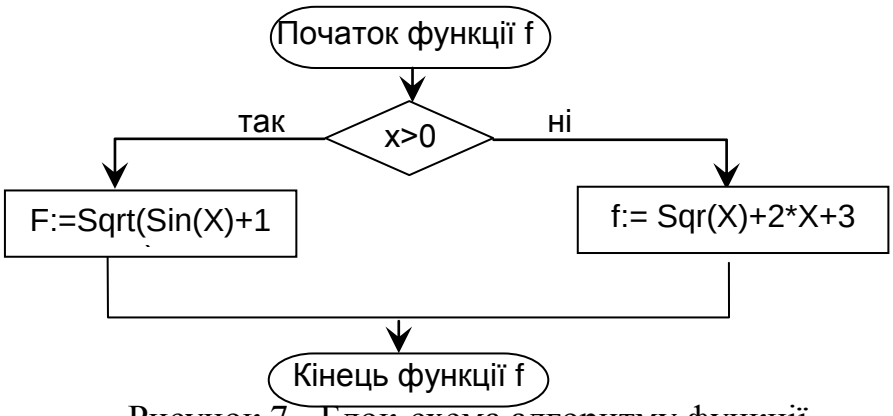


Рисунок 7 - Блок-схема алгоритму функції

Блок-схема алгоритму основної програми

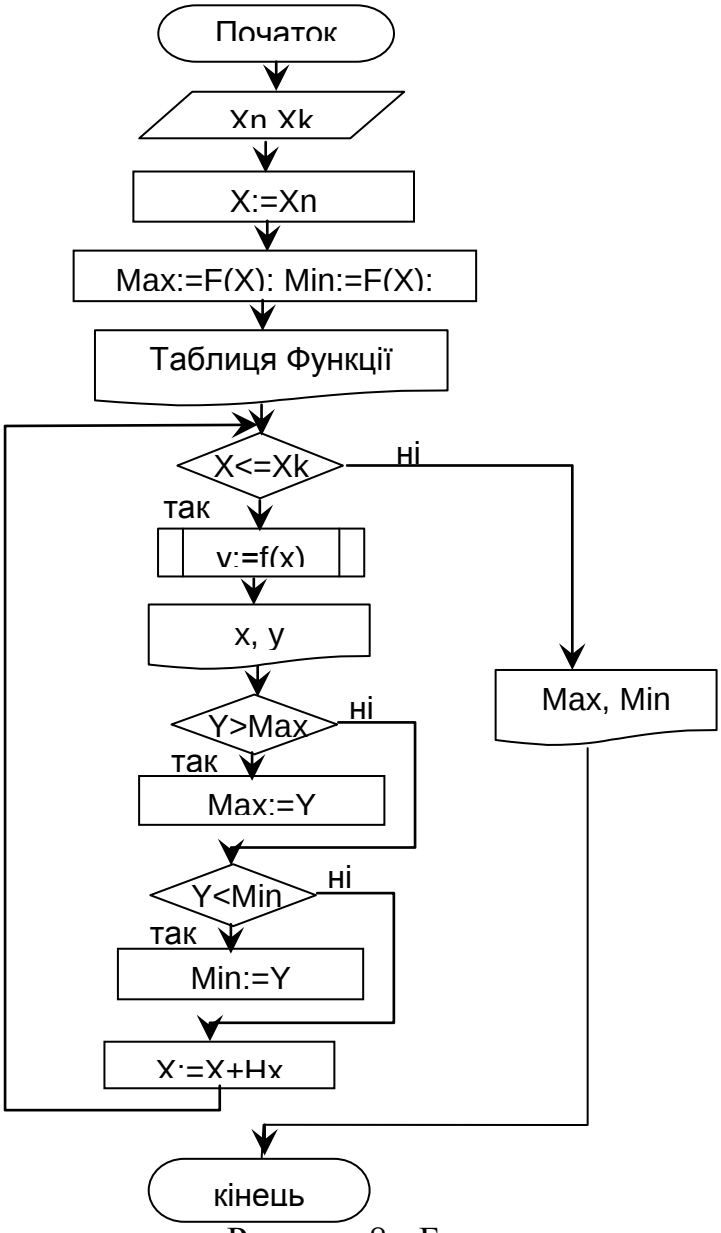


Рисунок 8 - Блок-схема алгоритму основної програми

Текст програми

```
Program Lab2;
{Лабораторна Робота №2.Табуляція І Пошук Екстремумів}
Uses Crt;
Var  Xn,Xk,Hx,X,Y,Max,Min : Real;
{-------------Обчислення Функції---------}
Function F(X:Real):Real;
Begin
If X>0 Then F:=Sqrt(Sin(X)+1)
Else F:=Sqr(X)+2*X+3;
End;
{---------------Основна Програма---------}
Begin
Clrscr;
Write(' Введіть Початкове Значення Відрізка ');
Readln(Xn);
Write(' Введіть Кінцеве Значення Відрізка ');
Readln(Xk);
Write(' Введіть Крок '); Readln(Hx);
X:=Xn;
Max:=F(X);
Min:=F(X);
Writeln(' Таблиця Функції ');
Writeln('****************************');
Writeln('*     X      *      У      *');
Writeln('****************************');
While X<=Xk Do
Begin
У:=F(X);
Writeln('* ',X:10:5,' * ',Y:10:5,' *');
If Y>Max Then Max:=Y;
If Y<Min Then Min:=Y;
X:=X+Hx;
End;
```

```
Writeln('*****************************');
Writeln('  Max= ',Max:10:5,' Min= ',Min:10:5);
Readkey;
End.
```

5.3 Завдання до лабораторної роботи 2

1 Обчислити таблицю значень функції, заданої в табл. 7 (див. лаб.роб. № 1).

2 Методом перебору знайти екстремуми даної функції.

Початкове і кінцеве значення відрізка, а також крок табуляції, запрошувати в діалозі.

6 ЛАБОРАТОРНА РОБОТА №3 СЕЛЕКТИВНА ОБРОБКА МАСИВУ

6.1 Теоретичні відомості

До цих пір ми мали справу з простими типами даних. Базуючись на простих типах, можна будувати більш складніші – структуровані типи даних. Зокрема, масиви.

Масив – це структура, що складається з фіксованої кількості компонентів одного типу. Його можна розглядати як впорядкований набір даних одного типу, що має одне загальне ім'я. Масив характеризується своїм ім'ям і розмірністю.

Загальний вид опису типу «масив»:

TYPE T = ARRAY[T1,T2,...,Tk] OF TC;

де **T -** ім'я типу «масив»;

T1 ,..., Tk - типи індексів;

TC - тип компонентів масмву / базовий тип.

Кількість індексів **k** – розмірність масиву. Індекси можуть бути будь-якого типу, окрім REAL і INTEGER. Звичайно для індексів використовується тип «діапазон».

Кількість індексів **k**, необхідних при зверненні до елемента масиву, визначає к-розмірність масиву. При **k**=1 масив називається **одновимірним.** При k=2 – **двовимірним** і т.д. Одновимірний масив відповідає поняттю вектора (лінійної таблиці, рядка), двовимірний масив – поняттю матриці (набору векторів, прямокутної таблиці).

Після об’явлення типу масиву можна вводити змінні цього типу:

VAR M: T;

де **M** - ідентифікатор масиву.

Масиви можна описувати в розділі описів змінних безпосередньо:

VAR M: ARRAY[T1, T2 ... Tn] OF TC.

Приклад:

VAR M: ARRAY[1..8] OF REAL;

Тут масив з ім'ям **M** складається з восьми елементів (M1, M2 ..., M8) дійсного типу.

Доступ до будь-якого елемента масиву здійснюється вказівкою ідентифікатора масиву, за яким в квадратних дужках йдуть індексні вирази. Індексний вираз повинен давати значення, що лежать в діапазоні, визначуваному типом індексу. Для оголошеного вище масиву **M** у програмі доступні наступні індексні змінні: **M[1], M[2] ..., M[8].**

Знаходження найбільшого і якнайменшого елементів масиву методом перебору виконується в циклі, в якому поточний елемент масиву порівнюється з найбільшим (найменшим) зі всіх попередніх елементів масиву. Якщо поточний елемент виявиться більше (менше) найбільшого (найменшого) з попередніх, то його треба вважати новим найбільшим (найменшим). Інакше найбільше (найменше) зберігає старе значення, тобто:

$$Y_{\max} = \begin{cases} Y_i, & якщо\ Y_i > Y_{\max} \\ Y_{\max}, & якщо\ Y_i \le Y_{\max} \end{cases}$$

$$Y_{\min} = \begin{cases} Y_i, & якщо\ Y_i < Y_{\min} \\ Y_{\min}, & якщо\ Y_i \ge Y_{\min} \end{cases}$$

Перед виконанням циклу найбільшому (найменшому) значенню необхідно привласнити яке-небудь початкове (стартове) значення.

Сортування масиву полягає в розташуванні елементів масиву у порядку, що вимагається.

Розглянемо масив **a(1), a(2), ..., a(к)** . Хай вимагається переставити елементи цього масиву так, щоб після перестановки вони були впорядковані за збільшенням: **a(1)**≤ **a(2)**≤ **...**≤**a(n).** Ця задача називається задачею сортування масиву. Для вирішення цієї задачі можна скористатися, наприклад, наступними найпростішими алгоритмами:

1 Знайти елемент масиву, який має якнайменше значення, переставити його з першим елементом. Виконати те ж саме, почавши з другого елемента, і т.д. (сортування вибором).

2 Послідовно проглянути елементи **a(1), ..., a(n-1)** і кожний елемент, що проглядається, **a(i)** порівняти з наступним за ним **a(i +1).** Якщо **a(i) > a(i + 1)** – ці елементи поміняти місцями. Тим самим найбільше число пересунеться на останнє місце. Наступні перегляди починати знову спочатку, зменшуючи на одиницю кількість елементів, що проглядаються (сортування обміном або методом *пухирця)*.

3 Переглядати послідовно **a(2), a(3) ..., a(n)** і кожний новий елемент **a(i)** вставляти на відповідне місце у вже впорядковану сукупність **a(1), a(2) ..., a(i-1).** Це місце визначається послідовним порівнянням a(i) з впорядкованими елементами **a(1), a(2) ..., a(i-1)** (сортування вставками).

Ми розглянули сортування за збільшенням. Сортування за убуванням здійснюється аналогічно.

Селективна обробка масиву – це виділення з масиву елементів, що задовольняють умові, і обробка виділених фрагментів. Часто з виділених елементів формують новий (робочий) масив і потім його обробляють.

Найбільш часто зустрічаються такі умови обробки елементів масиву:

парні елементи A[i] mod 2 = 0
непарні елементи A[i] mod 2 <> 0
кратні k A[i] mod k = 0
некратні k A[i] mod k <> 0
на парних місцях i mod 2 = 0
на непарних місцях i mod 2 <> 0
додатні елементи A[i] > 0
від’ємні елементи A[i] >= 0
в інтервалі (x1,x2) (A[i]>x1) and (A[i]<x2)

6.2 Приклади виконання лабораторної роботи 3

Приклад 3а. Знайти середнє арифметичне додатних парних елементів кожного з масивів **A**[10] і **B**[15].

Блок-схеми (рис. 9,10,11,12)

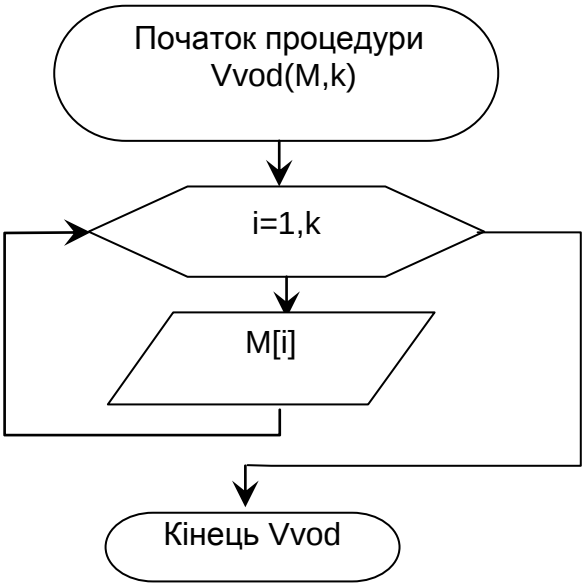


Рисунок 9 - Блок-схема алгоритму процедури Vvod

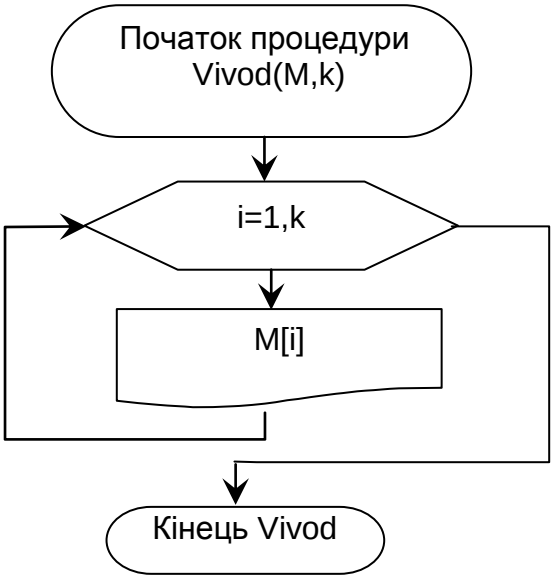


Рисунок 10 - Блок-схема алгоритму процедури Vivod

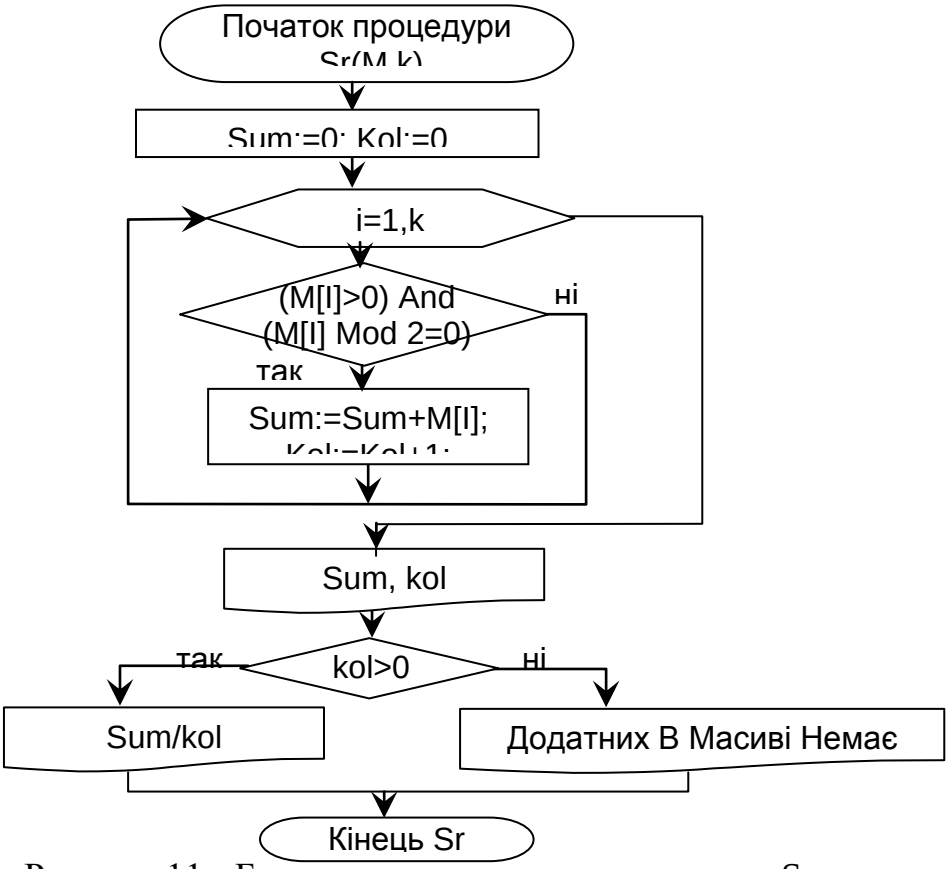


Рисунок 11 - Блок-схема алгоритму процедури Sr

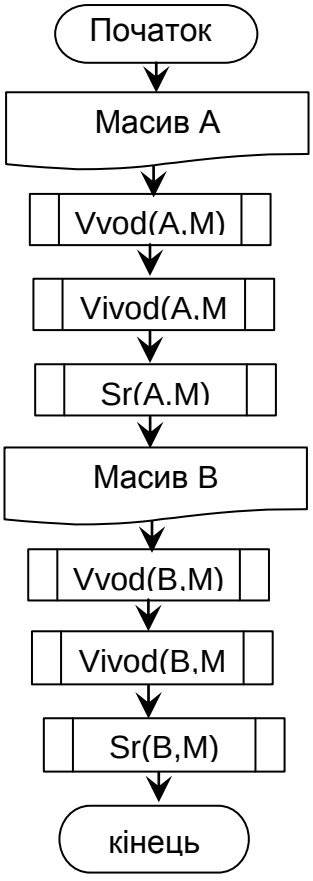


Рисунок 12 - Блок-схема алгоритму основної програми

Текст програми

```
Program Lab3a;
{Селективна Обробка Масивів }
Uses Crt;
Const M=10;  N=15;
Type  Mas=Array[1..N] Integer;
Var  A, B : Mas;  I : Integer;
{-------Процедура Введення Масиву-------}
Procedure Vvod(Var M:Mas;K:Integer);
Begin
Writeln(' Введіть ',K:2,' Чисел Масиву ');
For I:=1 To K Do Read(M[I]);
Writeln;
End;
{-------Процедура Виведення Масиву-------}
Procedure Vivod(Var M:Mas;K:Integer);
Begin
Writeln(' Початковий Масив ');
For I:=1 To K Do Write(M[I]:4);
Writeln;
End;
{-------Процедура Обчислення Серед. Аріф.---}
Procedure Sr(M:Mas;K:Integer);
Var Sum, Kol : Integer;
Begin
Sum:=0; Kol:=0;
For I:=1 To K Do
If (M[I]>0) And (M[I] Mod 2=0) Then
Begin
Sum:=Sum+M[I];
Kol:=Kol+1;
End;
Writeln(' Сума Додатних Парних= ',Sum:4);
Writeln(' Кількість Додатних Парних = ',Kol:2);
If Kol>0 Then
Writeln(' Середнє Арифмет. = ',Sum/Kol:7:3)
```

```
Else
Writeln(' Додатних В Масиві Немає ');
End;
{-------Основна Програма-------}
Begin
Clrscr;
Writeln(' Масив A ');
Vvod(A,M);
Vivod(A,M);
Sr(A,M);
Writeln(' Масив B ');
Vvod(B,N);
Vivod(B,N);
Sr(B,N);
Readkey;
End.
```

Приклад 3b. У кожному з масивів **A** і **C** поміняти місцями перший і якнайменший додатний елементи. Розмірність масивів довільна.

Блок-схеми (рис. 9,10,13,14)

Блок-схема алгоритму процедур Vvod та Vivod такі ж як у прикладі 3а.

Текст програми

```
Program Lab3b;
{Заміна Першого І Найменьшого Додатного }
Uses Crt;
Const V=50;
Type Mas=Array[1..V] Real;
Var A, C: Mas; N,M,I: Integer;
{-------Процедура Введення Масиву-------}
Procedure Vvod(Var X:Mas;Var K:Integer);
  Var K1:Integer;
Begin
Writeln(' Введіть Кількість Ел-Тів Масиву');
Readln(K1);
Writeln(' Введіть ',K1:2,' Чисел Масиву');
For I:=1 To K1 Do Read(X[I]);
```

```
Writeln;
K:=K1;
End;
{-------Процедура Виведення Масиву-------}
Procedure Vivod(X:Mas;K:Integer);
Begin
For I:=1 To K Do Write(X[I]:5:2);
Writeln;
End;
{-------Процедура Обробки Масиву-------}
Procedure Obrab(Var X:Mas;K:Integer);
Var Min:Real; Imin:Integer;
Begin
Imin:=0;
For I:=1 To K Do
If X[I]>0 Then
Begin
Min:=X[I];
Imin:=I;
End;
For I:=1 To K Do
If (X[I]>0) And (X[I]<Min) Then
Begin
Min:=X[I];
Imin:=I;
End;
If Imin>0 Then
Begin
Writeln(' Найменьший = ',Min:5:2);
Writeln(' Його Номер = ',Imin:3);
If Imin>1 Then
Begin
X[Imin]:=X[1];
X[1]:=Min;
End
```

```
Else
Writeln(' Перший Елемент Є Найменьшим ')
End
Else
Writeln(' В Масиві Немає Додатних ');
End;
{-------Основна Програма-------}
Begin
Clrscr;
Vvod(A,M);
Vvod(C,N);
Clrscr;
Writeln(' Масив А ');
Vivod(A,M);
Obrab(A,M);
Writeln(' Перетворений Масив А ');
Vivod(A,M);
Writeln(' Масив С ');
Vivod(C,N);
Obrab(C,N);
Writeln(' Перетворений Масив С ');
Vivod(C,N);
Readkey;
End.
```

Приклад 3с. Обчислити суму двох найбільших і двох якнайменших непарних елементів масиву. Розмірність масиву довільна.

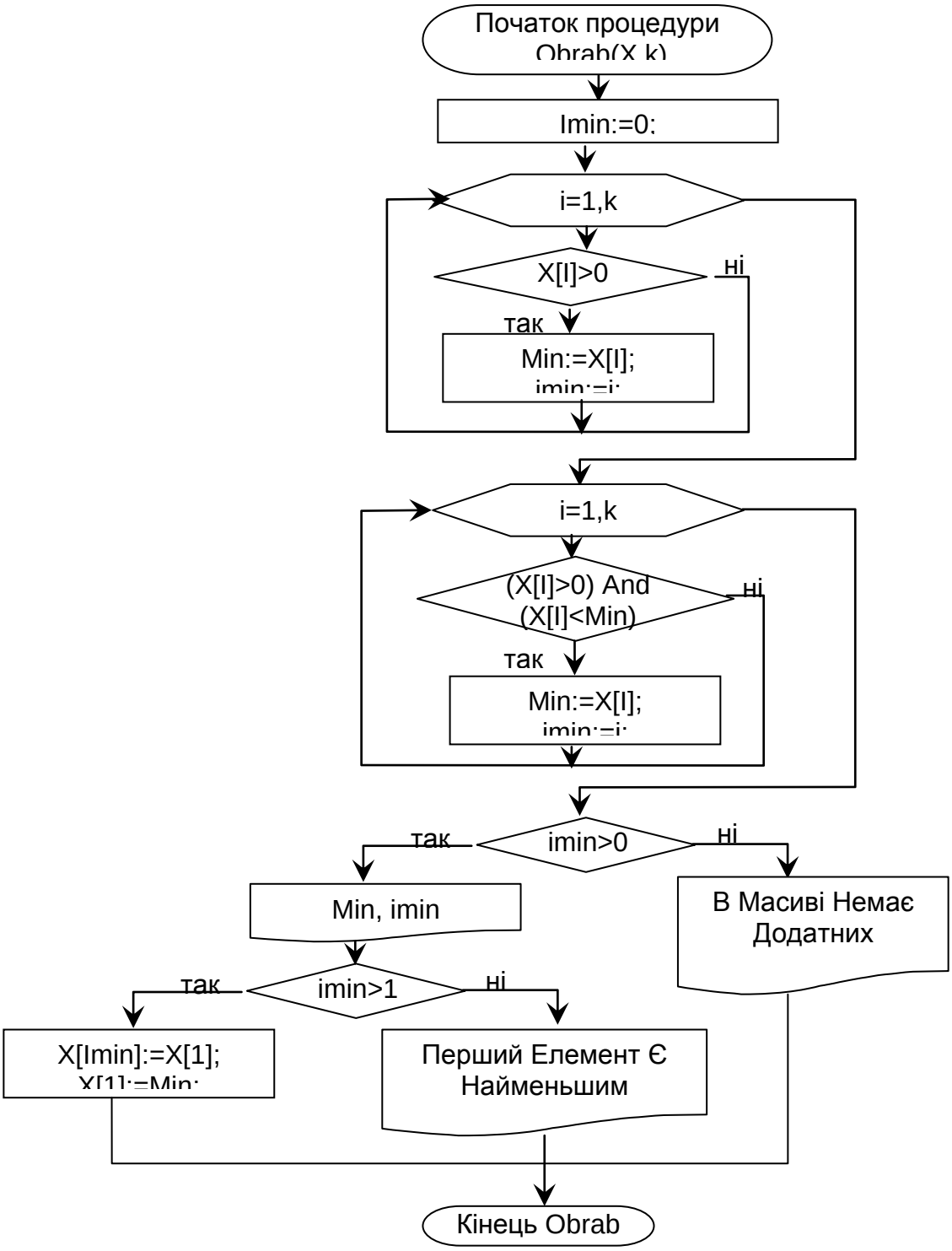


Рисунок 13 - Блок-схема алгоритму процедури Obrab

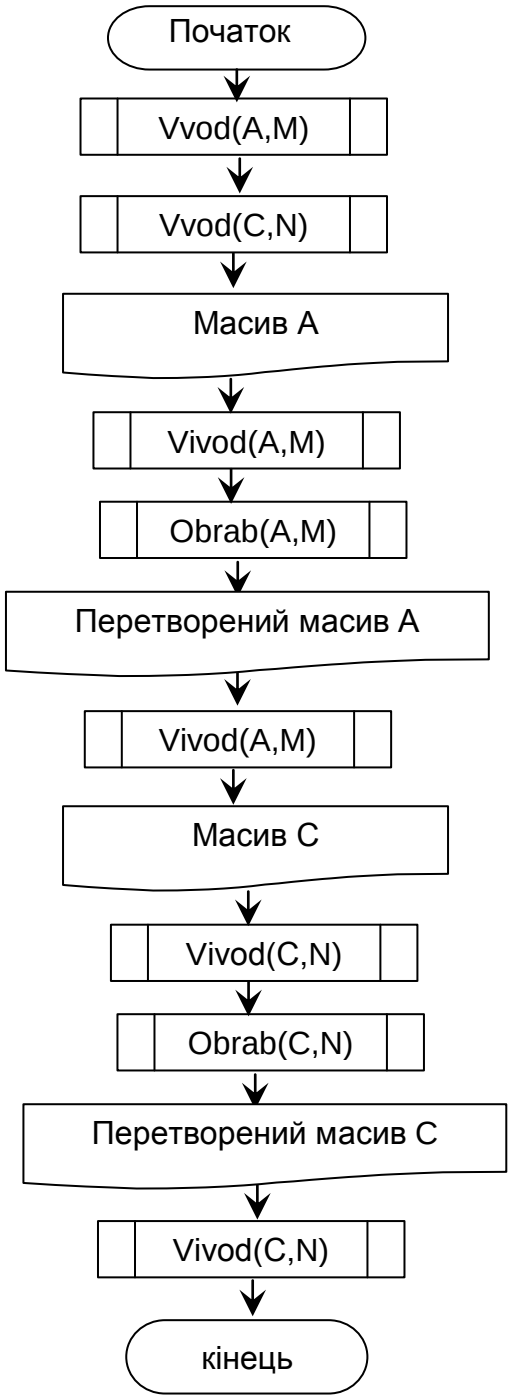


Рисунок 14 - Блок-схема алгоритму основної програми

Блок-схеми (рис. 15, 16, 17, 18, 19)

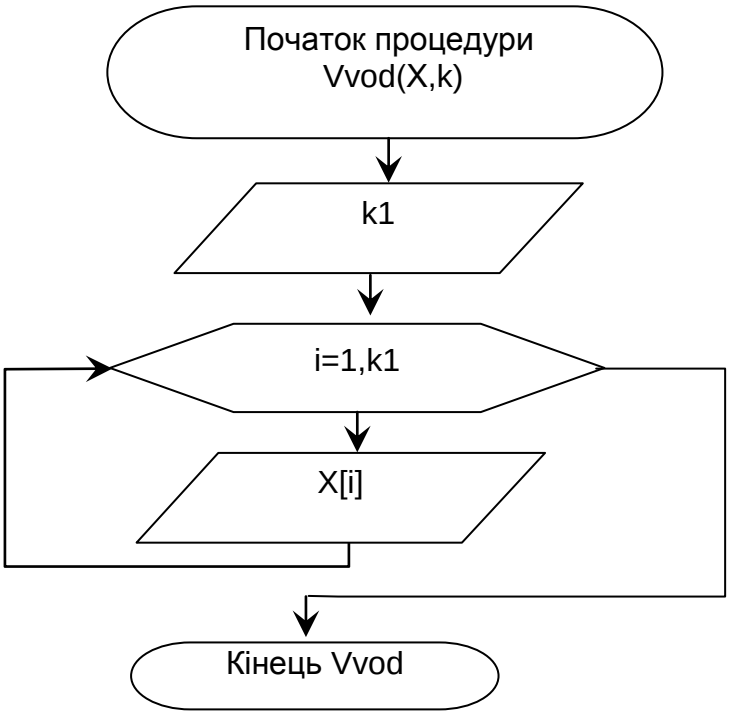


Рисунок 15 - Блок-схема алгоритму процедури Vvod

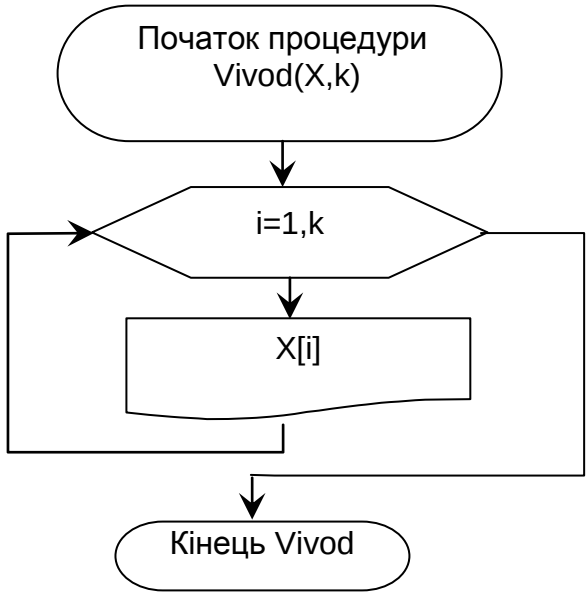


Рисунок 16 - Блок-схема алгоритму процедури Vivod

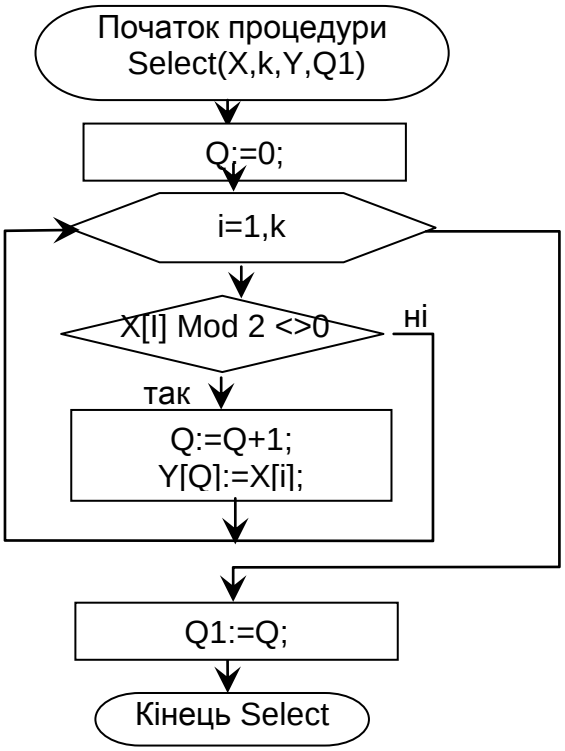


Рисунок 17 - Блок-схема алгоритму процедури Select

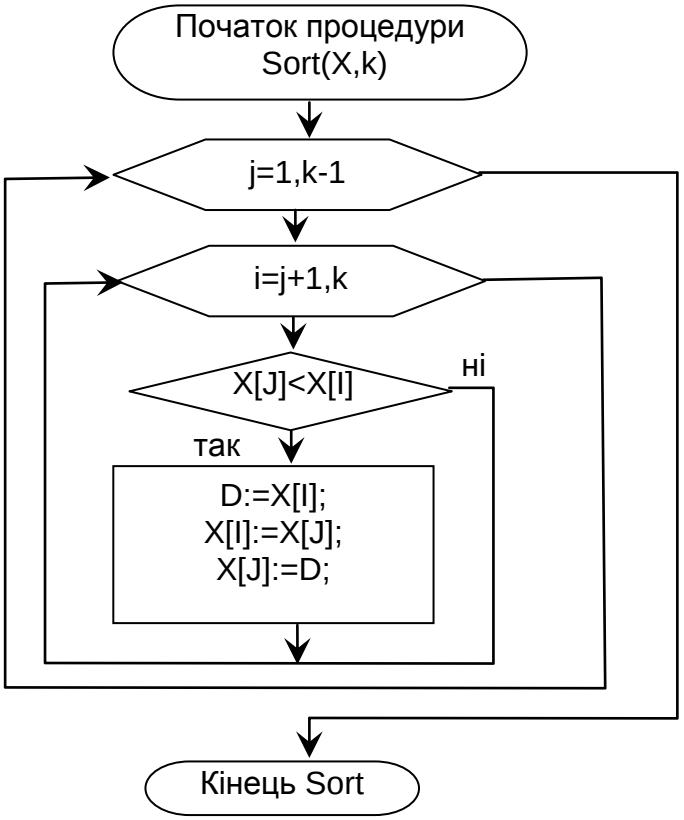


Рисунок 18 - Блок-схема алгоритму процедури Sort

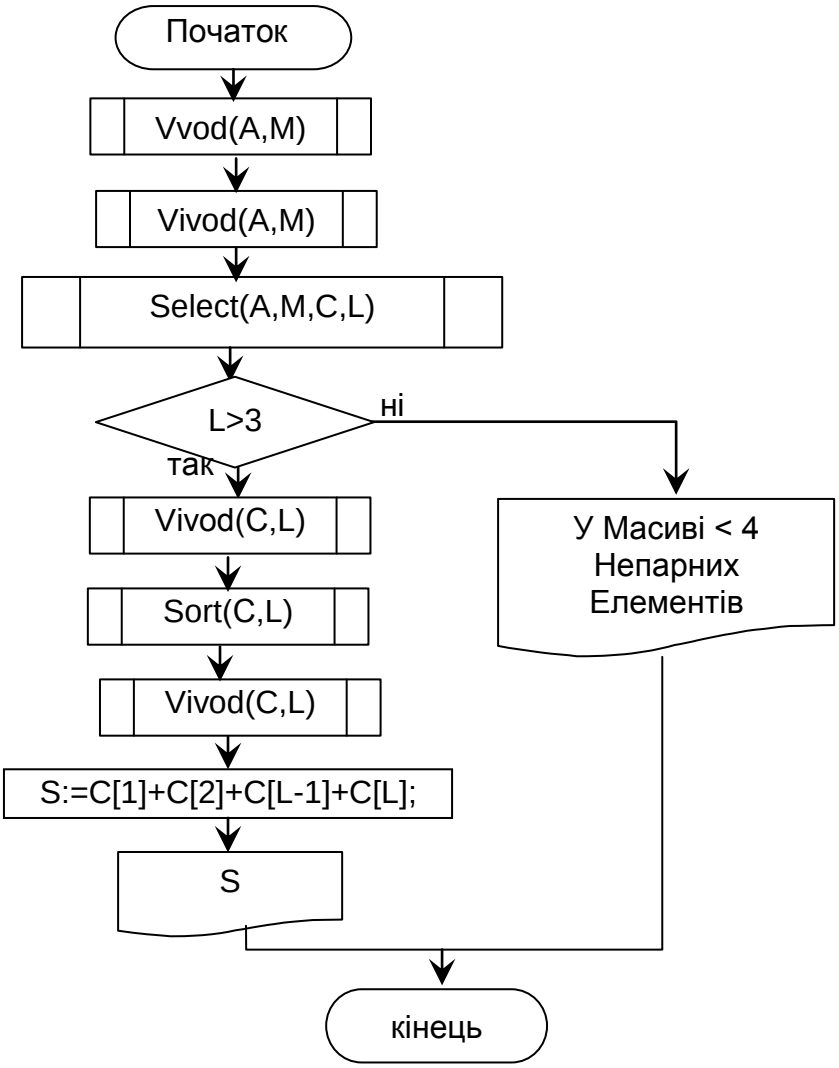


Рисунок 19 - Блок-схема алгоритму основної програми

Текст програми

```
Program Lab3c;
{Сума Двох Найбільших І Двох Найменьших Непарних}
Uses Crt;
Const V=50;
Type Mas=Array[1..V] Integer;
Var A, C: Mas;
Var S,L,M,I: Integer;
{-------Процедура Введення Масиву-------}
Procedure Vvod(Var X:Mas;Var K:Integer);
Var K1:Integer;
Begin
```

```
Writeln(' Введіть Кількість Ел-Тів Масиву ');
Readln(K1);
Writeln(' Введіть ',K1:2,' Чисел Масиву');
For I:=1 To K1 Do Read(X[I]);
Writeln;
K:=K1;
End;
{-------Процедура Виведення Масиву-------}
Procedure Vivod(X:Mas;K:Integer);
Begin
For I:=1 To K Do Write(X[I]:4);
Writeln;
End;
{-------Процедура Вибору-------}
Procedure Select(X:Mas;K:Integer;Var У:Mas;
Var Q1:Integer);
Var Q:Integer;
Begin
Q:=0;
For I:=1 To K Do
If X[I] Mod 2 <>0  Then
Begin
Q:=Q+1;
У[Q]:=X[I];
End;
Q1:=Q;
End;
{---------Процедура Сортування-------}
Procedure Sort(Var X:Mas;K:Integer);
Var J,D: Integer;
Begin
For J:=1 To K-1 Do
For I:=J+1 To K Do
If X[J]<X[I] Then
Begin
```

```
D:=X[I];
X[I]:=X[J];
X[J]:=D;
End;
End;
{-------Основна Програма-------}
Begin
Clrscr;
Vvod(A,M);
Writeln(' Початковий Масив ');
Vivod(A,M);
Select(A,M,C,L);
If L>3 Then
Begin
Writeln('Робочий Масив');
Vivod(C,L);
Sort(C,L);
Writeln('Відсортированый Массив');
Vivod(C,L);
S:=C[1]+C[2]+C[L-1]+C[L];
Writeln('Сума = ',S:3);
End
Else
Writeln(' В Масиві < 4 Непарних Елементів ');
Readkey;
End.
```

6.3 Варіанти завдання до лабораторної роботи 3

1 Скласти програму для вирішення наступних задач. Варіанти задач наведені в табл.8.

Таблиця 8 - Варіанти завдання

Варіант	**Зміст задачі**
1	Знайти суму додатних і кількість непарних елементів в масивах A(10) і M(12)
2	Обчислити середнє геометричне елементів в масивах T(15) і A(12)
3	Обчислити середнє арифметичне парних і суму непарних елементів в масивах C(10) і F(10)
4	Знайти кількість елементів в масивах O(16) і M(12), кратних 4 і не більших заданого числа A
5	Обчислити середнє геометричне парних елементів масивів C(10) і M(10), не менших за 10
6	Знайти суму і кількість парних елементів в масивах P(13) і Z(8), що стоять на непарних місцях
7	Знайти кількість елементів, кратних 5, що належить до інтервалу [10;50] в масивах F(14) і M(10)
8	Знайти середнє геометричне парних елементів в масивах C(12) і A(8), має парний індекс
9	Знайти різницю суми додатних і добуток від'ємних чисел в масивах A(14) і C(9)
10	Знайти модуль різниці добутку парних і суми непарних елементів в масивах H(16) і A(8)
11	Знайти суму і кількість елементів в масивах C(10) і A(12), у яких індекс кратний 3
12	Обчислити суму тих елементів масивів A(10) і B(12), числові значення яких належать напівінтервалу [i; i+1]
13	Знайти добуток і кількість негативних елементів в масивах B(12) і A(10), має парний індекс
14	Обчислити суму і кількість елементів масивів T(12) і M(12), які мають непарний індекс
15	Знайти середнє арифметичне негативних елементів масивів B(12) і C(10), що стоїть на парних місцях
16	Знайти добуток кількості парних і кількості непарних елементів в масивах H(16) і A(12)

Продовження табл.8

Варіант	Зміст задачі
17	Обчислити добуток і кількість непарних елементів масивів P(15) і M(9)
18	Обчислити середнє геометричне елементів масивів K(10) і C(8), не кратних 3
19	Знайти середнє арифметичне парних елементів в масивах A(15) і C(9) з інтервалу [10;30]
20	Знайти суму і кількість додатних елементів в масивах B(13) і A(9), що стоїть на парних місцях
21	Знайти добуток суми парних і суми непарних елементів масивів C(10) і A(9)
22	Знайти середнє геометричне додатних елементів в масивах M(16) і A(8), що стоїть на непарних місцях
23	Знайти різницю між добутком парних чисел і сумою додатних чисел масивів B(12) і A(7)
24	Обчислити суму квадратів додатних елементів масиву P(16) і B(7), які мають непарні індекси
25	Підрахувати кількості додатних і парних чисел в масивах A(13) і B(9)

2 Для кожного з двох масивів A(10) і B(12) скласти програму для вирішення наступних задач. Варіанти задач наведені в табл.9.

Таблиця 9 - Варіанти завдання

Варіант	Зміст задачі
1	Знайти номер найбільшого додатного елемента
2	Знайти суму найбільшого і якнайменшого елементів
3	Знайти добуток номера найбільшого додатного і якнайменшого від’ємного елементів
4	Знайти номер і значення якнайменшого додатного непарного елемента

Продовження табл.9

Варіант	Зміст задачі
5	Знайти різницю максимального і мінімального додатних парних елементів
6	Знайти суму квадратів максимального і мінімального елементів
7	Знайти максимальний і мінімальний елементи і поміняти їх місцями
8	Знайти частку від ділення найбільшого на якнайменший елементів
9	Поміняти місцями перший і максимальний елемент
10	Знайти максимальний по модулю елемент і його номер
11	Знайти різницю мінімального парного і мінімального непарного елементів
12	Знайти суму мінімального додатного елемента і його номера
13	Поміняти місцями максимальний додатний і мінімальний від'ємний елементи
14	Знайти добуток модулів найбільшого від'ємного елемента і якнайменшого парного елемента
15	Поміняти місцями другий і найбільший додатний елементи
16	Записати +1 замість максимального парного елемента, а число -1 замість мінімального непарного елемента
17	Знайти номер і значення найбільшого по модулю елемента
18	Записати 100 замість максимального кратного 3 елемента
19	Поміняти місцями максимальний парний і мінімальний непарний елементи
20	Знайти номер і значення найбільшого кратного 5 елемента
21	Записати -100 замість мінімального парного елемента
22	Знайти мінімальний кратний 4 елемент і його номер
23	Обчислити середнє арифметичне максимального і мінімального елементів
24	Обчислити середнє геометричне номерів максимального і мінімального елементів
25	Обчислити частку від ділення максимального елемента на його порядковий номер

3 В масивах M(13) і B(9) знайти (A) чисел, умови A наведені в табл.10. Масив довільний.

Таблиця 10 - Варіанти завдання

Варіант	**Умова A**
1	Суму трьох найбільших додатних
2	Суму трьох якнайменших від'ємних , кратних 3
3	Різниця двох найбільших
4	Частку від ділення двох найбільших, кратних 3
5	Добуток найбільшого і якнайменшого парних
6	Суму двох найбільших від'ємних
7	Модуль різниці двох найбільших
8	Добуток трьох найбільших додатних
9	Добуток трьох якнайменших від'ємних
10	Суму двох найбільших і трьох якнайменших парних
11	Суму чотирьох найбільших, кратних 5
12	Суму чотирьох якнайменших парних від'ємних
13	Добуток найбільшого і якнайменшого від'ємних , кратних 3
14	Суму найбільшого і двох якнайменших парних
15	Добуток чотирьох найбільших додатних, кратних 5
16	Частку від ділення суми двох найбільших позитивних і двох найбільших від'ємних
17	Суму двох найбільших парних і двох якнайменших непарних
18	Добуток трьох найбільших додатних і двох найбільших від'ємних
19	Суму трьох якнайменших, що стоять на парних місцях
20	Суму двох найбільших і двох якнайменших, що стоять на непарних місцях
21	Добуток найбільшого додатного і двох якнайменших, що стоять на парних місцях
22	Добуток п'яти найбільших додатних парних
23	Суму трьох якнайменших від'ємних, кратних 5
24	Суму найбільшого і якнайменшого від'ємних
25	Частку від ділення суми двох найбільших додатних і суми двох якнайменших від'ємних

7 ЛАБОРАТОРНА РОБОТА 4. ВКЛАДЕНІ ЦИКЛИ. ОБРОБКА ДВОМІРНИХ МАСИВІВ

7.1 Теоретичні відомості

У **Pascal** багатовимірні масиви можуть визначатися послідовно: спочатку оголошується один масив, потім другий, елементами якого є оголошені раніше масиви, і т.д. Один масив вкладається в інший, і ступінь такого вкладення необмежена. Наприклад, для матриці A(2,3)

```
TYPE STROKA = ARRAY [1..3] OF REAL;
MATR = ARRAY [1..2] OF STROKA;
VAR  V : STROKA; A : MATR;
```

Змінна **A** – це двомірний масив з двох рядків, до кожного з яких включено по три елементи.

Опис **A** можна скоротити, виключивши визначення типу STROKA у визначенні типу MATR:

```
TYPE MATR = ARRAY [1..2] OF ARRAY [1..3] OF REAL;
```

або ж

```
TYPE  MATR = ARRAY [1..2,1..3] OF REAL;
```

Якщо вказаний тип використовується для визначення одного масиву в програмі, то зручно оголошувати масив у розділі опису змінних:

```
VAR  A : ARRAY [1..2,1..3] OF REAL;
```

Як вже наголошувалося, поняття двомірного масиву відповідає поняттю матриці або таблиці. При цьому перший індекс відповідає рядку, а другий – стовпцю. Максимальна кількість рядків (**m**) і стовпців (**n**) визначається при описі типу масиву. Посилання на елемент матриці **A**, що знаходиться на перетині **i**-го рядка і **j**-го стовпця, має вигляд **A[i,j].**

При обробці матриць часто доводиться виділяти елементи:

- k-го рядка A[к,j] j=1, ..., m
- k-ї колонки A[i,k] i=1, ... , n

а для квадратних матриць (m=n) також:

головної діагоналі	A[i,i]	i=1, ..., n
побічної діагоналі	A[i,n+1-i]	i=1, ..., n
наддіагональні	A[i,j]	i>j
піддіагональні	A[i,j]	i<j

Для двомірних масивів зручно використовувати вкладені цикли – цикли, в тілі яких містяться інші циклічні структури. Цикл, що містить в собі інший цикл, називається зовнішній, а цикл, що міститься в іншому – внутрішнім.

При виконанні вкладених циклів спочатку повністю виконується внутрішній цикл, потім зовнішній цикл змінює значення свого параметра циклу на наступне і знову виконується внутрішній цикл і т.д. до повного виконання зовнішнього циклу.

7.2 Приклад виконання лабораторної роботи 4

Приклад 4а. Знайти мінімальний додатний елемент матриць A(5,4) і B(4,3), а також номер рядка і стовпця, де він знаходиться.

Блок- схеми (рис. 20, 21, 22, 23)

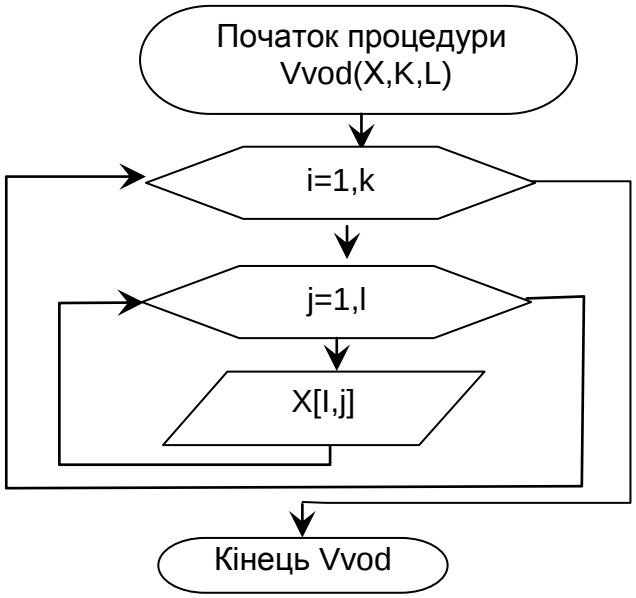


Рисунок 20 - Блок-схема алгоритму процедури Vvod

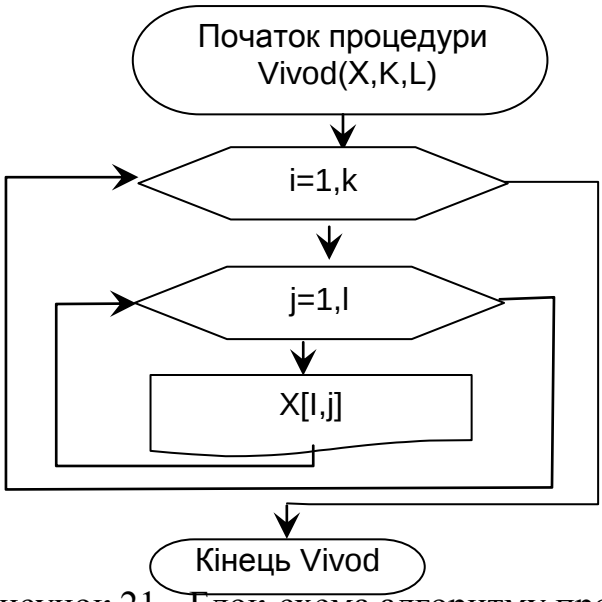


Рисунок 21 - Блок-схема алгоритму процедури Vivod

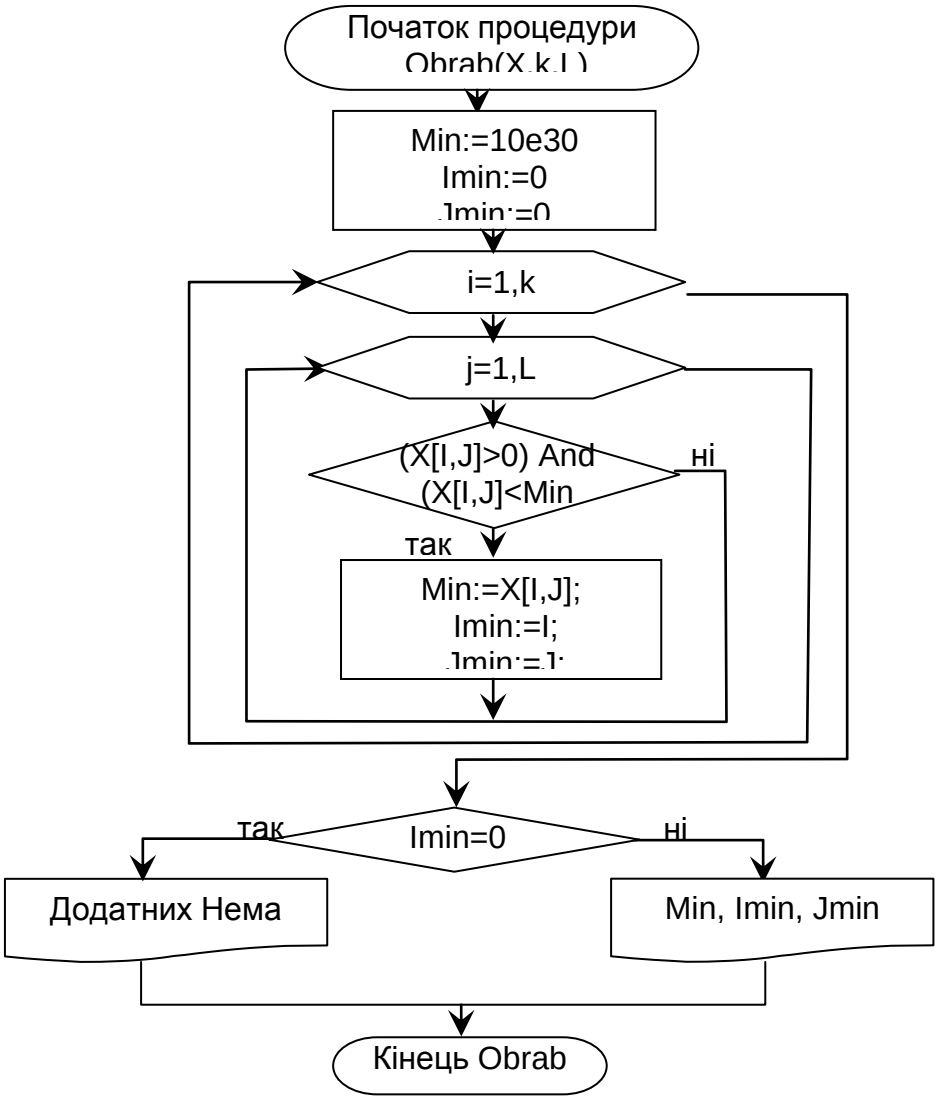


Рисунок 22 - Блок-схема алгоритму процедури Obrab

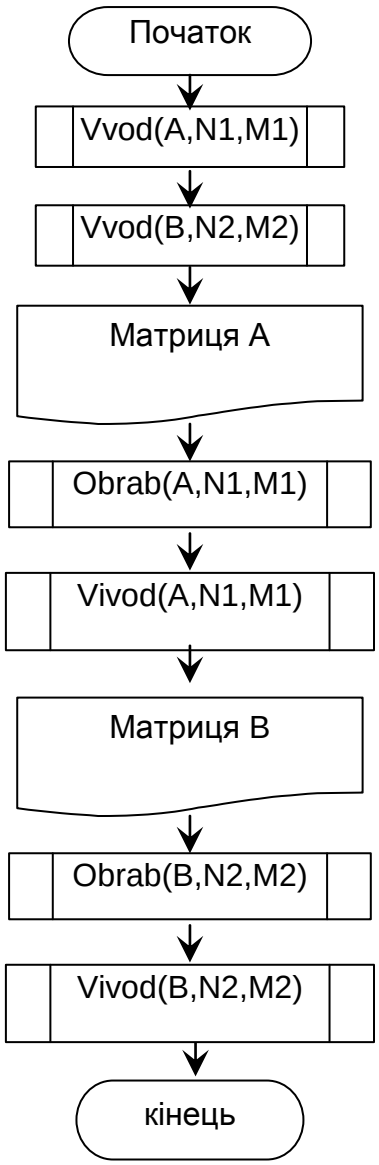


Рисунок 23 - Блок-схема алгоритму основної програми

Текст програми

```
Program Lab4a;
{Селективна Обробка Матриць}
Uses Crt;
Const N1=5; M1=4; N2=4; M2=3;
Type  Matr=Array[1..N1,1..M1] Real;
Var  A, B : Matr;  I, J : Integer;
{-------Процедура Введення Матриці-------}
Procedure Vvod(Var X:Matr;K,L:Integer);
Begin
Writeln(' Введіть Матрицю ',K:2,' X ',L:2);
For I:=1 To K Do
Begin
Writeln(I:2,'-й Рядок');
For J:=1 To L Do Read(X[I,J]);
```

```
Writeln;
End;
End;
{-------Процедура Виведення Матриці-------}
Procedure Vivod(Var X:Matr;K,L:Integer);
Begin
Writeln(' Початкова Матриця ');
For I:=1 To K Do
Begin
For J:=1 To L Do Write(X[I,J]:5:2,' ');
Writeln;
End;
Writeln;
End;
{-------Процедура Обробки-------}
Procedure Obrab(X:Matr;K,L:Integer);
Var Min : Real; Imin, Jmin : Integer;
Begin
Min:=10e30;
Imin:=0;
Jmin:=0;
For I:=1 To K Do
For J:=1 To L Do
If (X[I,J]>0) And (X[I,J]<Min) Then
Begin
Min:=X[I,J];
Imin:=I;
Jmin:=J;
End;
If Imin=0 Then Writeln('Додатних Нема')
Else
Begin
Writeln('Найменший Додатний Елемент = ',Min:6:2);
Writeln(' Рядок -',Imin:2,' Стовпець - ',Jmin:2);
End;
Writeln;
End;
{-------Основна Програма-------}
Begin
Clrscr;
Vvod(A,N1,M1);
Vvod(B,N2,M2);
Clrscr;
Writeln('Матрица A');
Vivod(A,N1,M1);
```

```
Obrab(A,N1,M1);
Writeln('Матрица B');
Vivod(B,N2,M2);
Obrab(B,N2,M2);
Readkey;
End.
```

Приклад 4b. Для кожної з матриць **A** і **B** обчислити добуток сум додатних елементів головної діагоналі і від'ємних елементів 1-го стовпця. Розмірність матриць довільна.

Блок- схеми (рис. 24, 25, 26, 27)

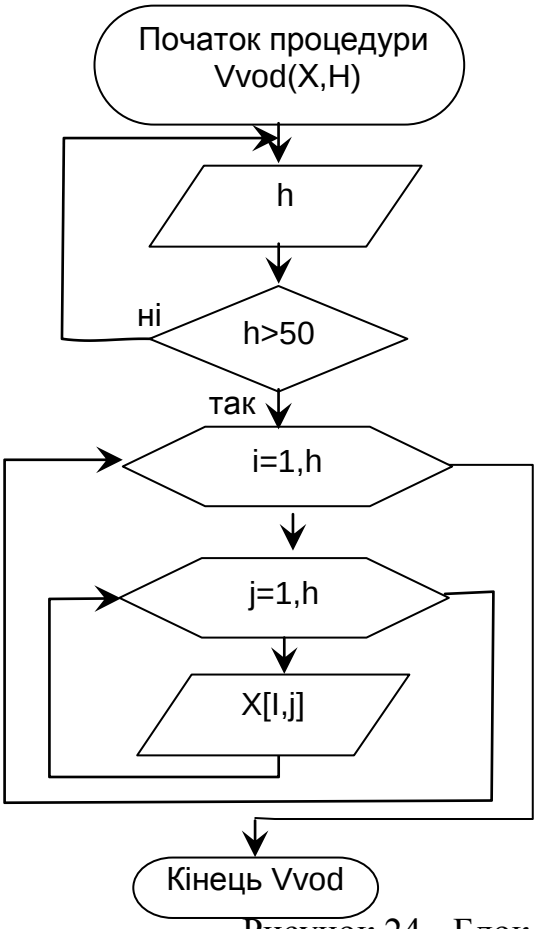


Рисунок 24 - Блок-схема алгоритму процедури Vvod

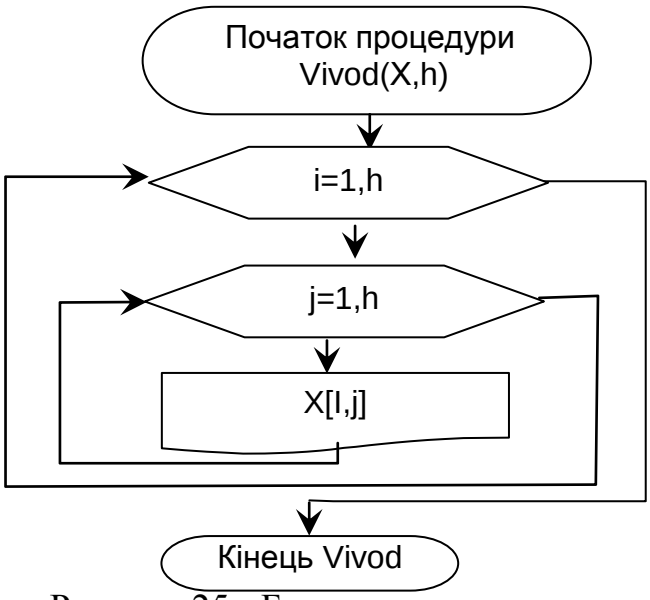


Рисунок 25 - Блок-схема алгоритму процедури Vivod

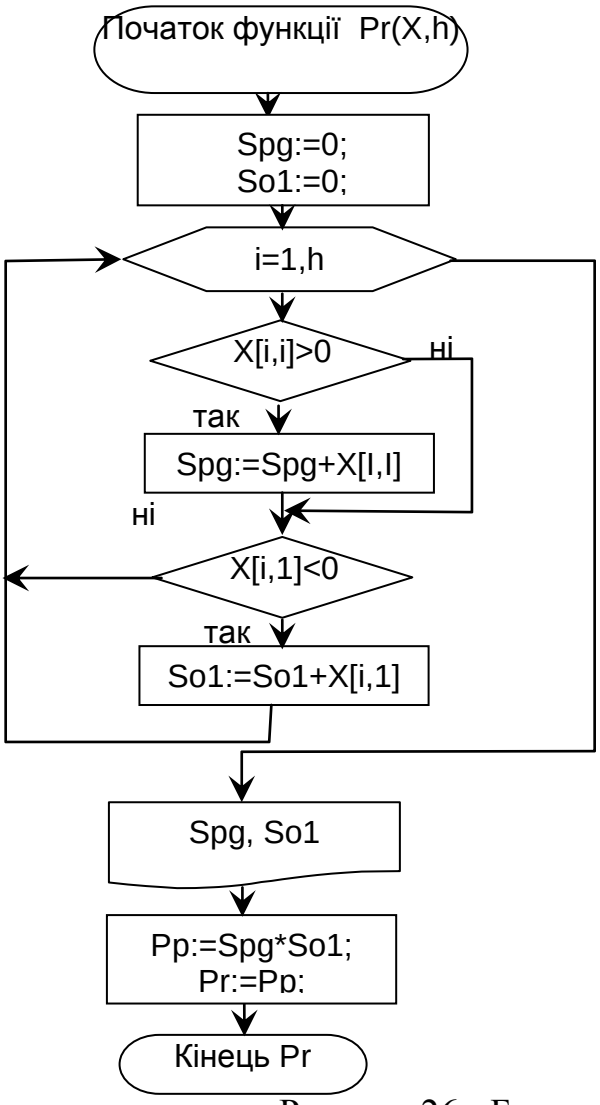


Рисунок 26 - Блок-схема алгоритму функції Pr

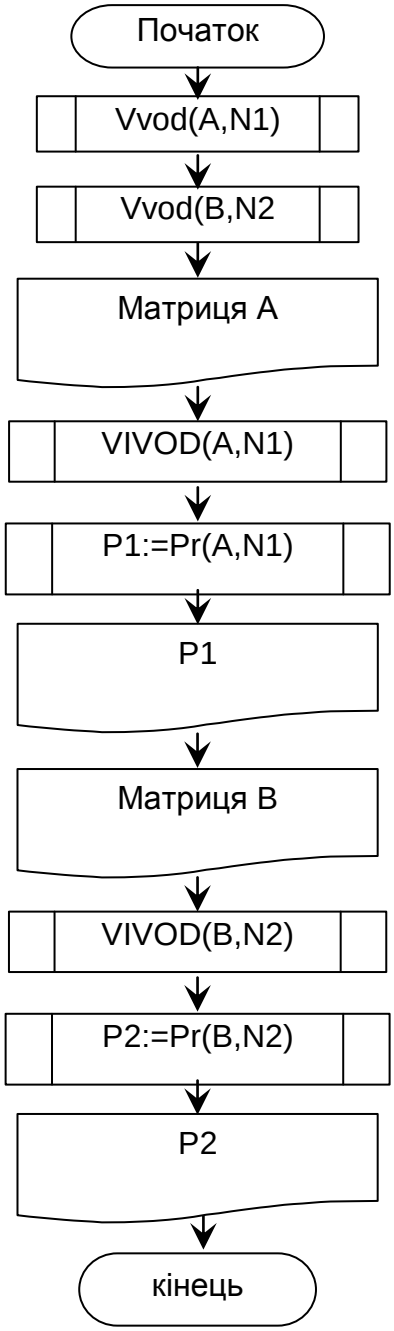


Рисунок 27 - Блок-схема алгоритму основної програми

Текст програми

```
Program Lab4b;
{Селективна Обробка Матриць}
Uses Crt;
Type  Matr=Array[1..50,1..50] Real;
Var  A, B : Matr;
I, J,N1, N2 : Integer;
P1, P2 : Real;
```

```
{-------Процедура Введення Матриці-------}
Procedure Vvod(Var X:Matr;Var H:Integer);
Label 1;
Begin
1:Writeln(' Введіть Розмірність Матриці <50');
Readln(H);
If H>50 Then Goto 1;
Writeln(' Введіть Матрицю ',H:2,' X ',H:2);
For I:=1 To H Do
Begin
Writeln(I:2,' -Ий Рядок);
For J:=1 To H Do Read(X[I,J]);
Writeln;
End;
End;
{-------Процедура Виведення Матриці-------}
Procedure Vivod(Var X:Matr;H:Integer);
Begin
Writeln(' Початкова Матриця ');
For I:=1 To H Do
Begin
For J:=1 To H Do Write(X[I,J]:5:2,' ');
Writeln;
End;
Writeln;
End;
{-------Функція Обробки-------}
Function Pr(X:Matr;H:Integer):Real;
Var Spg, So1, Pp : Real;
Begin
Spg:=0;
So1:=0;
For I:=1 To H Do
Begin
If (X[I,I]>0) Then Spg:=Spg+X[I,I];
```

```
If (X[I,1]<0) Then So1:=So1+X[I,1]
End;
Writeln('Сума додат.елем головн. діаг.= ',Spg:6:2);
Writeln('Сума від'ємн.елем. 1-го стовп= ',So1:6:2);
Pp:=Spg*So1; Pr:=Pp;
Writeln;
End;
{-------Основна Програма-------}
Begin
Clrscr;
Vvod(A,N1);
Vvod(B,N2);
Clrscr;
Writeln('Матриця A');
Vivod(A,N1);
P1:=Pr(A,N1);
Writeln('Добуток = ',P1:7:2);
Writeln('Матриця B');
Vivod(B,N2);
P2:=Pr(B,N2);
Writeln('Добуток = ',P2:7:2);
Readkey;
End.
```

7.3 Варіанти завдання до лабораторної роботи №4

1 У кожній з матриць **X** і **У** знайти (умова А). Умова А наведена в табл.11. Матриці довільні.

Таблиця 11 - Варіанти завдання

Варіант	**Умова А**
1	Суму додатних елементів
2	Суму від'ємних парних елементів

Продовження табл.11

Варіант	Умова А
1	Суму додатних елементів
2	Суму від’ємних парних елементів
3	Добуток парних елементів
4	Максимальний додатний елемент
5	Максимальний парний від’ємний елемент
6	Добуток мінімального і максимального елементів
7	Номери мінімального і максимального елементів
8	Максимальний по модулю елемент і номер рядка і стовпця, де він знаходиться
9	Квадрат мінімального від’ємного елемента
10	Добуток мінімального елемента на суму додатних елементів
11	Кількість парних елементів
12	Кількість непарних додатних елементів
13	Кількість від’ємних і кількість додатних елементів
14	Добуток суми додатних елементів на їх кількість
15	Суму парних елементів і Добуток непарних елементів
16	Кількість елементів, більших заданого числа С
17	Добуток позитивних елементів і суму парних елементів
18	Максимальний кратний 3 елемент і його індекс
19	Частку від ділення максимального на мінімальний елементів
20	Максимальний кратний 3 і мінімальний кратний 5 елементи
21	Добуток елементів, менших заданого числа Т
22	Кількість елементів, менших мінімального кратного 5
23	Суму додатних елементів, кратних 5
24	Добуток від’ємних парних і додатних непарних елементів
25	Індекс максимального по модулю кратного 3 елемента

2 Варіанти завдань приведено в табл. 12.

Таблиця 12 - Варіанти завдання

Варіант	**Зміст завдання**
1	Знайти суму елементів, розташованих за периметром
2	У кожному стовпці матриці A(5,5) знайти максимальний елемент і поміняти його місцями з відповідним елементом головної діагоналі
3	Задані матриці A(4,5) і B(4,5). Утворити нову матрицю, для якої C(I,J) = 2A(I,J) – B(I,J)
4	У матриці C(5,7) знайти мінімальний елемент 3 рядка і поміняти його місцями з максимальним елементом 2 рядка
5	Для матриці C(6,5) сформувати вектор B, компоненти якого утворюються з суми елементів відповідних рядків матриці C
6	У матриці замінити всі додатні елементи числом +1, а від’ємні -1
7	Визначити і надрукувати кількість додатних елементів в кожному стовпці заданої матриці C(5,7)
8	Сформувати і надрукувати матрицю, кожний елемент якої є цілою частиною відповідного елемента заданої матриці B(5,4)
9	Визначити і надрукувати мінімальний елемент кожного стовпця заданої матриці B1(3,4)
10	Для кожного рядка заданої матриці P визначити і надрукувати суму елементів стовпців з парними номерами
11	Для кожного рядка заданої матриці B знайти і надрукувати номери стовпців, що містять нульові елементи
12	Для кожного стовпця заданої матриці A визначити суму елементів, що лежать нижче головної діагоналі
13	Для кожного стовпця заданої матриці A визначити добуток елементів, що лежать вище головній діагоналі
14	Знайти мінімальний елемент кожного рядка матриці B(5,4) і поміняти його місцями з першим
15	Знайти максимальний елемент кожного рядка матриці T(5,5) і поміняти його місцями з відповідним діагональним
16	Розташувати елементи кожного стовпця заданої матриці за

	збільшенням значень. Отриману матрицю надрукувати
17	Розташувати елементи кожного рядка матриці P(3,4) по убуванню. Отриману матрицю надрукувати
18	Розділити елементи кожного стовпця заданої матриці B(3,4) на останній елемент стовпця. Перетворену матрицю надрукувати
19	Визначити мінімальний елемент головної діагоналі матриці A(4,4) і надрукувати рядок і стовпець, в яких він знаходиться
20	Визначити максимальний елемент побічної діагоналі матриці B(3,3) і надрукувати рядок, в якому він знаходиться
21	Поміняти місцями мінімальний і максимальний елемент головної діагоналі матриці M(4,4)
22	Поміняти місцями мінімальний елемент головної і максимальний побічної діагоналей матриці T(4,4)
23	Обчислити середнє арифметичне матриці B(3,4) і сформувати матрицю K, кожний елемент якої дорівнює відповідному елементу матриці B, помноженому на це число
24	Знайти суму додатних елементів головної діагоналі і добуток парних елементів 2 стовпця
25	Знайти середні геометричні кожного стовпця матриці A(5,4) і записати їх в масив C

8 ЛАБОРАТОРНА РОБОТА 5. ФАЙЛИ

8.1 Теоретичні відомості

Поняття файла пов'язано з типом даних і з урахуванням розміщення інформації на зовнішніх пристроях (зокрема, на магнітних дисках).

Файлом називається послідовність компонентів одного типу. Компоненти файла можуть бути будь-якого типу, за винятком типу файла. Кількість компонентів у файлі не обмежена. Розміщуються компоненти на зовнішньому пристрої і можуть зберігатися тривалий час. (Записаний на магнітному диску файл зберігається і після виключення комп'ютера.) Для зв'язку з файлами використовуються файлові змінні.

Над файловими змінними не можна виконувати ніяких операцій (привласнювати значення, порівнювати і т.д.). Файлові змінні можна лише використовувати для обробки файлів (читання інформації з файла, запису інформації у файл і т.д.).

Загальний вид опису файлового типу:

Type T = FILE OF TK;

де **T** - ідентифікатор типу; **TK** - тип компоненти.

Кожну змінну типу «файл» треба описати в розділі **VAR**:

var F: T;

де **F** – ідентифікатор змінної типу «файл»;

T - ідентифікатор типу «файл».

Тип «файл» може бути описаний і безпосередньо при описі змінної типу «файл»:

var F: FILE TK;

Перш ніж здійснювати введення-виведення, файлова змінна повинна бути зв'язана з конкретним зовнішнім файлом за допомогою стандартної процедури **ASSIGN** (див. нижче). Потім файл повинен бути відкритий для

читання або запису. Після цього можна здійснювати організацію введення-виведення.

У кожний конкретний момент для обробки доступний тільки один компонент файла (поточний). Говорять, що на цей компонент встановлено покажчик поточної позиції файла.

Звичайно всі файли вважаються файлами послідовного доступу. Це означає, що почати писати у файл можна тільки з самого його початку, дописуючи нові компоненти послідовно, один за другим в кінець файла. Для читання також треба почати обробку файла із самого його початку. Проте за допомогою стандартної процедури **SEEK** можна встановити режим довільного доступу, переміщаючи покажчик поточної позиції файла на потрібний компонент.

Після роботи з файлом його необхідно закрити за допомогою стандартної процедури **CLOSE**.

Для роботи з файлами використовуються наступні стандартні процедури (надалі мається на увазі, що F – ідентифікатор типу «файл»):

1 **ASSIGN(F,ST**) – зв'язує файлову змінну **F** із зовнішнім файлом, ім'я якого задається в рядковому виразі **ST (**типу STRING). Ім'я файла повинне відповідати вимогам операційної системи і може містити маршрут: ім'я пристрою і імена каталогів (папок). Усі подальші операції з **F** проводитимуться над зовнішнім файлом з ім'ям, заданим в **ST.**

2 **REWRITE(F)** – створює і відкриває новий файл з ім'ям, пов'язаним із змінною **F** процедурою **ASSIGN**. Якщо файл з заданим ім'ям існує, він знищується (стирається) і замість нього створюється новий порожній файл. Якщо файл з таким ім'ям відкритий, він закривається, знищується і створюється заново. Покажчик поточної позиції файла встановлюється в початок порожнього файла (на ознаку кінця файла). Функції **EOF**(F) привласнюється значення **TRUE.**

3 **WRITE(F,A)** – записує значення змінної **A** у файл **F** на місце поточного компонента і просуває покажчик поточної позиції файла на наступний

компонент. Змінна **A** повинна бути того ж типу, що і компонент файла. Якщо **EOF(F)=TRUE**, в кінець файла додається новий компонент. Інакше здійснюється заміна (коректування) поточного компонента.

4 **RESET(F)** – відкриває існуючий файл з ім'ям, пов'язаним із змінною **F.** Файл з таким ім'ям повинен існувати. Якщо файл вже відкритий, він спочатку закривається, а потім відкривається знову. Покажчик поточної позиції файла встановлюється в початкове положення.

5 **READ(F,A)** – прочитує з файла поточний компонент, привласнює його значення змінній **A** і просуває покажчик поточної позиції файла на наступний компонент. Змінна **A** повинна бути того ж типу, що і компонент файла. Якщо **EOF(F)=TRUE**, то при спробі читання виникає помилка.

6 **SEEK(F,n)** – переміщає покажчик поточної позиції файла на компонент з номером **n**. Перший компонент файла має номер 0. Для переміщення на кінець файла можна використовувати процедуру **Seek (F, FileSize (F)).**

7 **CLOSE(F)** – закриває відкритий файл, пов'язаний із змінною **F.**

При роботі з файлами можна використовувати наступні стандартні функції:

1 **EOF(F)** – повертає статус кінця файла. Результат функції **TRUE** (істина), якщо покажчик поточної позиції файла знаходиться на ознаці кінця файла, тобто за останнім компонентом файла, і **FALSE** (брехня) – інакше.

2 **FILESIZE(F)** – повертає поточний розмір файла, тобто кількість компонентів у файлі. Якщо файл порожній, **FileSize**(F)=0**.**

3 **FILEPOS(F)** – повертає поточне значення покажчика файла. Якщо поточне значення покажчика відповідає початку файла **FilePos** повертає 0, якщо кінцю файла (тобто **EOF(F)=True**), то **FilePos(F)** = **FileSize(F).**

Функцію **EOF** можна використовувати в умовному операторі **IF** або в операторі циклу **WHILE**. Для перевірки того, що файл не закінчений (готовність до читання), можна застосовувати, наприклад, наступну конструкцію:

```
while not EOF(F) do
begin
...
READ (F,A);
...
end ;
```

У цій конструкції здійснюється послідовне читання (і обробка) компонентів файла до тих пір, поки покажчик поточної позиції указує на черговий компонент (тобто у файлі є ще не оброблені компоненти). Цикл завершиться, як тільки покажчик поточної позиції просунеться за кінець файла (тобто будуть оброблені всі компоненти файла).

На практиці дуже часто компонентами файла є записи.

Запис – це структурований тип даних, що складається з фіксованої кількості компонентів, які називаються полями. В одному полі дані мають один і той же тип, а в різних полях можуть мати різні типи. Тому записи називають комбінованими типами даних.

У записі кожне поле має своє ім'я (ідентифікатор). Кількість полів, тип і ідентифікатор кожного поля фіксуються у визначенні типа, до якого відноситься запис. Визначення починається з ключового слова **RECORD**, за яким слідує перелік всіх полів з вказівкою через двокрапку їх типів. Поля відділяються один від одного крапкою з комою. Поля запису можуть бути будь-якого типу. Якщо декілька полів мають один і той же тип, то імена полів можна перелічувати через кому і потім вказати загальний тип. Завершує визначення типу запису ключове слово **END.**

Загальний вид опису типу-запису:

```
Type T=RECORD
I1:T1;
I2:T2;
...
IN:TN
end;
```

Після введення типу запису можна задати змінні цього типу:

var A,B:T;

де **T** - ідентифікатор типу;

Ik - ім'я поля або список розділених комами імен полів;

Tk - типи полів;

A, B - ідентифікатори (імена) змінних типу запису.

Приклад. Запис має чотири поля: місто, вулиця, номер будинку і номер квартири, тобто запис є адресою. Тоді адресу можна визначити таким чином:

```
Type ADRES=RECORD
G,U:STRING[15];
D,K:integer
end;
```

Запис типу DATA включає рік, місяць, день, описується таким чином:

```
Type DATA=RECORD
YEAR, MONTH, DAY:INTEGER
end;
```

Змінні **A** типу ADRES і **D1,D2** типу DATA задаються таким чином:

```
var A:ADRES; D1,D2:DATA;
```

Доступ до полів змінних типу запису здійснюється вказівкою імені змінної і імені поля, записуваного через крапку.

Наприклад, щоб заслати у визначену вище змінну **A** адреса: "м. Краматорськ, вул. Кришталева, будинок 52, кв.128", треба виконати наступні оператори привласнення:

```
A.G:=’Краматорськ’;
A.U:=’Кришталева’;
A.D:=52;
A.K:=128;
```

Для запису в змінну **D1** дати 25.10.1996 р. слід виконати оператори:

```
D1.DAY:=25;
D1.MONTH:=10;
D1.YEAR:=1996;
```

Можна звертатися не тільки до окремого поля запису, але і до запису в цілому, використовуючи оператор привласнення. Наприклад, щоб переслати дату із змінної **D1** в **D2** можна виконати один оператор:

```
D2:=D1
```

Дія цього оператора рівнозначна дії групи операторів:

```
D2.YEAR:=D1.YEAR;
D2.MONTH:=D1.MONTH;
D2.DAY:=D1.DAY;
```

Іншими словами, звернення до змінної типу запису – це звернення одночасно до всіх полів цієї змінної: виконання якихось дій над змінною-записом – це виконання даних дій над кожним її полем.

Для того, щоб не записувати кожного разу ім'я змінної при зверненні до полів запису, можна використовувати оператор над записами **WITH**. У цьому випадку можна указувати тільки поле змінної. Наприклад, для занесення в змінну **A** адреси замість наведеного раніше фрагмента програми можна використовувати наступний:

```
with A do
begin
G:=’Краматорськ’;
U:=’Кришталева’;
D:=52;
K:=128
end;
```

Тип запису можна вводити безпосередньо при визначенні змінних. Наприклад, визначити наведені вище змінні типу запису **A, D1** і **D2** можна так:

```
var A: record
G,U:string[15];
D,K:integer
end;
D1,D2:record
YEAR, MONTH, DAY:integer
```

```
end;
```

Як наголошувалося вище, поля запису можуть бути будь-якого типу, у тому числі записами або масивами, наприклад:

```
type STRUCT=record
FIO:string[13];
KOL:integer;
A:ARRAY[1.10] integer
end;
```

У свою чергу, тип запису можна задавати не тільки для окремих змінних, але і для компонентів масиву. Іншими словами, можна визначати масиви, компонентами якого є записи. Наприклад, після введення вищенаведеного типу запису ADRES можна визначити наступну змінну типу «масив» (для створення масиву адресів):

```
var MAS=ARRAY[1..20] ADRES;
```

Доступ до полів в цьому випадку здійснюється так, як мовилося вище. Проте в цьому випадку після імені змінної слід вказати ще і її індекс, наприклад:

```
MAS[7].G:='Краматорськ';
MAS[7].U:='Шкадінова';
```

Використовування оператора **WITH** дозволяє спростити це звернення :

```
with MAS[7] do
begin
G:='Краматорск';
U:='Шкадинова';
end;
```

8.2 Приклад виконання лабораторної роботи 5

Скласти програму формування на магнітному диску телефонного довідника, кожний рядок (компонент) якого складається з наступних полів:

- прізвище;
- ім'я;

- вулиця;
- будинок;
- квартира;
- телефон.

У цій програмі передбачити друкування довідника осіб, прізвища яких починаються на задані букви.

При написанні цієї програми її краще розбити на дві. Перша призначена для введення даних та формування файла. Друга – виконує обробку файла.

Нижче наведено текст першої програми.

```
program lab5a;
uses CRT;
Type ADRES=record
F,I,U:string[15];
D,K:integer;
TEL:string[8];
end;
Type FIL1=file of ADRES;
Var FF:FIL1;
vv:integer;
A:ADRES;
st:string[5];
{-----------------------------------------------}
begin
assign (FF,'ADRESA.DAT');
{---------- Створення файла ---------------------}
rewrite (FF);
with A do
repeat
clrscr;
write    ('фамилия  '); readln (F);
if F <> '*' then
begin
```

```
write ('имя      '); readln (I);
write ('улица    '); readln (U);
write ('дом      '); readln (D);
write ('квартира '); readln (K);
write ('телефон  '); readln (TEL);
write (FF,A);
end;
until F='*';
close (FF)
end.
```

Треба відмітити, що при кожному запуску цієї програми файл, створений при попередньому запуску програми, знищується і на його місці створюється новий. Таким чином, файл кожний раз формується заново. Дозапис в файл неможливий. Для забезпечення такої можливості треба замість процедури

```
rewrite (FF);
```

використати процедури:

```
reset (FF);
seek(ff,FileSize(ff));
```

Але попередньо необхідно забезпечити створення файла, хай навіть пустого. Текст програми, що здійснює обробку файла наведено нижче.

```
program lab5a;
uses CRT;
Type ADRES=record
F,I,U:string[15];
D,K:integer;
TEL:string[8];
end;
Type FIL1=file of ADRES;
Var FF:FIL1;
vv:integer;
A:ADRES;
st:string[5];
```

```
cc:char;
{-------------------------------------------}
begin
assign (FF,'ADRESA.DAT');
{------------ Обробка файла --------------------}
repeat
clrscr;
write ('Введите перші символи або "*".  ');
readln (ST);
if ST <> '*' then
begin
reset (FF);
vv:=0;
while not eof (FF) do
begin
read (FF,a);
if (copy(A.F,1,length(ST))=ST) then
begin
with A do
writeln(F:15,' ',I:15,' ',U:15,' ',d:3,k:4,' ',TEL);
vv:=vv+1;
end
end;
writeln;
if (vv=0) then
writeln('Нема прізвищ, що починаються з ',ST);
writeln;
writeln ('Нажмите будь-яку клавішу');
CC:=ReadKey;
end;
until ST='*';
{-------------------------------------------}
close (FF)
end.
```

8.3 Варіанти завдань до лабораторної роботи 5

Створити файл і виконати його обробку відповідно до табл. 13. Обробка полягає у виборі з файла і видачі на екран інформації, що вимагається. Варіанти структури файла наведені в табл. 14.

Примітки:

1 Якщо в таблиці 10 сказано "заданий" (задане місто, задана група і т.д.), значення такого даного необхідно визначити в процесі виконання програми, тобто організувати введення такого значення з клавіатури.

2 У задачах, де йдеться про вік, необхідно обчислити вік як різницю між заданою датою (роком) і роком народження

Таблиця 13 - Варіанти завдання

Варіант завдання	**Варіант структури файла**	**Зміст завдання**
1	1	Список неуспішних (мають двійки) студентів заданої групи
2	5	Список співробітників пенсійного віку (чоловіки старше 60, жінки – 55 років)
3	6	Список іногородніх пацієнтів (які проживають за межами м. Краматорська) старших 60 років
4	3	Список студентів, що проживають не там, де народилися (що змінили місце проживання)
5	2	Список студентів, у яких батько або мати підприємці (в полі "місце роботи" вказано "підприємець")
6	4	Наявність в бібліотеці книг, заданого автора
7	1	Список студентів заданої групи, у яких немає двійок
8	2	Список студентів заданої групи, у яких батько і мати працюють в одному і тому ж місці

Продовження табл.13

Варіант завдання	**Варіант структури файла**	**Зміст завдання**
9	4	Список книг, виданих раніше заданого року
10	3	Список студентів заданої групи з багатодітних сімей (які мають більше двох братів і сестер)
11	7	Список рейсів в задане місто, на які є які-небудь квитки
12	6	Список пацієнтів, що проживають там, де народилися
13	5	Список співробітників заданого відділу, які поступили на підприємство після 1996 р.
14	1	Список студентів із заданого міста, у яких немає двійок
15	7	Список всіх рейсів з часом відправлення з 20.00 до 8.00
16	3	Список студентів м. Краматорська, у яких немає братів і сестер
17	4	Список книг заданого автора, що мають більше 300 сторінок
18	5	Список співробітників відділу ВЦ, що пропрацювали в установі більше 6 років
19	6	Список пацієнтів старше 40 років, що мають заданий діагноз
20	2	Список студентів, що проживають не в м. Краматорську, але хоча б один батько яких працює на ЗАТ НКМЗ
21	7	Список всіх рейсів в задане місто, що відправляються після 18.00 до 8.00
22	5	Список співробітників відділу Маркетингу, молодше 25 років
23	1	Список студентів груп, шифр (найменування) яких починається з символів ЕСА

Продовження табл.13

Варіант завдання	**Варіант структури файла**	**Зміст завдання**
24	4	Інформацію про місцезнаходження книги, що вимагається (задається прізвище автора і найменування книги)
25	3	Список студентів заданої групи, які проживають там, де народилися
26	7	Список всіх рейсів, на які є квитки першого класу
27	6	Список чоловіків, молодше 40 літ, що мають заданий діагноз
28	5	Список співробітників, посада яких починається з символів "нач" або "рук"
29	1	Список студентів заданої групи, що не здали іноземну мову
30	7	Список рейсів, на які продані всі квитки

Таблиця 14 - Варіанти структури файла

Варіант структури	**Структура файла**
1	1) Прізвище студента; 2) Найменування групи; 3) Місце проживання; 4) Оцінка: з фізики, математики, іноземної мови, програмування
2	1) Прізвище студента; 2) Найменування групи; 3) Місто, де він живе; 4) Вулиця, де він живе; 5) Місце роботи батьків: батька, матері
3	1) Прізвище студента; 2) Найменування групи; 3) Місце проживання; 4) Місце народження; 5) Кількість братів і сестер
4	1) Номер книги (код, шифр); 2) Прізвище автора; 3) Найменування книги; 4) Рік видання; 5) Кількість сторінок; місце знаходження
5	1) Прізвище співробітника; 2) Назва відділу; 3) Посада; 4) Рік

	народження; 5) Дата вступу в підприємство; 6) Стать (м – чоловіча, ж – жіноча)
6	1) Прізвище пацієнта; 2) Місце народження; 3) Рік народження; 4) Місце народження; 5) Діагноз; 6) Стать (м – чоловіча, ж – жіноча)
7	1) Номер рейса літака; 2) Пункт призначення; 3) Час відправлення; 4) Наявність квитків 1 класу; 5) Наявність квитків 2 класу

9 ЛАБОРАТОРНА РОБОТА 6. ЕЛЕМЕНТИ ГРАФІКИ

9.1 Теоретичні відомості

До теперішнього часу ми використовували екран ЕОМ в текстовому режимі. Для зображення на екрані рисунків, креслень, графіків функцій і т.д. необхідно перемкнути екран ЕОМ в графічний режим. У графічному режимі зображення на екрані складається з точок (pixel) залежно від монітора – чорно-білих або кольорових, по горизонталі 320 або 640 точок, а по вертикалі - 200 або 480. Можлива й інша кількість точок (1024 і більш).

Для ідентифікації точок на екрані в графічному режимі використовується система ЕКРАННИХ (ФІЗИЧНИХ) координат, початок якої розташований у верхньому лівому кутку, а вісь **У** направлена вниз. Фізично координати вимірюються в цілих числах від нуля до максимальних значень, які однозначно обумовлюються номером графічного режиму. Ці значення можна отримати за допомогою функцій **GetMaxX** і **GetMaxY**

Геометричні побудови можуть проводитися як на всій робочій ділянці екрану, так і на будь-якій прямокутній ділянці меншого розміру, яка називається ГРАФІЧНИМ ВІКНОМ. Вікно визначається оператором **SetViewPort** .

Геометричні побудови виконуються в кольорі. Як і в текстовому режимі, розрізняються колір фону і колір переднього плану – колір малювання. Коди кольорів такі ж, як і в текстовому режимі (див. стандартний модуль **CRT**).

Засоби роботи в графічному режимі зосереджені в стандартному модулі **graph.tpu**, який має бути підключеним до програми за допомогою оператора

USES GRAPH;

Цей модуль містить великий набір стандартних процедур, функцій, констант, змінних, які дозволяють будувати лінії, дуги, різні фігури (як незаповнені, так і заповнені): прямокутники, багатокутники, кола, еліпси,

сектори. Це базові елементи. На підставі них можна будувати вже будь-які графічні зображення.

Розглянемо деякі процедури.

1 Включення графічного режиму (звичайно екран знаходиться в текстовому режимі):

```
Gd=0
InitGraph (Gd,Gm,'')
```

Здійснюється перевірка устаткування і автоматично вибирається самий відповідний драйвер і режим. Можна примусово задати номер драйвера (**Gd > 0**) і режим (**Gm**). Проте в цьому випадку потрібне знання технічних і програмних засобів.

2 Відключення графічного режиму і перехід в текстовий:

```
CloseGraph
```

3 Визначення кольору фону:

```
SetBkColor(Color)
```

4 Визначення кольору переднього плану:

```
SetColor(Color)
```

5 Визначення товщини (**t=1,3**) і форми (**f=0,1,2,3**) лінії:

```
SetLineStyle(t,0,f)
```

6 Визначення орнаменту (**0 <= Pat <= 11**) і кольору заповнення фігур:

```
SetFillStyle(Pat,Color)
```

7 Виведення (побудова) окремої точки з координатами (**x,y**) заданого кольору:

```
PutPixel(x,y,Color)
```

8 Побудова лінії, що сполучає:

а) точки (**x1,y1**) і (**x2,y2**):

```
Line(x1,y1,x2,y2)
```

b) поточну точку і точку (**x,y**):

```
LineTo(x,y)
```

c) поточну точку (**x1,y1**) і точку (**x1+dx**,**y1+dy**):

```
LineRel(dx,dy)
```

9 Побудова кола з центром в точці (**x,y**) і радіусу **r**:

Circle(x,y,r)

10 Побудова дуги кола з центром в точці (**x,y**), радіусу **r** і укладеній між радіус-векторами з кутами **StAngle** і **EndAngle**. Кути відлічуються проти годинникової стрілки від напряму осі абсцис у градусах.

Arc(x,y,StAngle,EndAngle,r)

11 Побудова дуги еліпса з напівосями **Xred** і **Yred**. Решта параметрів – як в попередній процедурі.

Ellipse(x,y,StAngle,EndAngle,Xrad,Yrad)

12 Побудова заповненого заданим орнаментом еліпса:

FillEllipse(x,y,Xrad,Yrad)

13 Побудова заповненого заданим орнаментом сектора еліпса:

Sector(x,y,StAngle,EndAngle,Xrad,Yrad)

14 Побудова заповненого заданим орнаментом сектора круга:

PieSlice(x,y,StAngle,EndAngle,r)

15 Побудова прямокутника з протилежними вершинами (**x1,y1**) і (**x2,y2**):

a) не заповненого:

Rectangle(x1,y1,x2,y2)

b) заповненого:

Bar(x1,y1,x2,y2)

Будь-який графічний режим разом з відтворенням геометричних об'єктів дозволяє виводити і текстову інформацію.

16 Виведення на екран послідовності символів:

a) щодо поточної точки:

OutText(Text)

b) щодо точки (x,y):

OutTextXY(x,y,Text)

Не можна виводити значення цифрових змінних, необхідно заздалегідь ці значення перетворити в текстовий тип (тип STRING). Це можна зробити, наприклад, за допомогою стандартної процедури STR:

STR(<змінна>:<шаблон>,<stroka>)

Зате можна міняти шрифт, напрям тексту і т.д.:

SetTextJustify(Horiz,Vert)

SetTextStyle(Font,Direction,Size)

9.2 Приклади побудови графічних зображень

Програмні (МАТЕМАТИЧНІ) або СВІТОВІ координати точок, що задають контури зображення, можуть не співпадати за своїми межами з фізичними межами графічного вікна. Приведення МАТЕМАТИЧНИХ (СВІТОВИХ) координат до ФІЗИЧНИХ повинне здійснюватися програмою користувача. У мові немає вбудованих засобів перетворення. Для заповнення цих пропусків розробимо декілька стандартних програм.

Хай **Xmin**, **Xmax**, **Ymin**, **Ymax** – межі зміни математичних координат. А межі зміни екранних можна визначити так:

0,GetMaxX,0,GetMaxY.

Позначимо через **kX** і **kY** відношення меж зміни екранних і математичних координат уздовж осі **X** і **У** відповідно:

$$kX = \frac{GetMaxX}{X\max - X\min}, \quad kY = \frac{GetMaxY}{Y\max - Y\min} \qquad (1)$$

де **kX** і **kY** – це коефіцієнти пропорційності (або масштабні множники). Тоді математичні координати крапки (**X,Y**) перетворюються в екранні (**Xэ,Yэ**) таким чином:

$$Xэ = (X\max - X) : kX, \quad Yэ = (Y\max - Y) : kY \qquad (2)$$

Визначення коефіцієнтів **kX і kY (1)** винесемо в процедуру **InitMat**, а власне перетворення **(2)** – у функції **EkrX(x) і EkrY(y).** Помістимо ці

підпрограми разом з необхідними змінними в окремий зовнішній модуль **ModGr**:

```
{-Інтерфейс математичних і екранних координат }
unit ModGr;
interface
procedure InitMat;
function EkrX(X: Real): Integer;
function EkrY(У: Real): Integer;
var      XMax, XMin, YMax, YMin, KX, KY : Real;
X0, Y0: Integer;
var Gd, Gm : Integer;
implementation
uses Graph,Crt;
{--- Процедура ініціалізації інтерфейсу------}
procedure InitMat;
begin
InitGraph(Gd, Gm, ''); {Ініціалізація графіки }
KX := GetMaxX / (XMax - XMin);
KY := GetMaxY / (YMax - YMin);
X0 := EkrX(0);
Y0 := EkrY(0);
end;
{---------- Інтерфейс по координаті X ----------}
function EkrX;
begin
EkrX := Round((X - XMin) * KX)
end;
{----------- Інтерфейс по координаті У ---------}
function EkrY;
begin
EkrY := Round((YMax - У) * KY)
end;
end.
```

Слід нагадати, що зовнішній модуль компілюється в окремий файл. На магнітному диску буде освічений файл ModGr.tpu, який можна використовувати в будь-якій програмі, помістивши у неї оператор USES ModGr;

Тепер можемо писати конкретні програми.

Приклад 6а. Побудувати графік довільної функції y=f(x). Область визначення функції (інтервал зміни X) задавати у діалозі.

```
{------ Програма малювання графіка функції ------}
program LAB6a;
uses Graph, ModGr, Crt;
var Gd, Gm : Integer;
Xn, Xk, h, x, y : Real;
{----------------- Функція ----------------------}
function F(X : Real) : Real;
begin
F :=cos(2 * x * x);
end;
{------------ Процедура табуляції ---------------}
procedure TabFunc( Xn, Xk, h: real);
var X, У : Real;
begin
X := Xn;
YMax := 0;
YMin :=0;
while  X<= Xk do
begin
У := F(X);
if У > YMax then YMax := У;
if У < YMin then YMin := У;
X := X + h
end;
XMin := Xn;
if XMin > 0 then XMin := 0;
```

```
XMax := Xk;
if XMax < 0 then XMax := 0;
end;
{---------------Основна програма ----------------}
begin
WriteLn('Введите Xn, Xk, h'); ReadLn(Xn, Xk, h);
TabFunc(Xn, Xk, h);
InitMat;
Line(X0, 0, X0, GetMaxY);
Line(0, Y0, GetMaxX, Y0);
X := Xn;
while  X<= Xk do
begin
У := F(X);
PutPixel(EkrX(X), EkrY(Y), 7+128);
X := X + h
end;
ReadKey;
CloseGraph; { Відключення графіки }
end.
```

Приклад 6b. У масив A вводиться значення деякого показника, наприклад об'єм випуску продукції. Вивести на екран гістограму (східчастий графік), що відображає значення показників.

```
{------ Програма виведення гістограми -----------}
program LAB6b;
uses Graph, ModGr, Crt;
type mas=ARRAY[1..30] real;
var x1, x2, y : Real;
i, n, s, c : integer;
A: mas;
St : string[8];
{--------- Процедура введення ------------------}
procedure VVOD( var A:mas; n: integer);
begin
```

```
writeln('Введіть', n:3,' показника');
for i:=1 to n do read(A[i]);
writeln;
end;
{---------Визначення меж матем. координат--------}
procedure pr1(A:mas; n:integer);
begin
Ymin:=0;
Ymax:=0;
for i:=1 to n do
begin
if A[i] > Ymax then Ymax:=A[i];
if A[i] < Ymin then Ymin:=A[i];
end;
Ymax:=Ymax+(Ymax-Ymin)/20;
Xmin:=-1;
Xmax:=n+1;
end;
{--------------Основна програма -----------------}
begin
WriteLn('Введите  кількість показників <= 30');
ReadLn(n);
VVOD(A,n);
PR1(A,n);
InitMat;
Line(X0, 0, X0, GetMaxY);
Line(0, Y0, GetMaxX, Y0);
s:=0;
c:=0;
SetTextJustify(CenterText,TopText);
for i:=1 to n do
begin
x1 := i-1;
x2 := i;
y  := A[i];
```

```
s:=s+1;
c:=c+1;
if s > 11 then s:=0;
if з > 15 then c:=0;
SetFillStyle(s,c);
Bar(EkrX(x1), Y0, EkrX(x2), EkrY(y));
Rectangle(EkrX(x1), Y0, EkrX(x2), EkrY(y));
Str(y:4:1,St);
OutTextXY(EkrX((x1+x2)/2),EkrY(y)-10,St);
end;
ReadKey;
CloseGraph; { Відключення графіки }
end.
```

9.3 Варіанти завдання до лабораторної роботи 6

Побудувати графік функції

$$Y=\begin{cases} f1(x), & x<0 \\ f2(x), & 0\le x\le 1 \\ f3(x), & x>1 \end{cases}$$

Види функцій **f1, f2, f3** задані в табл.15.

Таблиця 15 - Види функцій f1, f2, f3

Варіант	f1(x)	f2(x)	f3(x)
1	$ctg(2x^3)$	$\sqrt[3]{x}-1$	$\cos(x-2)$
2	$\cos(3x^2)$	$\sqrt{x^3}$	$\ln(5x)$
3	$\sin(x^3+5)$	$\ln(4x+1)^2$	$\sqrt{x+1}$
4	x^4+2x^3-x	$\sin^2(x^3)$	$\sin(x^3)$

Варіант	f1(x)	f2(x)	f3(x)
5	x^5	$\ln(x+1)$	$\sqrt[3]{x}$
6	$\sin(4x^3)$	$\sqrt[5]{6x-x^2+1}$	$\mathrm{Sin}(2x+1)^3$
7	$\mathrm{ctg}(3x-1)^2$	$2+xe^{-x}$	$\sin^3(x^2)$
8	$\sin(x^3+1)$	$(x-1)^3$	$\sqrt{1+x^3})$
9	$\lvert x\rvert^2$	$\sin(x^2)$	$\ln^2(x)$
10	$\sin^2(x^3)$	$\ln(x^3+3)$	e^{-x}
11	$2xe^{-x}$	$\ln(x+1)$	x^x
12	$\ln(2x+5)$	$\sin(e^x)$	$\sqrt[3]{x}$
13	$\sin(2x+1)$	$\cos(x^3)$	$\sin(x)^2$
14	$\sin(x)^2$	$\sqrt[4]{x}$	$\ln(x^3)$
15	$3x-1)$	$\ln^2(x)$	$\sqrt{1+x^2}$
16	$\cos(x)$	$1/(\mathrm{tg}(2x)+1)$	x^2e^{-x}
17	$\lvert x\rvert^2$	$x^2\cos(x)$	$\sin(x^2)$
18	$\mathrm{tg}(2x)$	$\cos(2x)$	$\ln^2(x)$
19	$1/x^5$	e^{x+1}	$\sin(x^5)$
20	$3x^5$	$\sin(4x+1)^2$	$\ln(x+1)$
21	$\sqrt{4+x^2}$	3^{x+3}	5^{x+1}
22	e^{-3x}	$\sin^3(x^4)$	$\sqrt[3]{3x}$
23	$\sin(e^x)$	$\mathrm{tg}(2x)$	$\sqrt{1+x^2}$
24	$\cos(x^3)$	$1/x^5$	x^2e^{-x}
25	$\sqrt[4]{x}$	$3x^5$	$\sin(x^2)$

10 Питання для контролю

1 Алгоритм. Поняття алгоритму. Способи опису алгоритму.

2 Блок-схема. Основні елементи блок-схеми.

3 Середовище програмування **Turbo Pascal.**

4 Збереження й завантаження програм у середовищі **Turbo Pascal.**

5 Робота із фрагментами тексту в середовищі **Turbo Pascal.**

6 Створення ***Pascal-програм*** у середовищі ***Delphi*** у режимі консолі.

7 Ідентифікатор.

8 Коментар. Використання коментарів у програмі.

9 Види даних.

10 Типи даних.

11 Прості й структуровані дані.

12 Константи. Опис констант.

13 Змінні. Опис змінних.

14 Стандартні функції.

15 Вирази.

16 Арифметичні вирази.

17 Логічні вирази. Відношення.

18 Структура програми.

19 Запис операторів в ***Pascal.***

20 Складений оператор.

21 Підпрограма. Види підпрограм. Структура підпрограми.

22 Процедури.

23 Функції.

24 Формальні й фактичні параметри.

25 Види формальних параметрів. Способи передачі значень параметрів у підпрограму.

26 Стандартні функції.

27 Модулі. Призначення й використання.

28 Структура модуля.

29 Стандартні модулі.

30 Оператор присвоювання.

31 Оператор вводу даних.

32 Оператор виводу даних.

33 Розгалуження програми. Умовний оператор.

34 Оператор вибору варіанта.

35 Цикл. Елементи циклу.

36 Оператор циклу з попередньою перевіркою умови **WHILE.**

37 Оператор циклу з наступною перевіркою умови **REPEAT.**

38 Відмінність операторів **WHILE** та **REPEAT.**

39 Цикл із лічильником.

40 Табулювання функції.

41 Пошук экстремумів функції методом перебору.

42 Масиви. Визначення й обробка масивів.

43 Знаходження найменшого й найбільшого елементів масиву.

44 Сортування масивів.

45 Селективна обробка масивів.

46 Двовимірні масиви. Визначення й обробка.

47 Вкладені цикли.

48 Селективна обробка двовимірних масивів.

49 Файл. Поняття файлу.

50 Файлові змінні. Опис файлових змінних.

51 Робота з файлами.

52 Послідовна обробка файлів.

53 Процедури доступу до файлу для запису.

54 Процедури доступу до файлу для читання.

55 Визначення кінця файлу при його перегляді.

56 Записи. Призначення.

57 Визначення (опис) записів.

58 Вкладеність записів.

59 Доступ до полів запису.

60 Оператор доступу до полів запису Width.

СПИСОК ЛІТЕРАТУРИ, ЩО РЕКОМЕНДУЄТЬСЯ

1 Новичков В.С. Паскаль: Учебное пособие /В.С.Новичков, Н.И.Парфилова, А.Н.Пылькин. - М.: Высш. школа, 1990. - 223 с.

2 Васюкова Н.Д. Практикум по основам программирования. Язык Паскаль/ Н.Д.Васюкова, В.В.Тюляева. - М.: Высш. школа, 1991. - 160 с.

3 Белецкий Я. Турбо Паскаль с графикой для персональных компьютеров. - М.: Машиностроение, 1991. - 320 с.

4 Алексеев В.Е. Вычислительная техника и программирование: Практикум по программированию/ В.Е.Алексеев, А.С.Ваулин, Г.Б.Петрова. - М.: Высш. школа, 1991. - 324 с.

5 Абрамов Г.В. Введение в язык Паскаль/Г.В.Абрамов, Н.П.Трифонов, Г.Н.Трифонова. - М.: Наука, 1988. - 320 с.

6 Семашко Г.Л. Программирование на языке Паскаль/ Г.Л.Семашко, А.И.Салтыков. - М.: Наука,1988. - 128 с.

7 Абрамов С.А. Начала программирования на языке Паскаль/С.А.Абрамов, Е.В.Зима. - М.: Наука,1987. - 112 с.

8 Перминов О.Н. Язык программирования Паскаль. - М.: Радио и связь, 1989. - 128 с.

9 Вальвачев А.Н. Программирование на языке Паскаль для персональных ЭВМ ЕС/ А.Н.Вальвачев, В.С. Крисевич. - Минск: Выш. шк., 1989. - 223 с.

10 Сердюченко В.Я. Розробка алгоритмів та програмування мовою TURBO-PASCAL. – Харків: Парітет, 1995. – 352 с.

11 Форсайт Р. Паскаль для всех / пер. с англ. М.В. Сергиевского, А.В. Шалашова под ред. Ю.И. Тончеева. – М.: Машиностроение, 1986. – 288 с.

Навчальне видання

ФОКІН Анатолій Григорович

ГЕТЬМАН Ірина Анатоліївна

ПРОГРАМУВАННЯ В СЕРЕДОВИЩІ PASCAL

Навчальний посібник

для студентів вищих навчальних закладів

Редактор І.І.Дьякова

Верстка О.П.Ордіна

34/2006. Підп. до друку Формат 60х84/16.
Папір офсетний. Ум. друк. арк. Обл.-вид. арк.
Тираж прим. Зам. №
Видавець і виготівник
«Донбаська державна машинобудівна академія»
84313, м. Краматорськ, вул. Шкадінова, 72

Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру
серії ДК № 1633 від 24.12.2003 р.