ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА З'ВЯЗКУ

Кафедра «Обчислювальна техніка та системи управління»

І.Г. Філіппенко, В.О. Гончаров, В.С. Меркулов

ПРОГРАМУВАННЯ ІНЖЕНЕРНО-ТЕХНІЧНИХ ЗАДАЧ В СЕРЕДОВИЩІ QBASIC

Конспект лекцій з дисципліни
"*ОБЧИСЛЮВАЛЬНА ТЕХНІКА І ПРОГРАМУВАННЯ*"

Частина 3

Харків 2007

Філіппенко І.Г., Гончаров О.В., Меркулов В.С. Програмування інженерно-технічних задач в середовищі QBasic: Конспект лекцій. – Харків: УкрДАЗТ, 2007. – Ч. 3. - 132 с.

Даний конспект розроблено у відповідності до програми курсу "Обчислювальна техніка і програмування". До нього включено розділи, необхідні студентам для опанування роботою на персональному комп'ютері в середовищі QBasic, зокрема, основи програмування та прийоми відлагодження основних алгоритмічних структур.

Конспект має численні приклади розв'язання задач або їх фрагменти.

Рекомендується для студентів академії денної форми навчання, що вивчають дисципліни "Обчислювальна техніка і програмування", "Інформатика та комп'ютерна техніка", "Основи інформаційних технологій", "Комп'ютерна техніка та програмування" всіх спеціальностей та форм навчання.

Іл. 11, табл. 3, бібліогр: 7 назв.

Конспект лекцій розглянуто та рекомендовано до друку на засіданні кафедри "Обчислювальна техніка та системи управління 6 березня 2007 р., протокол № 7.

Рецензент
проф. Г.І. Загарій

І.Г. Філіппенко, В.О. Гончаров, В.С. Меркулов

ПРОГРАМУВАННЯ ІНЖЕНЕРНО – ТЕХНІЧНИХ ЗАДАЧ В СЕРЕДОВИЩІ QBASIC

Конспект лекцій з дисципліни
“Обчислювальна техніка і програмування”

Частина 3

Відповідальний за випуск Меркулов В.С.

Редактор Еткало О.О.

Підписано до друку 27.03.07 р.
Формат паперу 60х84 1/16 . Папір писальний.
Умовн.-друк.арк. 8,25. Обл.-вид.арк. 8,5.
Замовлення № Тираж 300 Ціна

Видавництво УкрДАЗТу, свідоцтво ДК 2874 від. 12.06.2007 р.
Друкарня УкрДАЗТу,
61050, Харків - 50, пл. Фейєрбаха, 7

УКРАЇНСЬКА ДЕРЖАВНА АКАДЕМІЯ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ
Кафедра ”Обчислювальна техніка та системи управління”

ПРОГРАМУВАННЯ ІНЖЕНЕРНО – ТЕХНІЧНИХ ЗАДАЧ В СЕРЕДОВИЩІ QBASIC

Конспект лекцій з дисципліни
“Обчислювальна техніка і програмування”

частина 3

Харків 2007

Зміст конспекту лекцій розглянуто та рекомендовано до друку на засіданні кафедри “Обчислювальна техніка та системи управління,,
6 березня 2007 р., протокол № 7

Конспект розроблено у відповідності з програмою курсу "Обчислювальна техніка і програмування". До нього включено розділи необхідні студентам для опанування роботою на персональному комп’ютері в середовищі Qbasic, зокрема. основи програмування та прийоми відлагодження основних алгоритмічних структур.

Конспект має чисельні приклади рішення задач або їх фрагменти.

Рекомендується для студентів академії денної форми навчання, що вивчають дисципліни, "Обчислювальна техніка і програмування", "Інформатика та комп’ютерна техніка, "Основи інформаційних технологій", "Комп’ютерна техніка та програмування" всіх спеціальностей та форм навчання.

Укладачі :
професс. І.Г. Філіппенко, доц. В.О. Гончаров, В.С. Меркулов

Рецензент
професс. Г.І. Загарій

ЗМІСТ

1	Середовище QBasic	7
1.1	Загальні відомості про середовище QBasic	7
1.2	Система меню	9
1.3	Введення і редагування програм	9
1.3.1	Редактор QBasic	10
1.4	Запуск програми на виконання і перегляд результатів	12
1.5	Відлагодження програм	12
1.6	Збереження програми на диску	13
1.7	Довідкова система QBasic	13
1.8	Вихід із середовища QBasic	14
2	Елементарні конструкції мови QBasic	15
2.1	Алфавіт	15
2.2	Службові слова і команди	16
3	Структура QBasic – програми	17
4	Типи даних	17
4.1	Константи	17
4.2	Змінні величини	22
4.3	Структури	26
4.4	Функції	27
4.5	Вирази	29
5	Оператори QBasic	34
5.1	Оператори присвоювання	34

5.2 Організація введення-виведення в QBasic – програмах 36
5.2.1 Оператори Data, Read, Restore 36
5.2.2 Оператор введення Input 38
5.2.3 Оператор виведення Print 39
5.3 Приклад програмування лінійного обчислювального процесу мовою QBasic 41
5.4 Програмування задач із використанням операторів передачі управління 44
5.4.1 Оператори умовного переходу 44
5.4.2 Оператори безумовного переходу 49
5.4.3 Приклад програмування розгалуженого обчислювального процесу 51
5.5 Оператори циклів 53
5.5.1 Оператори For … Next 53
5.5.2 Оператори While … Wend 56
5.5.3 Оператори Do …Loop 57
5.5.4 Приклад програмування вкладеного циклічного процесу 59
5.5.5 Приклад вкладеного циклічного процесу з розгалуженням 61
5.6 Деякі корисні команди мови QBasic 63
5.7 Програмування задач обробки масивів даних 64
5.7.1 Приклади програмування завдань обробки одновимірних і двовимірних масивів 66

5.8 Розробка програм з використанням табличних форм 70
5.8.1 Приклад програмування завдання з виведенням результатів у вигляді табличної форми 74
6 Модульне програмування 79
6.1 Типи процедур. Основні визначення 80
6.1.1 Редагування процедур 81
6.2 Процедури – функції або функції користувача (Function) 82
6.3 Підпрограми (Sub) 85
6.3.1 Передача параметрів у процедуру 87
6.3.2 Передача параметра за посиланням 88
6.3.3 Передача параметра за значенням 88
6.3.4 Передача масиву 89
6.4 Підпрограми типу Subroutine і DEF FN – функції 91
6.4.1 Використання підпрограм типу Subroutine – самостійно 91
6.4.2 Використання функцій типу DEF FN (нестандартні функції користувача) 94
7 Файли на магнітних носіях 96
7.1 Файли послідовного доступу (текстові файли) 99
7.2 Файли довільного доступу 105
8 Обробка символьних даних 107
8.1 Одержання ASCII- коду символу й одержання символу, що відповідає ASCII- коду 107
8.2 Введення символів у програму 108

8.3 Підтримка інтерфейсу між програмою й клавіатурою (Inkey$) 109
8.4 Визначення довжини текстового виразу 110
8.5 Вибір підрядка (виділення частини тексту) 110
8.6 Пошук підрядка в символьному виразі 111
8.7 Приведення тексту до різних варіантів написання ... 113
8.8 Формування тексту й видалення початкових або кінцевих проміжків 114
8.9 Перетворення текстових значень у числові, й навпаки 115
9 Графічні можливості QBasic 115
9.1 Графічні режими екрана 115
9.1.1 Кодування графічних зображень. Принципи подання зображень 116
9.2 Графічні примітиви в мові QBasic 121
9.2.1 Оператори Pset і Preset 123
9.2.2 Прямі лінії, відрізки, прямокутники 123
9.2.3 Оператор Circle 126
9.2.4 Використання кольору 127
9.2.5 Дуга, еліпс і сектор 128
9.3 Імітація руху на екрані 129
Список літератури 132

1 СЕРЕДОВИЩЕ QBASIC

1.1 Загальні відомості про середовище QBASIC

Назва мови програмування Basic - це перші літери англійських слів **B**eginner's **A**ll - purpose **S**ymbolic **I**nstruction **C**ode (багатоцільова мова програмування для початківців). Час створення першої версії - 1964 р.

В 1981 р. з'явилася розширена версія Basic-A, що підтримувала текстовий і графічний режими.

Розвитком Basic-A стала версія Quick-Basic, що включала підпрограми і функції з локальними й глобальними змінними, засобами підтримки графіки й звуку, алфавітно-цифрові мітки і т. ін.

Простота граматики й синтаксису, легкість освоєння зробили Basic популярним серед користувачів-непрофесіоналів для математичних і науково-технічних розрахунків. Тому він і досі широко застосовується для навчання основам програмування та для розв’язання відносно простих задач програмування.

Усіченим варіантом Quick-Basic є система програмування QBasic.

QBasic – це інтегрована система програмування, що має власну управляючу оболонку, редактор текстів програм мовою програмування QBasic, засоби пуску, налагодження й прихованої зборки програм, потужний електронний довідник із системи.

Введений за допомогою екранного редактора текст програми необхідно перетворити в машинний код, що складається із двійкових даних і інструкцій процесора. Таке завдання вирішує спеціальна програма - транслятор. Найбільш простий спосіб трансляції полягає в негайному перекладі в машинні коди кожного слова програми - інтерпретація (тлумачення).

Інтерпретатор QBasic може працювати у двох режимах: безпосередньому й режимі виконання програми.

У безпосередньому режимі дія, задана службовим словом, виконується відразу ж після його введення з клавіатури.

При роботі в режимі виконання текст програми спочатку повністю записується операторами мови, далі програма

заноситься до пам’яті і запускається на виконання. У цьому випадку інтерпретатор слово за словом зчитує з пам'яті текст програми, переводить його на мову команд процесора й пропонує готові коди процесору для їхнього виконання.

В інтерпретатора є велика перевага - простота відлагодження програм: виконання програми можна в будь-який момент зупинити і відредагувати її текст. Але є й істотні недоліки: по-перше, програма здатна працювати тільки при наявності в пам'яті самого інтерпретатора (середовища) і, по-друге, програма, що працює з інтерпретатором, робить всі дії дуже повільно. Час іде на розпізнавання слів мови, на пошук відповідних послідовностей команд процесора й на багато чого іншого. Ці недоліки усуваються за допомогою іншого способу трансляції - компіляції (зборка): дії не виконуються негайно, а збираються з наявних у компіляторі стандартних послідовностей у програму в кодах, що не залежать від яких-небудь трансляторів. Головне в компіляторі - зробити програму максимально швидкодіючою.

QBasic має тільки інтерпретатор, а от Quick Basic дає можливість компілювати програми - створювати файли, що виконуються (exe-файли).

Після запуску QBasic на екрані з’являється перше вікно, (рисунок 1.1).

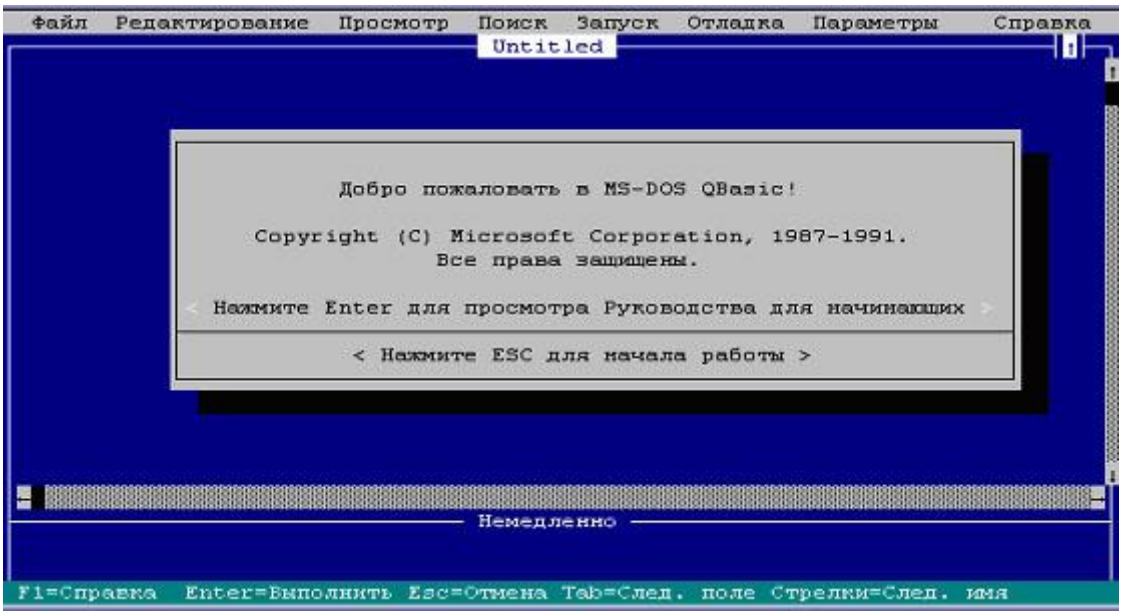


Рисунок 1.1

1.2 Система меню

У верхній частині екрана розташовано головне меню QBASIC, що містить назви основних функцій середовища (таблиця 1.1).

Таблиця 1.1

F1	Виведення підказки для того елемента програми, на який указує курсор
F2	Виведення списку підпрограм. Розбивка екрана на дві частини для одночасного перегляду більших програм
F3	Пошук у тексті
F4	Перегляд екрана виведення
F5	Продовження виконання програми з поточного оператора
F6	Переміщення у вікно швидкого виконання (Immediate)
F7	Виконання програми до поточного положення курсора
F8	Виконання наступного оператора програми (покрокове виконання)
F9	Установка або видалення контрольної точки
F10	Виконання наступного оператора програми (пропускаючи процедуру)
Shift+F1	Перехід у режим допомоги
Shift+F2	Перехід до наступної процедури
Shift+F5	Виконання програми спочатку
Shift+F6	Перемикання у вікно перегляду
Ctrl+F1	Перегляд наступної теми в режимі допомоги
Ctrl+F2	Перехід до попередньої процедури
Ctrl+F10	Перемикання між багатовіконним і повноекранним режимами
Alt+F1	Перегляд попередньої теми в режимі допомоги

1.3 Введення і редагування програм

Головне вікно QBasic розділене на дві основні частини - вікно редагування й вікно безпосереднього виконання.

Вікно редагування

На початку роботи QBasic розташовує курсор у вікні редагування. Це дозволяє вводити текст нової програми (Опція Головного меню **Файл** ⇒ команда **Новый**), завантажувати в це вікно тексти існуючих програм (Файл ⇒ **Открыть** ⇒ клавіша Tab - **вибрати зі списку**), редагувати введені тексти (**Редактирование**), запускати програми на виконання (**Запуск** ⇒ **Запуск**), зберігати їх у файлах на диску (**Файл** ⇒ **Сохранить**).

Вікно безпосереднього виконання

Це вікно розташовується в нижній частині екрана. У ньому можна безпосередньо одержувати результати виконання команди після її введення й натискання клавіші **Enter**. Перехід у це вікно здійснюється після натискання клавіші F6.

1.3.1 Редактор QBASIC

Розглянемо роботу редактора на прикладі простої програми, що має один оператор.

```
PRINT "Hello УКРДАЗТ"
```

Набравши на клавіатурі оператор, натиснемо клавішу **Enter**. Якщо при введенні оператора припустилися помилки, то її можна виправити, використовуючи клавіші переміщення курсора й клавіші видалення символу:

- **Delete** - видаляє символ над курсором;
- **Backspace** - видаляє символ ліворуч від курсора, крім того, клавіші
- **Shift** - перемикач режимів введення великих літер (верхній регістр) і малих (нижній регістр);
- **Caps Lock** - фіксація режиму введення великих літер і скасування його.

Текст програми можна вводити в будь-якому регістрі. Однак після натискання клавіші **Enter** всі ключові (службові) слова програми будуть виведені великими літерами.

Робота із блоками тексту

У боротьбі з помилками й для підвищення швидкості введення програм необхідно освоїти роботу із блоками тексту:

- виділення блока;
- копіювання;
- переміщення;
- видалення.

Тільки виділений текст можна скопіювати, перемістити або знищити. Виділений текст легко відрізняється від навколишнього тексту більш яскравим підсвічуванням.

Існують кілька способів виділення тексту:

- невеликий блок (слово, кілька слів, рядок, кілька рядків, екран) зручніше виділити одночасним натисканням Shift і однієї із клавіш керування курсором ↑, ↓, ←, →, **Home, End, PgUp, PgDn**. Зняти виділення можна натисканням миші на невиділеній частині тексту. Якщо не все виділено або прихоплено зайве й уже відпущено клавішу керування - не відпускайте **Shift** і натискуйте клавішу керування у зворотному напрямку. Комбінації **Ctrl**+**Shift**+← та **Ctrl +Shift**+→ виділяють слово ліворуч і праворуч від курсора;
- комбінації **Shift+Ctrl+Home** і **Shift+Ctrl+End** дозволяють виділити текст програми, відповідно від початку до поточного рядка й від кінця до поточного рядка. Спочатку натискається й утримується **Shift**, а потім послідовно натискаються й відпускаються інші клавіші.

Натисканням клавіші **Del** виділений блок видаляється без запам'ятовування в буфері обміну. Комбінацією клавіш **Ctrl+ Del** виділений блок зберігається в буфері.

Виділений блок можна скопіювати в буфер сполученням **Ctrl + Ins**, після чого сполученням **Shift + Ins** його можна вставляти в будь-які місця будь-яке число раз у будь-які програми;

- все вище назване можна робити за допомогою опцій меню <Редагування>.

1.4 Запуск програми на виконання і перегляд результатів

Для виконання введеної програми треба викликати команду **Запуск** з меню **Запуск** або використати комбінацію клавіш **Shift+F5**.

На екран буде виведений результат виконання програми і повідомлення:
"Press any key to continue" (Для продовження натисніть будь-яку клавішу).

1.5 Відлагодження програм

Транслятор QBASIC виявляє помилки двох типів:

- синтаксичні, які виникають в результаті порушення правил написання конструкцій мови;
- семантичні, пов'язані з неприпустимими значеннями параметрів, неприпустимими діями над параметрами і т. ін.

При виявленні помилки видається відповідне повідомлення на екран монітора і відбувається підсвічування місця її знаходження в тексті.

Крім того, на етапі безпосереднього виконання відслідковуються деякі помилкові ситуації: ділення на нуль, обчислення квадратного кореня з від'ємного числа, переповнення - вихід за праву границю діапазону типу числа і ряд інших. У цих випадках виконання програми або припиняється, або триває з видачею повідомлення про помилку.

Помилки в логіці роботи алгоритму не приводять до зупинки програми. Виявити їх можна тільки аналізом проміжних і остаточних результатів. Для покрокового аналізу проміжних результатів у текст рекомендується включати виведення таких результатів, практикувати виключення фрагментів програми з її роботи шляхом перетворення їх у коментар, використовуючи апостроф.

Переглянути результати можна, натиснувши **F4**.

Зупинити виконання програми можна, натиснувши **Ctrl+Break**, а продовжити натиснувши **F5**.

Для призупинення виконання програми можна поставити у відповідному місці оператор **STOP**.

Значно легше виявити логічні помилки, використовуючи спеціальні клавіші або опцію Головного меню **Отладка**.

Буває корисно виконати програму до рядка, зазначеного курсором - **F7**. Аналогічні результати досягаються установкою в опції **<Отладка>** Контрольная точка остановка программы - **F9** (вона ж - зняття).

Після натискання клавіші **F8** або **F10** (команди **Шаг**, **Процедура на шаг**) - реалізується виконання одного оператора, що дозволяє виконувати програму крок за кроком.

Команда **Трассировка** дозволяє відстежити послідовність виконання операторів програми.

Команда **<Установить следующее значение>** використовується для зміни послідовності виконання програми так, що наступним виконується оператор, на якому встановлений курсор.

1.6 Збереження програми на диску

Для зберігання набраної програми на диску у вигляді файла треба виконати такі дії:

Файл ⇒ команда **Сохранить** ⇒ клавіша **Enter**

Якщо програма ще не має імені, то вона в середовищі QBasic буде позначена як **Untitled**. Введіть ім'я файла, з яким ви бажаєте зберегти свою програму, і натисніть клавішу **Enter.**

Ім'я файла повинне бути унікальним, рекомендується використати для цієї мети послідовність літер та цифр довжиною до 8 символів. Розширення файла **.bas** буде записано автоматично.

1.7 Довідкова система QBASIC

У вікні **СПРАВКА** можна побачити таку інформацію:

- теми довідки;
- основні ключові слова.

Використання довідки QBasic

Щоб одержати довідку по ключовому слову QBasic, встановіть на ньому курсор і натисніть **F1** або праву кнопку миші.

Щоб одержати довідку щодо меню, команд або діалогового вікна QBasic, установіть курсор на темі меню або на кнопці **СПРАВКА** і натисніть **F1**.

Щоб переглянути теми **СПРАВКА** QBasic, натисніть **Alt+З**, для вибору теми - натисніть букву, виділену підсвічуванням.
Щоб перемістити курсор у вікно довідки, натисніть **Shift+F6**.

Щоб переглянути всю довідкову інформацію, використайте **PgDn** або **PgUp**.

Щоб скопіювати інформацію з **СПРАВКА** (приклади програм) у вікно редагування, використайте команди з меню **Редактирование** QBasic.

Щоб закрити вікно довідки, натисніть **Esc**.

Щоб перемістити курсор до теми **СПРАВКА**, використовують клавішу табуляції або натискають першу літеру теми. Щоб побачити інформацію з теми або ключового слова, треба встановити курсор на темі або ключовому слові й натиснути **F1** або праву кнопку миші.

QBasic зберігає останні 20 тем довідки, які були переглянуті. Щоб переглянути попередні теми, треба натиснути **Alt+F1** або натиснути кілька разів кнопку миші на кнопці <**Назад**>.

1.8 Вихід із середовища QBASIC

Для виходу із середовища QBASIC необхідно виконати послідовно такі дії:

Головне меню: опція **Файл** ⇒ команда **Выход** ⇒ клавіша **Enter**.

2 ЕЛЕМЕНТАРНІ КОНСТРУКЦІЇ МОВИ BASIC

2.1 Алфавіт

Алфавіт алгоритмічної мови - це набір припустимих знаків (символів), які можна використати для запису програм.

Всі символи можна розділити на групи:

- латинські літери (26 літер) (A-Z, a-z);
- російські літери (кирилиця) (А-Я, а-я);
- цифри (0-9).

Розділові символи (обмежники)

. десяткова крапка	, кома
: двокрапка	" лапки
( ) круглі дужки	_ пропуск
; крапка з комою	' одинарні лапки (апостроф)

Спеціальні знаки

$ знак грошової одиниці	@ комерційне ЕТ
& комерційне І (амперсанд)	_ символ підкреслення
# номер	% знак відсотка
? ! {} і ін.	

Знаки арифметичних операцій

* множення;	/ ділення (прямий слеш);
^ піднесення до степеня;	+ додавання;
- віднімання;	
MOD - залишок від цілочислового ділення чисел	\ цілочислове ділення (зворотний слеш) - повертає кількість входжень другого цілого числа в перше

Знаки операцій відношення:

Основні	Похідні операції
> знак більше ніж;	>= знак більше або дорівнює
< знак менше ніж;	<= знак менше або дорівнює
= знак дорівнює	<> знак не дорівнює

2.2 Службові слова і команди

Програма мови програмування високого рівня складається з окремих лексичних одиниць - команд (операторів).

Оператор - основна мінімальна логічно завершена конструкція мови, що містить ім'я оператора і його параметри. Оператор визначає об'єкти, тип і послідовність дій над ними. Для написання операторів і найменувань різноманітних об'єктів використовуються латинські літери.

Російські літери використовуються тільки для запису символьних констант і коментарів.

QBasic не робить різниці між великими і малими латинськими буквами в службових словах, позначеннях змінних і інших об'єктів.

За призначенням оператори розділяють на такі групи:

- описові оператори - для опису даних, типів змінних, розмірів масивів, нестандартних функцій і т.п.;
- оператори присвоювання - для завдання початкового значення або зміни поточного значення змінної;
- оператори введення-виведення даних - для введення і виведення інформації;
- оператори управління процесом обробки інформації: оператори переходу, організації розгалужень і циклів і т.п.;
- інші оператори, що забезпечують додаткові можливості: оператори для роботи з файлами даних, оператори для графічних побудов, одержання звукових ефектів і т.п.

Оператори складаються з нероздільних елементів мови: службових слів, чисел, символів операцій і т.п. Мова QBasic налічує більше 200 типів операторів, наприклад

DATA - дані	NEXT - наступний
DEF FN – визначення функції	ON - при
DIM - розмір	OPTION BASE – задати базу
END - кінець	PLAY - грати
FOR-TO-STEP – для – до - крок	PRINT - друкувати
GOSUB – перейти до підпрограми	READ - читати
GOTO – перейти до	REM - пояснення
IF-THEN – якщо-то	RESTORE - обновити
INPUT - ввести	RETURN - повернутися
LET- нехай	STOP - зупинитися

3 СТРУКТУРА QBASIC – ПРОГРАМИ

Програма складається з програмних рядків довжиною до 255 символів. В одному рядку можна розмістити один або кілька операторів, розділених двокрапкою.

Рядок може мати номер - ціле число, що трактується системою як мітка. Мітка може бути і символьною, наприклад, Label1. Після символьної мітки ставиться двокрапка.

4 ТИПИ ДАНИХ

Залежно від того, чи змінюють свої значення дані в процесі виконання програми, розділяють постійні величини (константи) і змінні.

4.5 Константи

Константи - це об'єкти, значення яких не можуть бути змінені в процесі виконання програми.

Константи за типом поділяються на числові, логічні і символьні.

а) числові константи

За відсутністю або наявністю дробової частини і за способом зберігання в пам'яті ПК діляться на цілі і речовинні (дійсні).

Для подання чисел у пам'яті комп'ютера використаються 2 формати:

- з фіксованою крапкою;
- із плаваючою крапкою.

У форматі з фіксованою крапкою зберігаються тільки цілі числа, а у форматі із плаваючою крапкою - дійсні числа (цілі і дробові).

Діапазон значень залежить від розміру комірок пам'яті, які використовуються для їхнього зберігання.

В k-розрядній комірці може зберігатися 2^k різних значень цілих чисел.

З урахуванням знака числа в 16 - розрядній комірці (2 байти) зберігаються цілі числа в діапазоні від -2^{k-1} (-32768) до $2^{k-1} - 1$(32767).

Двійкові розряди в комірці нумеруються від 0 до k. Старший, k-й розряд, у внутрішньому поданні будь-якого додатного числа - 0, від'ємного - 1.

Формат із плаваючою крапкою використовує подання дійсного числа **R** у вигляді добутку мантиси **m** на основу системи числення **n** у деякому цілому ступені **p**, що називають порядком:

$$\mathbf{R=m*n^p}$$

Подання в цій формі неоднозначне

$$25.324=2.5324*10^1=0.0025324*10^4=2532.4*10^{-2}$$

В ПК використовується нормалізоване подання числа у формі із плаваючою крапкою. При цьому мантиса повинна задовольняти умову 0.1<=m<=1, тобто повинна бути менше 1 і перша значуща цифра – не дорівнювати „0".

У пам'яті ПК мантиса уявляється як ціле число, що містить тільки значущі цифри (0 цілих і кома не зберігаються).

В 4-байтовій комірці:

- знак числа (1 розряд): 0- плюс, 1 - мінус;
- порядок (7 розрядів): числа від 0000000 до 1111111 (від 0 до 127 у десятковій системі числення; усього 128 значень). Порядок очевидно може бути як додатним, так і від'ємним (від -64 до 63). Машинний порядок (Мр) зміщений щодо математичного (р) і приймає тільки додатні значення. Зсув вибирається так, щоб мінімальному математичному значенню відповідав 0.

$$Мр=р+64$$

- мантиса (24 разр.).

+/ маш.порядок	МАН	ТИ	СА
1 байт	**2 байт**	**3 байт**	**4 байт**

Цілі константи - послідовність цифр і спеціальних символів зі знаком або без нього.

За способом завдання в програмах розрізняють:

- цілі константи в десятковому поданні:

- одинарної точності (тип INTEGER) - належать діапазону від -32768 до 32767 і зображуються в ПК 2-байтовими двійковими числами. У програмі запис цих констант завершується символом % (цілі короткі).

Приклад: 123%, -89%;

- подвійної точності (тип LONG) - належать діапазону від 2147483648 до 2147483647 і зображуються в ПК 4-байтовими двійковими числами. Запис цілочислових констант цього типу завершується символом & (цілі довгі).

Приклад: 123897890%, -78989%;

- цілі константи у вісімковому поданні:

- послідовність вісімкових цифр без знака, яким передує &

Приклад: &123, &2;

- цілі константи в шістнадцятковому поданні:

- послідовність шістнадцяткових цифр без знака, яким передує &H.

Приклад: &H123, &HA56E.

Для запису дійсних констант використовують дві форми: природну та експоненційну.

Дійсна константа в природній формі - послідовність десяткових цифр зі знаком і крапкою, що розділяє цілу і дробову частини.

Приклад: 0.65; -5.34; +34.09.

Дійсна константа в експоненційній (показовій) формі (з плаваючою крапкою) має вигляд:

$$\pm mE \pm p$$

Мантиса записується аналогічно запису відповідної константи в природній формі;

E (D - для подвійної точності) - символ ознаки порядку; значення порядку записується зі знаком або без нього.

Приклад: 0.56E+12, 6. 345D-2

0.56E+12 або 5.6E+11 відповідає $0.56*10^{12}$

Ця форма зручна для запису дуже великих або дуже маленьких чисел.

Знак плюс можна опускати. Знак мантиси визначає знак числа. Порядок може бути тільки цілим числом: він визначає положення десяткової крапки.

Кажуть "крапка плаває", тому що розміщення десяткової крапки в мантисі явно не вказує на величину числа.

За обсягом зайнятої пам'яті дійсні константи поділяються на:

- дійсні константи одинарної точності (тип SINGLE):

- діапазон для додатних чисел: від 2.8E-45 до 3.4E+38; для від'ємних – від -3.4E+38 до -2.8E-45 ;
- містить не більше 7 цифр; в експоненційній формі записується з літерою E. Припустимо запис констант такого типу закінчувати символом '!';
- для їх зберігання використовуються 4-байтові поля.

Приклад: 45.23, -.6,1.3E-04,29.5!;

- дійсні константи подвійної точності (тип DOUBLE):

- діапазон для додатних: від 4.9D-324 до 1.8D+308, для від'ємних - від -1.8D+308 до -4.9D-324;
- містить не більше 18 цифр; в експоненційній формі записується з літерою D. Запис таких констант закінчується символом #;
- для їхнього зберігання використовуються 8-байтові поля.

Приклад: 234568.61, -1.0965434D12, 3241.6#;

б) логічні константи

Вони можуть мати тільки два фіксованих значення TRUE (істина) або FALSE (хибність);

в) символьні (рядкові, текстові) константи (тип STRING)

Укладена в лапки послідовність символів (літер, цифр, спеціальних знаків - усього, що можна набрати на клавіатурі). Максимальна довжина символьної константи - 32767 символів. Якщо заздалегідь відомо, що число символів у константі не перевищує **n**, то тип задається, як STRING* n.

Приклад: " аудиторія 2. 123 ", "=Ю?*+"

Константам можна присвоювати імена за допомогою команди оголошення констант:

CONST <ім'я змінної 1>=<вираз1>
.....
<ім'я змінної n>=<вираз n>

Обчислюється значення виразу, і результат присвоюється константі. Використане у форматі позначення < > означає обов'язковий елемент конструкції.

Приклад: CONST PI=3.1415926: A=" програмування "

4.5 Змінні величини

Змінні - це об'єкти, значення яких заздалегідь не визначені і можуть змінюватися в процесі виконання програми.

Змінним можна присвоювати значення констант, інших змінних, результатів обчислень або вихідних даних.

Кожна змінна позначається унікальним ім'ям - **ідентифікатором**. Необхідно чітко уявляти, що кожному ідентифікатору в пам'яті ПК відповідає цілком визначена адреса комірки, і зміна значення змінної зводиться до зміни вмісту відповідної області пам'яті.

Ідентифікатор може містити від 1 до 40 алфавітно-цифрових символів. Першим символом повинна бути літера. В ідентифікаторах повинні використовуватися тільки латинські літери. Допускається використання крапок. Ім'я не повинно збігатися з яким-небудь із зарезервованих службових слів QBASIC (DATA, END, THEN і т.д.).

Приклади ідентифікаторів: A, b12, Beta C_mod, GammaEnd.of.text.

Імена змінних визначають їхній тип, а також точність числових змінних.

Змінні називаються числовими, якщо вони можуть приймати тільки числові значення.

Як і константи, змінні за типом діляться на числові, логічні і символьні.

Тип змінних в програмі можна задати чотирма способами:

1) принцип умовчання;
2) спеціальні символи;
3) команди опису типу DEF;
4) команди оголошення змінних DIM.

Принцип умовчання

Якщо в програмі немає опису типу, то будь-які змінні вважаються дійсними. Однак символьні дані потрібно описувати завжди.

Явне завдання типу змінної - за допомогою спеціального символу (%, &, !, #, $), що завершує ім'я змінної.

При позначенні змінної 2-байтного цілочислового типу використовується символ % наприкінці змінної.

Приклад: A%, X1%.

Така змінна може набувати будь-якого значення, що припустимо для цілої константи нормальної довжини. Для позначення змінної 4-байтного цілочислового типу використовують символ &.

Приклад: A&, X1&.

Дійсна змінна одинарної точності (4 байти) зазвичай позначається ім'ям без спеціальних символів (іноді запис ідентифікатора закінчується символом !).

Приклад: B2, Aud!.

Для позначення змінної дійсного типу подвійної точності (8 байтів) використовується символ # наприкінці ідентифікатора.

Приклад: B2#, Aud#.

Логічна змінна - символічно позначена логічна величина, що може набувати тільки значення TRUE(Істина) або FALSE (Хибність).

Символьна змінна закінчується символом $.

Приклад: змінна Y$ може містити дані символьного типу.

Неявне завдання типу змінної (команди опису типу) - за приналежністю першого символу ідентифікатора заданому діапазону літер.

Воно здійснюється за допомогою операторів:

- DEFINT Двобайтовий цілочисловий;
- DEFLNG Чотирибайтовий цілочисловий;
- DEFSNG Чотирибайтовий дійсний;
- DEFDBL Восьмибайтовий дійсний;
- DEFSTR Символьний тип.

Всі ці оператори мають однаковий формат:

DEF<ТИП> <ДІАПАЗОН_ЛІТЕР>

де <ДІАПАЗОН_ЛІТЕР> - літера або діапазон літер латинського алфавіту, з яких може починатися ім'я змінної відповідного типу.

Приклад

```
DEFSTR A-D
A = "ЦЕ РЯДОК"
B = "СИМВОЛІВ"
PRINT A + B
```

У результаті буде виведено на екран повідомлення:

ЦЕ РЯДОК СИМВОЛІВ

Операція A+B в даному випадку називається конкатенацією.

DEFINT A, P-S, Z

всі змінні, що починаються з A,H,Q,R,S,Z вважаються 2-байтними цілого типу.

На змінні, описані явним способом, ідея операторів неявного оголошення типів змінних не поширюється.

Приклад

У програмі поряд з оператором DEFSNG C-F можуть існувати і цілочислові змінні з іменами, COLONKA%, DLINA& і т.п.

Неявний опис типу поширюється тільки на ті змінні, імена яких не завершуються зазначеними вище спеціальними символами.

Використання команди оголошення змінних DIM

Традиційно цей оператор використовується для оголошення масивів. Однак він може бути використаний і для скалярних змінних в такому вигляді:

DIM<список 1 змінних> AS <тип1>

.....

<список n змінних> AS <тип n>

Приклад: Цілі змінні A,B, дійсні C,D і текстові F,H можна описати так:

```
DIM  A,B AS INTEGER, C,D AS DOUBLE
DIM F,H AS STRING*10
```

Перед виконанням програми всім числовим змінним автоматично присвоюються нульові значення. Довжина кожної змінної символьного типу встановлюється також нульовою, тобто всім символьним змінним присвоюються значення "порожнє".

Поряд з константами і змінними (скалярними) у програмах можуть використовуватися масиви і структури, тобто групи змінних, на які посилаються за одним загальним ім’ям. Окремі елементи таких груп використовуються в програмах за тими ж правилами, що і прості змінні.

4.5 Структури

Запис - структурований набір даних, призначений для зберігання в оперативній пам'яті та обробки даних, що складається з фіксованого числа компонентів даних різних типів (полів). Іноді записи називають структурою. Зрозуміло, типи компонентів можуть бути й однаковими.

Формат
TYPE < ім'я запису>
< ім'я поля 1> AS < тип поля 1>;
.....
< ім'я поля n> AS < тип поля n>;
END TYPE

Імена записам і полям дає користувач. Типом поля може бути кожний зі стандартних типів (INTEGER, LONG, SINGLE, DOUBLE, STRING*n) або інший запис. Типом поля не може бути тип невідомої довжини, тобто STRING.

Полями запису, крім стандартних числових типів даних і строкових змінних заданої довжини, можуть бути типи користувача. Для опису складних структур у записах в якості поля можна використати інші записи. Виходить структура, що нагадує своїм зовнішнім виглядом матрьошку.

Приклад

Запис анкетних даних студентів: прізвище, ім'я, дата народження і середній бал, можна описати так:

```
TYPE grupa
namel AS STRING*20
surname AS STRING*15
birthday AS TYPE
year AS INTEGER
month AS INTEGER
day AS INTEGER
END TYPE
sball AS SINGLE
END TYPE
```

Змінну типу **ЗАПИС** оголошують за допомогою команди

DIM < ім'я змінної> AS < ім'я запису>

Із записів можна створити масив.

Приклад: оголосити змінну **a** і масив записів **stud** можна так:

DIM a AS grupa.
DIM stud(20) AS grupa.

Доступ до конкретного поля запису дає складене ім'я виду **< ім'я змінної>.< ім'я поля>**

Приклад: У програмі змінним **a** і **stud(1)** можна задати такі значення

a.name1="Петро"
stud(1).surname="Іванович": stud(1).birtday.month=5

4.4 Функції

Як операнди у програмах поряд з константами, змінними та обумовленими користувачем функціями можна застосовувати функції, визначені в самій мові (вбудовані функції). На використання більшості з них не накладаються ніякі обмеження.

Для зручності всі функції поділяють на групи, зв'язані не тільки з типом функції, але і з її застосуванням і типом аргументів:

1) математичні функції;
2) числові функції символьних аргументів;
3) символьні функції;
4) функції введення, виведення і доступу до пам'яті;
5) системні змінні.

Математичні функції

У математичних функцій аргументами є арифметичні вирази, а значеннями - числа. Тригонометричні функції використовують радіанний вимір кутів.

Стандартні математичні функції:

SIN(X)	**синус X**	**sinX, аргумент в радіанах**
COS(X)	**обчислення косинуса X**	***cosX,* аргумент в радіанах**
TAN(X)	**тангенс X**	**tgX, аргумент в радіанах**
ATN(X)	**арктангенс X**	**ArctgX, $-\pi/2<X<\pi/2$,**
ABS(X)	**модуль X**	**|X|**
SQR(X)	**квадратний корінь X**	**$\sqrt{X}$, X>0**
EXP(X)	**експонента X**	**e^X**
LOG(X)	**натуральний логарифм X**	**lnX, X>0**
CDBL(X)	**перетворення в число подвійної точності**	
CINT(X)	**округлення до цілого**	**CINT(15.5)=16 CINT(-6.7)=-7**
CLNG(X)	**(довгого цілого) значення**	**CINT(-6.2)=-6**
CSNG(X)	**перетворення в число простої точності**	
FIX(X)	**усікання до цілого**	**FIX(-5.6)=-5**
	(відкидання цілої частини)	**FIX(24.07)=24.00**
INT(X)	**знаходження найбільшого**	**INT(15.5)=15 INT(-6.2)=-7**
	цілого, що не перевищує X	**INT(-6.7)=-7**
RND(X)	**генерація псевдовипадкових чисел від 0 до 1**	
SGN(X)	**знак числа X:**	**SGN(X) = -1, якщо X < 0**
		SGN(0) = 0, якщо X = 0
		SGN(X) = +1, якщо X > 0

Числові функції символьних аргументів

Значеннями цих функцій є числа, а аргументами - символьні вирази або функції.

ASC, CVI, CVS, CVD, INSTR, LEN, VAL і ін.

Символьні функції

Значеннями цих функцій є ланцюжки символів.

CHR$, LEFT$, RIGHT$, MID$, STRING$, LTRIM$ і ін.

Всі функції 2-ї і 3-ї груп в основному використовуються для перетворення даних, що є результатом роботи операторів введення-виведення та обробки символьної інформації.

Функції введення - виведення і доступу до пам'яті

Сюди входять функції, пов'язані із введенням - виведенням, а також такі, що дозволяють одержати інформацію про стан різних пристроїв і транслятора QBASIC.

CSRLIN - повертає номер рядка поточного положення курсора.

FRE - обсяг вільної частини пам'яті робочої області QBASIC.

POINT - координати точки екрана.

PEEK - вміст байта пам'яті і ін.

Системні змінні

TIME$ - системний час DATE$ - системна дата та ін.

Повністю список всіх функцій можна знайти у довідці QBasic і системній документації.

4.5 Вирази

Вираз - це послідовний запис констант, змінних, індексних змінних (елементів масивів), функцій або будь-яких їхніх комбінацій, утворений за допомогою знаків арифметичних і логічних операцій, операцій відношень, операцій над рядками символів.

Розрізняють арифметичні, логічні і символьні вирази.

Арифметичні вирази

Арифметичні вирази відповідають загальноприйнятим алгебраїчним виразам. До них можуть входити константи, змінні, функції, з'єднані знаками арифметичних операцій. Результатом арифметичного виразу є число. Число або змінна також вважаються арифметичним виразом. Для позначення арифметичних операцій використовуються знаки

+	-	*	/	\	MOD

Операція \ означає ділення націло (дробова частина відкидається)

Приклад: 17\2=8.

Операція MOD означає обчислення залишку (ділення по модулю)

Приклад: 25MOD8=1.

Якщо операнди цих операцій дійсні, то вони попередньо округляються.

Приклади:

$$\frac{ab}{c} = A*B/C$$

$$\frac{x+2}{c+d} = (X+2)/(C+D)$$

$$\frac{a-\sqrt{d}}{2x} = (A-SQR(D))/(2*X)$$

$$\sin^2 x - \cos x^2 = SIN(X)\text{^}2 - COS(X\text{^}2)$$

$$3a^2 + \frac{b}{4} + \frac{7}{1-a} = 3*A\text{^}2 + B/4 + 7/(1-A)$$

$$\frac{y_j y_{j+1}}{\sqrt{j}} = Y(J)*Y(J+1)/SQR(J)$$

Пріоритет виконання операцій

Всі операції в арифметичному виразі виконуються в певній послідовності: порядок їхнього виконання задається правилами пріоритету

1) обчислення числових функцій;
2) унарний мінус;
3) піднесення до степеня;
4) множення / ділення;
5) ділення націло;

6) обчислення залишку;
7) додавання / віднімання.

Послідовність операцій одного пріоритету виконується зліва направо. Для зміни цього порядку можна використати круглі дужки. Вирази , що знаходяться в дужках, обчислюється в першу чергу.

Приклад: A =8 B =4 C =2

A+B/C^2 =9 (A+B)/C^2 - =3
(A+B/C)^2 =100 A+(B/C)^2 - =12

Якщо у виразі кілька операцій мають однаковий пріоритет, то вони виконуються один по одному зліва направо. Виключенням є піднесення до степеня

X^{Y^Z} =X^(Y^Z).

Для обчислення кореня довільного степеня використовується еквівалентний вираз

Приклад: $\sqrt[3]{\cos(x)}$ =COS(X)^(1/3).

Логарифм за довільною основою обчислюється за формулою

$$\log_a b = \frac{\ln b}{\ln a}.$$

Приклад: $\log_{10}(x+1)$ =LOG(X+1)/LOG(10).

Зважаючи на те, що LOG(10)=2.3..., а 1/ LOG(10)=0.434, можна записати

$$\log_{10}(x+1) = 0.434 * \log(x+1).$$

Не можна ставити два знаки арифметичних операцій підряд – треба використовувати дужки, наприклад, $\frac{a}{-b}$ =>A/(-B).

Від'ємні значення підносити до дробового степіня забороняється, тому що $x^a = e^{a \ln x}$.

У зв'язку з наближеним поданням дійсних чисел в ПК рівність X/Y*Y=X не виконується.

Відзначимо випадок так званих особливих ситуацій при обчисленні арифметичних виразів:

- ділення на нуль - видається відповідне повідомлення; обчислення тривають, а в результаті виходить 1.701412E+38 машинна нескінченність;
- переповнення - видається відповідне повідомлення, і виконання програми припиняється.

Логічні вирази

Логічні вирази складаються з логічних операцій і логічних відношень. В них присутні два операнди, які QBasic розглядає як шістнадцяткові бінарні ланцюжки, над якими побітно зліва направо виконуються дії за допомогою таких операторів:

NOT	заперечення (НІ)	*XOR*	АБО, що виключає
AND	кон'юнкція (І)	*EQV*	еквівалентність
OR	диз'юнкція (АБО)	*IMP*	імплікація

Всі вони повертають значення "істина" (не-нуль) або "хибність" (нуль), що використовуються при ухваленні рішення про подальший хід обчислювального процесу.

Приклад: 63 AND 16=16 4 OR 2=6

111111 AND 10000=010000 100 OR 010=110

Порядок виконання логічних операцій задається пріоритетом (NOT, AND, OR) і круглими дужками.

Логічні відношення

Операції відношення порівнюють два числових або символьних значення і якщо умова виконується, то результат -1, інакше - 0.

Приклад: b2-4ac>0 B^2-4*A*C>0 i№j I<>J рядок a =рядку b A$=B$

46=41 - 0 10<8 - 0 15<=20 - 1 15<>20 - 1 2>1 -1 3>=2 – 1

Якщо порівняння робиться над числами, то припустиме порівняння цілого і дійсного типів.

Символьні величини можна порівнювати тільки з символьними, при цьому порівняння здійснюється посимвольно зліва направо із урахуванням кінцевих пропусків. Більш короткий рядок вважається меншим. Порівняння засноване на відносних значеннях їх шістнадцяткових кодів. При розбіжності символів більшим вважається рядок, що містить символ з більшим кодом.

Приклад: "AVZ">"AGN"; код"V"=086; код"G"=071.

Якщо логічні вирази містять логічні відношення і логічні операції, то спочатку виконуються логічні відношення, а потім логічні операції відповідно до пріоритету.

Приклад: -4<= X <=4 (-4<=X) AND (X<=4)
X=0 або X=1 (X=0) OR (X=1)

Символьні вирази

Складаються із символьних констант, змінних, функцій або будь-яких їхніх комбінацій, розділених знаками логічних відношень або спеціальних символьних операцій.

Спеціальна символьна операція називається **конкатенація.** Вона позначає об'єднання зазначених значень і позначається символом " + ".

Приклад:

A$="студент" B$=" гр. 5-І-В", тоді A$+B$="студент гр. 5-І-В"

Довжина рядка, що є результатом, не повинна перевищувати 255 символів.

5 ОПЕРАТОРИ QBASIC

5.1 Оператори присвоювання

Задати значення змінної можна за допомогою оператора присвоєння. Але цей найбільш простий спосіб має два істотних недоліки:

- при великій кількості вхідних даних необхідно записати в програмі відповідну кількість операторів присвоєння;
- для рішення завдання з іншими вхідними даними потрібно змінити всі оператори присвоєння.

Формат оператора присвоєння:

[<Рядок/мітка >] <ім'я_змінної> = <вираз>

Цей оператор присвоює значення, задане(обчислене) в правій частині, змінній, записаній в лівій частині.

Використане у форматі позначення [ ] означає необов'язковий елемент конструкції.

Рядок - ціле число від 1 до 32767(номер рядка) або набір символів, що починається з літери, може мати до 40 літер і/або цифр і повинен закінчуватися двокрапкою (мітка). Позначаються тільки рядки, до яких у програмі передбачена передача управління, інші позначати не потрібно.

Приклад: A1 = 222. D=B*B-4*A*C X(I,J)=X(I,J)/X2

Z$="Привіт" I=I+1 : Z=SI(I+X) : D=LOG(A/X)

У правій і лівій частинах повинні бути дані одного типу (числового і числового, символьного і символьного, логічного і логічного).

До моменту виконання оператора значення змінних у правій частині повинні бути визначені.

Якщо права частина - арифметичний вираз, то в ньому можуть вживатися величини різної точності; при цьому числа однієї точності перетворяться в числа іншої точності за такими правилами:

1 Значення, що присвоюють змінній, перетворяться до точності цієї змінної:

```
A%=23.42          [A%]=23.
```

2 При перетворенні до меншої точності відбувається округлення числа:

```
C=55.8834567# [C]=55.88346
A%=2.5            [A%]=3.
```

3 Перетворення числа до більшої точності може змінити його зовнішнє подання

```
A=2.04            B#=A
[B#]=2.039999961853027.
```

4 При обчисленні виразів операнди двомісцевих арифметичних операцій перетворяться до точності операнда з більш високою точністю, а результат - до точності змінної, якій він присвоюється.

Оператор присвоювання не має властивості алгебраїчної рівності, хоча записується подібно.

Приклад

Запис X=X+D в алгебрі недоцільний. У програмуванні це означає, що до значення X додається значення D і отримане значення записується в ту ж саму комірку X.

5.2 Організація введення-виведення в QBASIC –програмах

5.2.1 Оператори DATA, READ, RESTORE

Ці оператори використовуються тоді, коли в програмі багато вхідних даних і деякі з них необхідно часто змінювати при розрахунку різних варіантів (наприклад, у процесі відлагодження програми).

Формат:
READ <список змінних>
DATA <список констант>
RESTORE [<n>]
<список змінних>=<змінна 1>, <змінна 2>,..., <змінна N>
<список констант>=<константа 1>, < константа 2>,..., < константа N>

Значення змінних, що задаються оператором **READ**, розміщуються в списку констант оператора **DATA**, і при цьому формується спеціально відведена область пам'яті - блок даних.

Блок даних можна уявити як одновимірний масив, розміщений в оперативній пам'яті. Як тільки в програмі зустрічається оператор **DATA**, то всі константи, перераховані в ньому, переносяться до блока даних. Порядок їхнього розміщення в блоці даних у точності повторює їх порядок в списку констант оператора **DATA**. Константи можуть бути величинами будь-яких типів, і при цьому в одному списку можуть зустрічатися дані різних типів.

У програмі може бути кілька операторів **DATA**. Тоді наступна послідовність заноситься в блок даних слідом за попередньою.

Оператор **DATA** тільки зберігає дані, він є невиконуваним і може перебувати в будь-якому місці програми. У нумерованому рядку він повинний бути єдиним.

Розміщати всі оператори **DATA** рекомендується в одному місці програми - на початку або наприкінці - це полегшує її відлагодження.

Символьні константи записуються в лапках.

Оператор **READ** призначений для читання значень із блока даних: у процесі виконання програми. Він послідовно "витягає" чергове значення із блока даних і присвоює його відповідній змінній.

READ може бути записаний у будь-якому місці програми, в будь-якому рядку з іншими операторами.

READ і **DATA** у програмі не обов'язково повинні бути парними: можуть бути присутні декілька **DATA** і один **READ,** і навпаки.

Необхідно дотримуватись таких умов:

- типи змінних в **READ** повинні бути сумісні з типами значень, що присвоюють їм;
- загальна кількість змінних у всіх операторах **READ** не повинна перевищувати сумарного числа значень в операторах **DATA**.

Блок даних, за необхідності, можна використати повторно.

Оператор **RESTORE** (відновлення блока даних) дозволяє операторові **READ** ще раз прочитати значення в зазначеному операторі **DATA**.

n - мітка або номер рядка оператора **DATA**. Без операнда n цей оператор переводить умовний покажчик на елемент блока даних у початковий стан, при цьому стають доступними значення першого оператора **DATA**. Якщо операнд **n** заданий, то доступними стають дані зазначеного оператора **DATA**.

Приклад

```
DATA 123, "аудиторія"
READ A, B$
PRINT "Це "; A; "-я "; B$
' ще раз
RESTORE
READ A, B$
PRINT "Це "; A; "-я "; B$
```

Результат: Це 123-я аудиторія
Це 123-я аудиторія

5.2.2 Оператор введення **INPUT**

Його використовують для зчитування вхідних даних з клавіатури.

Формат оператора:

INPUT ["<запрошення>";/,] <список_введення>

де <запрошення> - (підказка) необов'язковий рядок, що відображається на екрані перед введенням користувачем даних; він використовується для виведення на екран відповідних повідомлень, про те, яку інформацію треба вводити. Крапка з комою після запрошення, додає знак "?" у рядок запрошення; кома - скасовує його.

<список_введення> - одна або декілька змінних, розділених комами, у яких зберігаються дані, введені з клавіатури.

Якщо даних декілька, вони вводяться через "," у відповідності до списку змінних. Змінні набувають введених значень після натискання клавіші **ENTER**, до цього їх можна редагувати у випадку помилки, а інакше - вводити заново.

Число даних, що вводять, повинне відповідати числу змінних у списку, інакше видається повідомлення про помилку ("повторить ввод" і т.п.)

Приклад:

Необхідно, щоб A=4.5 B==-3.4E7 C$=березень

INPUT " Уведіть A, B, C$ ", A, B, C$

INPUT " Уведіть A, B ", A, B

INPUT " Уведіть C$ " C$

Уведіть A, B, C$ 4.5,3.4E7,березень

Уведіть A, B 4.5,3.4E7

Уведіть C$ березень

5.2.3 Оператор виведення **PRINT**

Використовується для виведення інформації на екран у стандартному форматі. При виведенні інформації до друку використовується оператор **LPRINT**.

Формат оператора:

PRINT [<список виведення>]
LPRINT [<список виведення>]
<список виведення> = <вираз1>[; /,] [< вираз2>; /,]...[< вираз n>;/,]
вираз - вираз будь-якого типу, що складає список виведення.

Якщо вирази у списку розділені "," - виведення здійснюється в зонному форматі, при якому рядок умовно розділяється на 5 зон по 14 позицій у кожній і одну зону з 10 позицій. Кожне виведене значення розташовується у своїй зоні. Якщо остання зона в рядку заповнена, то виведення триває з першої зони наступного рядка. При виведенні чисел 1-а позиція кожної зони відводиться під знак числа. Дві коми підряд означають пропуск зони.

Якщо вирази розділені ";" - виведення здійснюється в компактному форматі (впритул). У цьому форматі числа виводяться через 2 або 1 позицію (якщо з "-"). Символьні значення при виведенні розташовуються підряд. Як роздільник поряд з "," може бути пропуск. Якщо довжина значення перевищує залишок рядка, виведення починається з початку наступного рядка.

Число стовпців (довжину рядка), як і число рядків можна задати в операторі WIDTH.

Приклад

```
A=1: B=2: C=-3: C$="САТУРН"
PRINT A,B,C,,C$
PRINT A;B C;C$
```

```
2               16              29              43              57
1               2               -3                              САТУРН
```

```
2       5       7       10
1       2       -3      САТУРН
```

Відсутність "," або ";" викликає перехід на новий рядок і навпаки їхня наявність блокує перехід.

Приклад

```
A=1:B=2:C=3
PRINT A,B,                      PRINT A,B
PRINT  C                        PRINT  C
1               2               3       1               2
                                        3
```

Виведення можна здійснювати з довільної позиції, розміщеної праворуч поточного положення курсора шляхом використання функції TAB.

TAB(<стовпець)

де <стовпець - номер стовпця нової позиції для друку (від початку рядка).

Приклад: PRINT TAB(7), "X="; X, TAB(20); "Y="; Y

Кома або крапка з комою після функції TAB(...) не викличе ніякого додаткового переміщення (кома автоматично перетвориться в крапку з комою). Якщо зазначено номер позиції (<стовпець>), більший ніж довжина рядка виведення, то

відбувається перехід на новий рядок; якщо ж (<стовпець>) менший за позицію курсора, то повернення назад не відбувається, а виведення здійснюється в зазначеній позиції наступного рядка. Якщо номер позиції заданий арифметичним виразом, то береться ціла частина цього виразу.

Функція **SPC(n)** означає виведення n пропусків - її зручно використати, якщо потрібно розділити виведені елементи декількома пропусками.

Приклад: PRINT X;SPC(6);SIN(X).

Позиціювання курсора можна також виконати за допомогою оператора **LOCATE:**

Формат оператора:

LOCATE <рядок>, <стовпець> [,<курсор>]

де <рядок> і <стовпець> - номер рядка і стовпця, куди переміщується курсор.

<курсор> - визначає режим відображення курсора: 0 - видимий, 1 - невидимий.

Приклад: **LOCATE** 10, 10: PRINT "Приклад"

Часто виникає необхідність оформити результати обчислень у вигляді таблиць, розташувавши їх у певних позиціях, опустити зайві знаки, додати якісь символи і т.п. В цьому випадку використовують оператор форматного виведення **PRINT USING**.

5.3 Приклад програмування лінійного обчислювального процесу мовою QBASIC

Необхідно обчислити значення трьох функцій Y, Z та Q за такими формулами:

$$Y = \frac{2{,}7 + \sqrt{a + b + cx}}{\sin x + a^2} + \frac{1}{mx} \quad ;$$

$$Z = \frac{x + 2y}{3m} + \frac{abc}{m + x};$$

$$Q = 4.1x + 2.84y + \sqrt{z^2 + m}\,.$$

Обчислення необхідно виконати за умови, що „X” являє собою довільне число, значення якого необхідно ввести в режимі діалогу (інтерактивному режимі), значення інших змінних такі:

a=2.64 b=3.875 c=12.8, m=15.41.

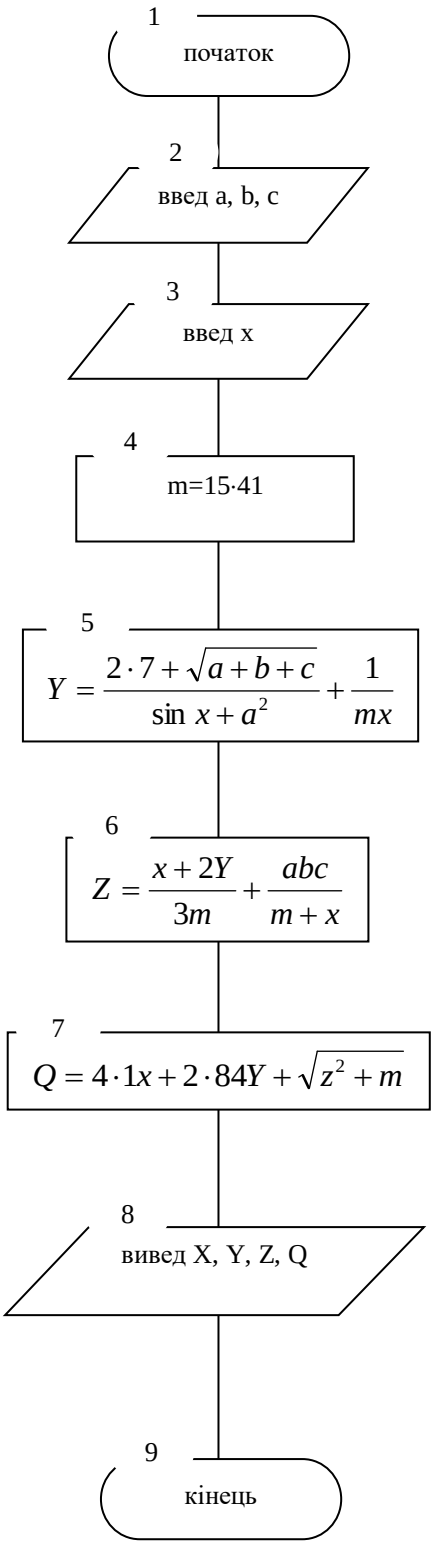
Для того, щоб Basic – програмі використати можливі варіанти завдання значень змінних, додатково вкажемо, що значення змінним a,b,c необхідно присвоїти за допомогою операторів **READ** та **DATA**, а змінну „m” задати за допомогою оператора присвоєння.

Будемо вважати, що на екран монітора після обчислення необхідно вивести не тільки значення функцій Y, Z та Q, а також і значення „X”, при якому вони отримані.

Один з варіантів схеми алгоритму реалізації таких процесів наведено на рисунку 5.1.

Чому ми стверджуємо, що один з варіантів і в чому тут інваріантність?

Справа в тому, що згідно з правилами реалізації лінійного алгоритму спочатку треба сформувати вхідні дані, потім виконати обчислення за формулами і врешті вивести результати. Що стосується послідовності обчислень, то вони обумовлені відповідними математичними залежностями і послідовність їх виконання відповідає послідовності їх запису, а от завдання вхідних даних багатоваріантне, наприклад, можна почати завдання змінних з m=15.41, або з введення значення „x”.




```
REM Лінійний обчислювальний процес
REM Обчислення функцій Y, Z, Q
CLS
INPUT “A=”, A
INPUT “B=”, B
INPUT “C=”, C
M=15.41
Y=2.7+SQR(A+B+C)/(SIN(X)+A^2)+1/(M*X)
Z=(X+2*Y)/3*M+A*B*C/(M+X)
Q=4.1*X+2.84*Y+SQR(Z^2+M)
PRINT “Результати обчислень:”
PRINT “X=”; X
PRINT “Y=”; Y
PRINT “Z=”; Z
PRINT “Q=”; Q
END
```

Рисунок 5.1

5.4 Програмування задач із використанням операторів передачі управління

5.4.1 Оператори умовного переходу

Умовні оператори QBasic здійснюють "розгалуження " програми, тобто передають управління на ту або іншу "гілку" залежно від результату виконання умови.

Оператор IF

IF...THEN...ELSE (ЯКЩО...ТО...ІНАКШЕ)

можна записати в лінійній або блоковій формах.

1) лінійна форма оператора **IF**

Якщо перевіряється одна або дві умови, то має сенс використати лінійну форму оператора **IF**

IF<УМОВА>THEN<ОПЕРАТОРИ_ТАК>[ELSE<ОПЕРАТОРИ_НІ>]

УМОВА - логічний вираз ("хибність" або "істина")

УМОВОЮ може служити навіть арифметичний вираз. Він набуває значення "хибність", якщо арифметичний вираз має значення нуль, і значення "істина" - в інших випадках.

ОПЕРАТОРИ_ТАК - виконуються при значенні логічного виразу "істина ".

ОПЕРАТОРИ_НІ - виконуються при значенні логічного виразу " хибність ".

Якщо ці ОПЕРАТОРИ складаються з декількох операторів QBasic, то вони розділяються двокрапкою і повинні обов'язково розміщатися в одному командному рядку.

На практиці використовують повну і скорочену форми запису оператора **IF**. Повна форма запису припускає наявність у записі гілки **ELSE**.

Приклад: З клавіатури вводяться чотири числа. Знайти найбільше. Вивести введені числа і результат.

```
CLS
INPUT "A=";A:INPUT "B=";B
INPUT "C=";C:INPUT "D=";D
IF A>B THEN MAX=A ELSE MAX=B
IF C>D THEN MAX1=C ELSE MAX1=D
IF MAX1>MAX THEN MAX=MAX1
PRINT
PRINT " УВЕДЕНІ ЧИСЛА ";A,B,C,D
PRINT " МАКСИМАЛЬНЕ З НИХ ";MAX
```

Скорочена форма запису зручна, коли частина ELSE відсутня. Тоді при значенні логічного виразу " хибність " виконання програми триває з наступного рядка.

Приклад: Вибір найбільшого із двох чисел A, B

```
INPUT " УВЕДІТЬ ЧИСЛА A, B===>";A,B
MAX=A
IF A<B THEN MAX=B
PRINT " MAX(";A;",";B;")=";MAX
```

Окремим випадком є команда виду

IF<УМОВА>THEN GOTO <РЯДОК>

рядок - мітка або номер наступного виконуваного рядка.

Якщо доводиться перевіряти більше, ніж дві умови (багаторівневі перевірки), то доцільно використати вкладені оператори **IF** або блокову форму **IF** - багаторівнева побудова (більш, ніж в один рядок).

Структура вкладених **IF** має такий формат:

IF<УМОВА 1>THEN IF<УМОВА 2> THEN <ОПЕРАТОРИ_ТАК>ELSE<ОПЕРАТОРИ_НІ>

<ОПЕРАТОРИ_ТАК> виконуються, якщо всі попередні умови мають значення "істина".

Якщо <УМОВА 1> - "істина", то перевіряється <УМОВА2>;

Якщо <УМОВА 1>- "хибність" - управління буде передано на наступний оператор без перевірки <УМОВА 2>.

Допускається використання **ELSE.** Кожна частина **ELSE** відповідає частині **THEN**, закриваючи найближчий оператор **IF**.

Глибина вкладення обмежується тільки довжиною рядка дисплея (оператор повинний бути записаний в один рядок).

Замість вкладених операторів можна використати логічні операції - відповідне ключове слово **THEN** заміняється операцією **AND**.

Приклад

```
IF A>B THEN IF A>C THEN PRINT "A - БІЛЬШЕ"
IF A>B AND A>C THEN PRINT "A - БІЛЬШЕ"
```

Вкладені структури з ELSE можуть бути досить складними, особливо якщо <ОПЕРАТОРИ_ТАК> складаються не з одного оператора, а з блока операторів. У цьому випадку використання багаторівневої структури спростить програму.

2) блокова форма оператора **IF**

Формат блокового **IF**

IF <умова1> **THEN**
[блок_операторів-1]

[**ELSEIF** < умова 2> **THEN**
[блок_ операторів-2]].
...
[**ELSE**
[блок_операторів-**n**]]

END IF

- умова1, умова2 - будь-які вирази, що можуть бути оцінені як "істина " (не нуль) або “хибність” (нуль);

- блок_ операторів-1, блок_ операторів-2,.... - один або декілька операторів в одному або декількох рядках.

Перевіряється <умова1>: якщо вона "істина" виконуються оператори блока **THEN**; якщо вона "хибність", аналізується кожна умова **ELSEIF**.

При виконанні якої-небудь із таких умов буде реалізований відповідний блок_операторів. Якщо жодна з умов **ELSEIF** не виконується, управління передається блоку **ELSE**, а якщо його немає - операторові, що розташований за **END IF**. У блокову структуру можна вставити будь-яку кількість умов **ELSEIF**.

Виконання кожного з альтернативних блоків автоматично завершується переходом на **END IF**.

Приклад

```
CLS:INPUT "Введіть число X=",X
IF X > 0 THEN
PRINT " X - Додатне "
ELSEIF X < 0 THEN
PRINT " X - Від’ємне "
ELSE
PRINT " X - нуль "
End If
End
```

Будь-який з блоків може містити вкладені блокові структури.

ОПЕРАТОР SELECT CASE... END SELECT (ВИБРАТИ ВАРІАНТ)

Можна використовувати при перевірці складних умов.

Це управляючий оператор, що виконує один з декількох блоків операторів залежно від результатів перевірки заданих умов.

Формат оператора:

SELECT CASE <вираз вибору> (<тест-вираз>)
CASE <список_виразів1>
[блок_операторів-1]
[CASE <список_виразів2>
[блок_операторів-2]]...
[CASE ELSE
[блок_операторів-n]]
END SELECT

<вираз вибору> або <тест-вираз> - будь-який числовий або символьний вираз.

<список_виразів> - один або кілька виразів для порівняння з виразом вибору.

У вираз перед будь-яким знаком відношення додається спеціальне ключове слово IS

<блок_операторів-1>, <блок_операторів-2> - один або кілька операторів, записаних в одному або декількох рядках.

Існують різні способи запису умовних виразів у блоках CASE. Аргументи списку виразів можуть набувати кожну з наступних форм або їхню комбінацію і повинні розділятися комами:

значення; - вираз і значення перевіряються на рівність один одному;

значення1 TO значення 2; - перевіряється, чи належить значення-вираз області [значення1 - значення2] (менше значення повинне бути першим). Якщо належить, то виконується відповідний блок операторів;

IS знак_відношення вираз

вираз - будь-який числовий або символьний вираз, сумісний з виразом вибору.

знак_відношення - один із знаків відношення QBasic

Якщо значення - вираз потрапляє в діапазон, зазначений в CASE, програма виконує відповідний блок операторів.

Умова, що перевіряється, може мати і більш складний вид: CASE IS<0, IS>90.

Якщо вираз вибору відповідає умовам списку виразів даного блока CASE, то виконуються оператори цього блока. Якщо жодна з умов не виконується, управління передається CASE ELSE; якщо CASE ELSE немає - операторові, що розташований за END SELECT.

Якщо вираз вибору задовольняє декілька умов CASE, виконується блок операторів, що йде першим.

Блоки SELECT CASE можуть бути вкладеними. Кожний блок повинен завершуватися END SELECT.

Приклад: Аналіз величини числа.

```
INPUT " ВВЕДІТЬ ЧИСЛО N ";N
SELECT CASE N
CASE 1,3,5,7 9
PRINT " НЕПАРНЕ"
CASE 2,4,6,8
PRINT " ПАРНЕ "
CASE IS<1
PRINT " ДУЖЕ МАЛЕНЬКЕ "
CASE IS>9
PRINT " ДУЖЕ ВЕЛИКЕ "
CASE ELSE
PRINT " НЕ ЦІЛЕ ЗНАЧЕННЯ "
END SELECT
```

5.4.2 Оператори безумовного переходу

Від старих версій Basic в QBasic збереглася можливість нумерації програмних рядків і звертання до них за номером.

Оператор GOTO <Рядок >

виконує безумовний перехід до зазначеного рядка.

Цей оператор повинен бути або єдиним у рядку, або останнім у багатооператорному рядку.

Приклад

```
GOTO 67
8 GOTO MP
....
MP: A=5
S=0:N=N+1:GOTO 120
```

Оператор множинного вибору On ... Goto

Формат оператора:
On <показник переходу> Goto <рядки>
виконує перехід до однієї із дій, позначених мітками залежно від значення виразу, записаного в показнику переходу. Вираз набуває значення в діапазоні від 0 до 255.
<рядки> - набір міток або номерів рядків.

При реалізації оператора обчислюється ціла частина виразу (при потребі з округленням). Якщо вона дорівнює 1, то програма переходить на першу мітку (на відповідний рядок); якщо - 2, то на другу і т.д. Якщо ціла частина виразу=0 або > числа міток, то управління передається операторові, що розташований за ON ...GOTO; якщо ж ціла частина виразу< 0, то буде повідомлення про помилку.

Приклад

```
20 INPUT „Ввести номер елемента N=”, N
ON N GOTO 100,150,80,210
GOTO 999
80 PRINT "ЛІТІЙ": GOTO 20
100 PRINT "КИСЕНЬ": GOTO 20
150 PRINT "ГЕЛІЙ": GOTO 20
210 PRINT "БАРІЙ": GOTO 20
999 END
```

5.4.3 Приклад програмування розгалуженого обчислювального процесу

Нехай до пам'яті ПК вводяться три числа a, b, c, які являють собою довжини сторін трикутника. Необхідно скласти схему алгоритму та програму мовою QBasic визначення та виведення на екран площі „S". У випадку, коли побудова трикутника неможлива, вивести на екран відповідне повідомлення.

Постановка задачі: Площа трикутника при відомих довжинах сторін може бути обчислена за формулою

$$S = \sqrt{p(p-a)*(p-b)*(p-c)},$$

де p – напівпериметр трикутника, значення якого може бути обчислене за формулою

$$p = (a+b+c)/2.$$

З геометрії відомо, що трикутник можна побудувати у тому випадку, якщо сума довжин двох будь-яких сторін більше довжини третьої.

При проектуванні алгоритму слід передбачити, що довжини сторін a, b, c вводяться довільно, тобто необхідно передбачити перевірки можливості побудови трикутника для випадків, коли будь-яка із сторін може бути найдовшою.

Кількість варіантів позначень сторін при побудові трикутника може бути визначена як число усіх перестановок P_n з „n" різних елементів.

В нашому випадку n=3, тому

$$P_n = n! = 1*2*3 = 6,$$

тобто для введення числових значень a, b, c можливі такі варіанти:

$$a,b,c;\ \ b,a,c;\ \ c,a,b;\ \ a,c,b;\ \ c,b,a;\ \ b,c,a.$$

Зробимо аналіз умов, що перевіряються:

$$\begin{cases} a+b>c \\ b+a>c \end{cases} \quad \begin{cases} c+a>b \\ a+c>b \end{cases} \quad \begin{cases} c+b>a \\ b+c>a \end{cases}$$

Звідки видно, що досить перевірити 3 умови:

$$a+b \succ c; \quad a+c \succ b \quad \text{та} \quad b+c \succ a.$$

На основі викладених міркувань та аналізу умов задачі схема алгоритму розв'язання може бути подана таким чином (рисунок 5.2):

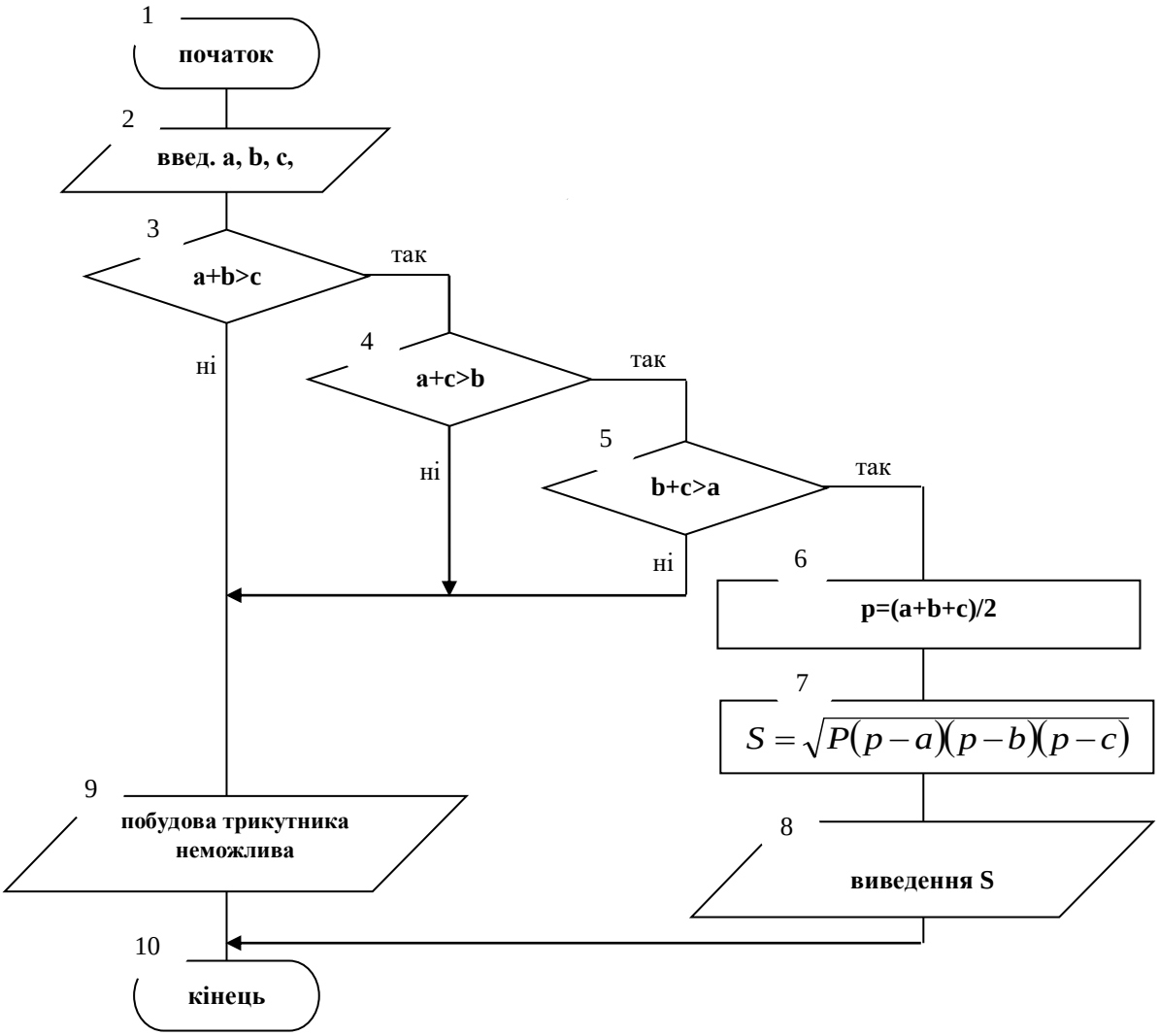


Рисунок 5.2

```
REM Обчислення площі трикутника при відомих довжинах сторін A, B, C
CLS
INPUT “Ввести довжини сторін A, B, C===>”; A, B, C
IF A+B>C AND A+C>B AND B+C>A THEN
P = (A+B+C)/2
S = SQR(P*(P-A)*(P-B)*(P-C))
PRINT “Площа трикутника S=”; S
ELSE
PRINT “Побудова трикутника неможлива!”
END IF
END
```

5.5 Оператори циклів

Призначені для багаторазової реалізації процесів, які описуються однією і тією ж послідовністю операторів з різними даними.

Оператор циклу складається із двох частин. Перша частина - це заголовок циклу, друга частина - кінець циклу. Між ними розміщається задана послідовність операторів - тіло циклу.

Головними операторами циклів мови QBasic є оператори:

- **FOR ... NEXT**
- **WHILE ... WEND**
- **DO ... LOOP**

5.5.1 Оператори FOR ... NEXT

Ці оператори дозволяють організувати виконання блока операторів певну кількість разів, яка задається безпосередньо в операторі або може бути легко обчислена до початку виконання. За типом - це цикл з передумовою.

Формат оператора:

FOR <лічильник> =<початок> TO <кінець>[STEP <збільшення>]
[тіло циклу]
NEXT [лічильник]

- FOR - визначає початок циклу (для);
- TO - визначає кінець циклу (до);
- STEP - крок зміни лічильника (параметра циклу);
- NEXT - кінець - наступний;
- лічильник - числова змінна (параметр циклу), яка визначає число повторень циклу;
- початок, кінець - арифметичні вирази, які визначають початкове і кінцеве значення лічильника;
- збільшення (крок) - числова константа або арифметичний вираз, що визначає зміну лічильника при кожному кроці циклу. Якщо крок не заданий, то за замовчуванням він приймається рівним одиниці;
- тіло циклу (блок операторів) - набір операторів, що виконуються багаторазово.

У процесі виконання цього оператора реалізуються такі дії:

1 Обчислюються значення арифметичних виразів (початок, кінець, збільшення).

2 Параметру циклу присвоюється початкове значення (формується лічильник циклу).

3 Перевіряється умова: якщо крок >0 і параметр <=кінцевого значення або крок <0 і параметр >=кінцевого значення, то виконується тіло циклу; значення параметра змінюється на величину кроку і повторюється п.3. У протилежному випадку виконуються програмні рядки, що розташовані за оператором. При кроці = 0 цикл стає нескінченним.

Правила застосування операторів **FOR ... NEXT:**

- параметр циклу, зазначений в **FOR**, повинен збігатися з параметром, зазначеним в **NEXT**;
- не можна передавати управління у середину циклу - у цьому випадку параметр циклу невизначений;

- параметр циклу, початок, кінець, крок не можна змінювати у середині циклу;
- можна виходити із циклу природно (тобто після виконання його задане число раз), а також за допомогою управляючих операторів з будь-якого оператора тіла циклу.

Наприклад, можна також вийти із циклу, не дочекавшись виконання всіх повторень, скориставшись альтернативним виходом із циклу за допомогою оператора **EXIT FOR**. Управління буде передано операторові, що розташований після **NEXT**.

- після завершення циклу значення параметра дорівнює кінцевому значенню плюс крок;
- із циклу припустиме звертання до підпрограми з наступним поверненням у нього.

Приклад: Програма виведення таблиці значень X, X^2, X^3

```
CLS:PRINT " ЗАДАЙТЕ ПАРАМЕТРИ:"
INPUT " ПОЧАТКОВЕ ЗНАЧЕННЯ X1=";X1
INPUT " КІНЦЕВЕ ЗНАЧЕННЯ X2=";X2
INPUT " КРОК X3=";X3
N=0
FOR X=X1 TO X2 STEP X3
PRINT X,X^2,X^3
N=N+1
NEXT X
PRINT " ЦИКЛ ВИКОНУЄТЬСЯ ";N;" РАЗІВ "
END
```

Цикли **FOR ... NEXT** можуть бути вкладеними. Кожний вкладений цикл повинен мати свої ідентифікатори параметра. Оператор **NEXT** для внутрішнього циклу повинен виконуватися раніше оператора **NEXT** для зовнішнього циклу. Глибина вкладення обмежується тільки розміром доступної пам'яті в робочій області QBasic.

Якщо кілька циклів мають одну загальну кінцеву точку, то можна вказати для них один оператор **NEXT**, перелічивши в

ньому параметри циклів у порядку, зворотному їхній вкладеності (від внутрішніх - до зовнішнього).

Приклад

```
FOR A=1 TO 3
      FOR B=5 TO 14
            C=A+B
            PRINT C
      NEXT B          }        NEXT B, A
NEXT A
```

5.5.2 Оператори WHILE ... WEND

Застосовуються, якщо необхідно обчислювальний процес перервати за певною умовою, і при цьому кількість повторень циклу не визначено.

Формат оператора:

WHILE <УМОВА>
[тіло циклу]
WEND

<УМОВА> - числовий або логічний вираз, який QBasic оцінює як “істина” або “хибність”.

Перш ніж буде виконуватися тіло циклу, визначається результат умови в рядку **WHILE**. Тіло циклу виконується, поки зазначена умова істина, після чого управління повертається в початок оператора **WHILE**, де знову перевіряється умова.

Якщо умова хибна - управління одержує наступний за **WEND** оператор.

Як і **FOR...NEXT**, даний оператор реалізує цикл із передумовою, але різниця полягає в тому, що потрібно визначити початкові і кінцеві значення до початку циклу і самим міняти значення змінної, що використовується в перевірці (інакше можливе зациклення).

Допускається вкладеність, але кожний **WHILE** повинен мати свій **WEND**.

Приклад: Програма обчислення наближеного значення EXP(X) для заданого значення X за формулою

$$EXP(X) = 1+X+X^2/2! +X^3/3! +X^4/4! +...$$

Обчислення суми ряду виконується, поки величина члена, що додається, не стане меншою за значення $E=10^{-6}$.

Обчислення нового елемента можна здійснити за формулою

$$A=A*X/N.$$

```
CLS
INPUT " УВЕДІТЬ ЗНАЧЕННЯ X ";X
INPUT " ЗАДАЙТЕ ТОЧНІСТЬ ОБЧИСЛЕНЬ ";E
Y=1:N=1:A=X
WHILE ABS(A)>E
    Y=Y+A
    N=N+1
    A=A*X/N
    WEND
    PRINT " EXP(";X;")=";Y
```

5.5.3 Оператор **DO ... LOOP**

Найбільші можливості (зокрема виконання тіла циклу з умовою закінчення як на початку, так і наприкінці циклу) надає конструкція, що відповідає циклу типу **DO ... LOOP**.

Цей оператор дуже схожий на **WHILE ...WEND**, але він більш гнучкий і в цьому його перевага.

Оператор має 4 форми запису:

- перевірка угорі (цикл з передумовою - цикл "ПОКИ")

```
Do [{WHILE|UNTIL}<УМОВА>]
    [тіло циклу]
    LOOP,
```

де
DO - виконати, WHILE - поки, UNTIL - поки не, LOOP - цикл.
< УМОВА> - вираз, що QBasic оцінює як істина або хибність.

Повторюється тіло циклу, поки умова вірна (WHILE) або поки умова не стане вірна (UNTIL).

Умова перевіряється на початку циклу, тому тіло циклу може взагалі жодного разу не виконатися, коли результат першої перевірки – хибність;

- перевірка внизу (цикл із післяумовою - цикл "ДО")

DO
    [тіло циклу]
LOOP[{WHILE|UNTIL}<УМОВА>]

Перевірка умови виконується наприкінці циклу, у зв'язку із цим тіло циклу завжди виконується хоча б один раз;

- DO у спрощеному вигляді
  DO
  [тіло циклу]
  LOOP

Створює нескінченний цикл (з нього можна вийти, використовуючи оператор **EXIT DO** або інший управляючий оператор).

Приклад: Програма обчислення за заданою формулою 100 членів послідовності $A_k=k^2/(k^3+1)$

```
CLS:K=1
DO
A=K^2/(K^3+1)
PRINT "A(";K;")=";A
K=K+1
LOOP UNTIL K>100
```

5.5.4 Приклад програмування вкладеного циклічного процесу

Уявімо собі, що кондуктор міського транспорту, наприклад трамвая, отримав гіпотетичну котушку з квитками, номери яких знаходяться в межах від 000001 до 999999. Серед цих квитків є, так звані, „щасливі” квитки. „Щасливим” прийнято вважати квиток , у якому сума перших трьох цифр дорівнює сумі останніх трьох цифр. Необхідно визначити скільки ж знаходиться „щасливих” квитків серед усіх 999999 квитків?

Найпростіший спосіб розв’язання задачі полягає в тому, що необхідно поступово збільшувати на одиницю номер квитка та перевіряти – чи не є він щасливим? Кожен з розрядів номера квитка може змінюватись в межах від 0 до 9. Всього розрядів – 6. Отже, позначимо кожен з розрядів відповідною літерою від „a” до „e” і тоді умова появи „щасливого” квитка формується як

$$a + b + c = d + e + f$$

Позначимо через KS кількість „щасливих” квитків в котушці. Тоді реалізуючи алгоритм, наведений на рисунку 5.3, ми дізнаємось, що всього їх буде 55252 штуки.

```
REM Розрахунок кількості щасливих квитків в котушці кондуктора
CLS
KS=0
FOR A=0 TO 9
FOR B=0 TO 9
FOR C=0 TO 9
FOR D=0 TO 9
FOR E=0 TO 9
FOR F=0 TO 9
IF A+B+C=D+E+F THEN KS=KS+1
NEXT F
NEXT E
NEXT D
NEXT C
NEXT B
```

NEXT A
PRINT “Кількість щасливих квитків ”;KS;” штук”
END

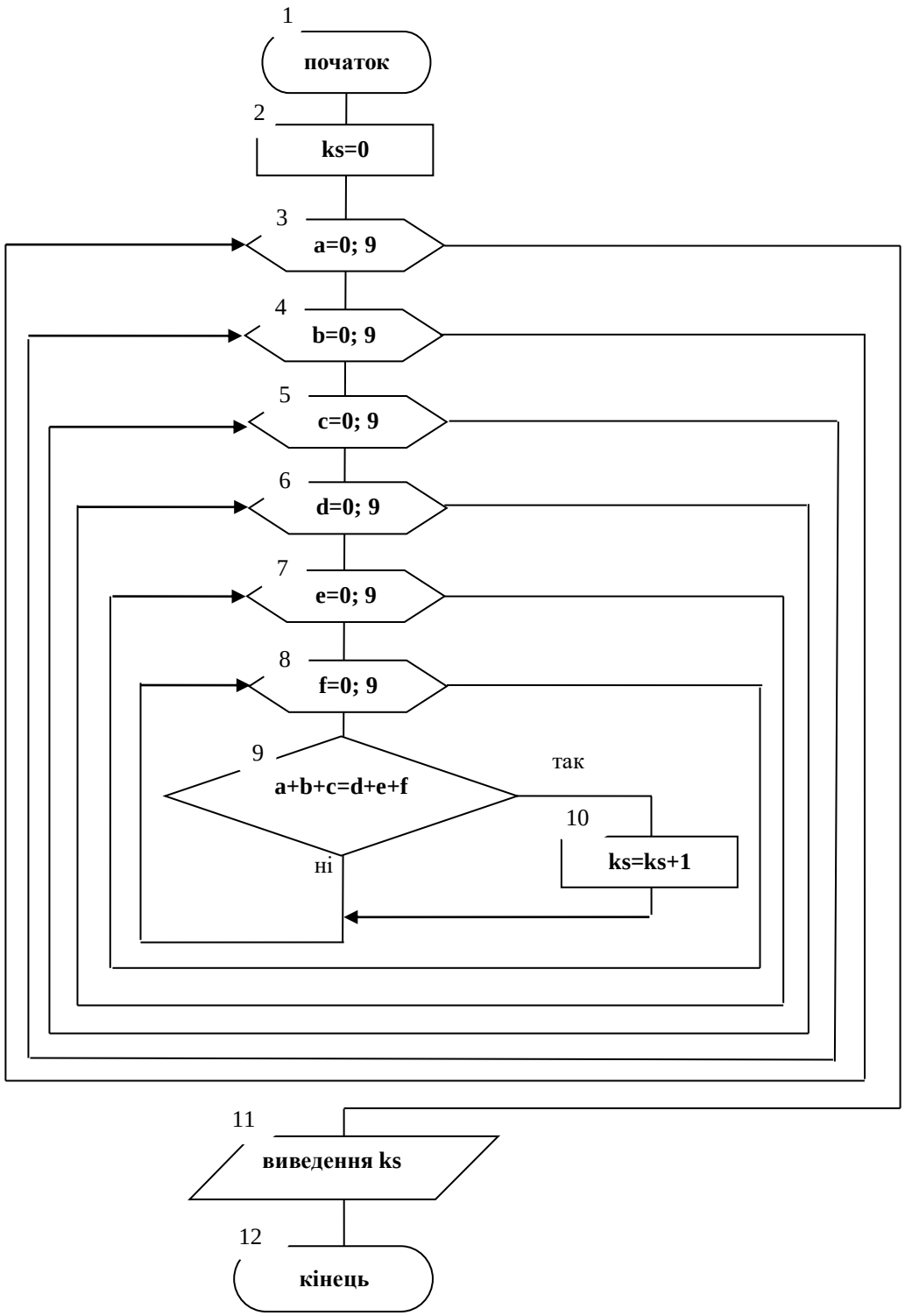


Рисунок 5.3

5.5.5 Приклад вкладеного циклічного процесу з розгалуженням

Відомо, що для компенсації відцентрової сили в кривих на залізниці підвищують зовнішню рейку колії. Величина підвищення рейки, позначимо її як h_p, залежить від радіуса кривої R та від середньої швидкості руху потяга $V_{ср}$.

Величина підвищення зовнішньої рейки визначається такими залежностями:

$$h_p = A_h \frac{12{,}5V_{ср^2}}{R} \le 150,$$

де

$$A_h = \begin{cases} 1, якщо Vср \le 120км/год \\ 1.2, якщо V_{ср} \succ 120км/год \end{cases}$$

З наведених формул видно, що максимальне підвищення зовнішньої рейки не може перевищувати 150 мм, (обмеження).

Виконаємо побудову схеми алгоритму та Basic – програми за умови зміни двох параметрів:

- радіуса кривої $R \in [2000;3000], h_r = 200$;
- швидкості руху $V_{ср} \in [80;160], h_{ср} = 10$.

Результати обчислень необхідно подавати в табличному вигляді таким чином:

$R = \cdots$ $\quad V_{ср} = \cdots$ $\quad h_p = \cdots$

Для спрощення запису фізичних величин скористаємося такими ідентифікаторами в алгоритмі та Basic-програми.

Фізичні величини	h_p	A_h	R	$V_{ср}$
Ідентифікатори	h	a	r	V

З урахуванням наведених ідентифікаторів схема алгоритму розрахунку підвищення зовнішньої рейки в кривих може мати вигляд, наведений на рисунку 5.4.

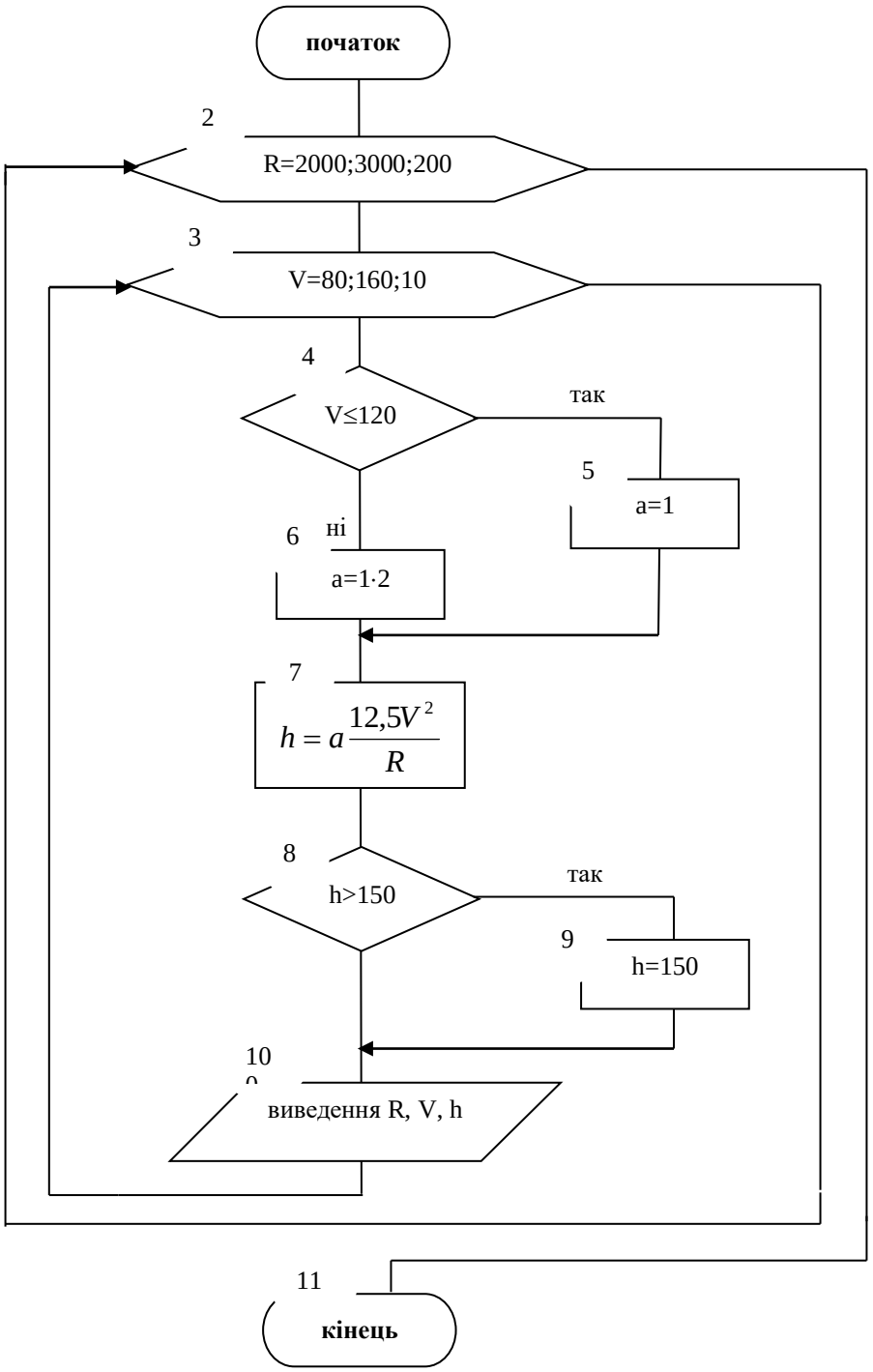


Рисунок 5.4

```
REM Розрахунок підвищення рейки в кривих

CLS: FOR R=2000 TO 3000 STEP 200
FOR V=80 TO 160 STEP 10
IF V<=120 THEN A=1 ELSE A=1.2
H=A*12.5*V^2/R
IF H>150 THEN H=150
PRINT “R=”;R;”м”, “V=”;V;”км / год”, “H=”;H;”мм”
NEXT V, R
END
```

5.6 Деякі корисні команди мови QBASIC

Розглянемо деякі команди мови Qbasic, що часто використовуються при написанні програм.

Команда **CLS**. Очищує з екрана монітора отримані раніше результати. Має такий вид:

CLS.

SLEEP (Пауза). Для організації паузи при роботі програми використовують команду **SLEEP(n),** яка затримує виконання програми на n секунд. Крім того, можна використати команду введення

Приклад

INPUT P$, де P$ - деяка змінна.

Якщо паузу потрібно закінчити, то варто натиснути алфавітно-цифрову клавішу і клавішу введення.

Команда - коментар REM використовується для внесення пояснень у текст програми (його можна замінити апострофом '):

REM <текст пояснення>

або

' <текст пояснення>

Ця команда може перебувати в будь-якому місці програми, але в рядку повинна бути єдиною або останньою. Вона не впливає на виконання програми.

Команда зупинки **STOP**. Команда зупинки виконання програми може перебувати в будь-якому місці програми. Ця команда зупиняє виконання програми. Часто її використовують, щоб переписати проміжні результати з екрана монітора. Виконання програми можна продовжити за допомогою натискання клавіші F5.

Команда **END** - команда закінчення роботи програми.

Команда **SWAP**. Дозволяє виконати обмін значеннями між двома змінними. Команда має вигляд SWAP A, B
де A і B - імена змінних, які обмінюються значеннями.

Приклад: У результаті виконання програми

```
A=7:B=5
SWAP A,B
PRINT A,B
```

на екран монітора будуть виведені значення 5 і 7.

У класичному Basic для обміну даними необхідно вводити проміжну змінну, наприклад C, і програма може виглядати так:

```
A=7: B=5
C=A: A=B: B=C
PRINT A,B
```

5.7 Програмування задач обробки масивів даних

Масиви являють собою впорядкований набір однотипних елементів, послідовно розміщених в оперативній пам'яті ПК.

Всі елементи масиву мають загальний ідентифікатор, відповідний певному типу даних, слідом за яким у круглих дужках записуються один або кілька індексів, що визначають місце розташування елемента в масиві.

Максимальне значення індексу 32767, мінімальне - за замовчуванням - 0. При необхідності воно може бути встановлено рівним 1. Для цього використовується оператор **OPTION BASE N**, де N - оголошення нижньої границі масиву (0 або 1). Надалі приймемо, що N=1.

Розмір масиву визначається кількістю елементів, які входять у нього.

Вимірністю називається кількість вимірювань масиву і кількісно воно визначається числом індексів при змінній.

Кожен елемент масиву є індексованою змінною, тобто змінною, що має порядковий номер. Ім'я змінної збігається з ім'ям масиву.

Приклад

A(5) - п'ята за порядком змінна одновимірного масиву A,

M(5,4) - індексована змінна двовимірного масиву M, розташована в 5-ому рядку і 4-ому стовпці.

Масиви, що використовуються в програмах, повинні бути обов'язково в них описані (оголошені).

Для опису масиву в QBasic є спеціальний оператор-описувач DIM, що розташовується в програмі раніше, ніж починається робота з відповідним масивом (звичайно всі вони розташовуються на початку програми для полегшення відлагодження).

Цим оператором одночасно:

- визначається ім'я масиву;
- описується тип елементів масиву або вказується тип даних для скалярних змінних (не масивів);
- обнуляються всі елементи числових масивів, а символьним - присвоюється значення порожнього рядка (" ");
- резервуються комірки пам'яті для масивів.

Формат оператора:

DIM[SHARED]<змінна>[(<індекс>)] [AS тип][,<змінна>[(<індекс>)[AS тип]]

- SHARED вказує, що змінні використовуються спільно всіма процедурами SUB або FUNCTION у модулі.
- <Змінна> - ім'я змінної масиву.
- <індекс> - розміри масиву у вигляді: [низ TO] верх [, [низ TO] верх]...
- низ - нижня границя індексу масиву. За замовчуванням нижня границя дорівнює нулю; верх - верхня границя.
- AS тип - описує тип елементів масиву або скалярну змінну (INTEGER, LONG, SINGLE, DOUBLE, STRING або тип даних, що задається користувачем).

Приклад

DIM V(100) - одновимірний масив зі 100 дійсних елементів.

DIM M%(2,3) - цілочислова матриця M% розміром 2*3.

DIM text$(50) - масив з 50 символьних елементів.

DIM A(100), X(50), P(14,26) резервує в пам'яті місця для двох одновимірних і одного двовимірного масиву.

DIM X(4 To 10) - одновимірний масив з індексами, які змінюються від 4 до 10.

DIM A(3 To 8, 0 To 3) - двовимірний масив з індексами, які змінюються від 3 до 8 та від 0 до 3

DIM Y(1 To 52) AS INTERGER - цілочисловий масив з індексами, що змінюються від 1 до 52.

5.7.1 Приклади програмування завдань обробки одновимірних і двовимірних масивів

В математиці одновимірні таблиці називаються векторами або стовпцями. Для введення, виведення та обробки одновимірних масивів використовуються прості цикли.

Двовимірна таблиця (масив) містить M*N елементів, і кожний елемент має два індекси.

Перший індекс показує номер рядка, а другий - номер стовпця, що відповідають місцезнаходженню елемента. У математиці двовимірні таблиці називаються матрицями.

Для введення, виведення та обробки двовимірних масивів використовуються вкладені цикли.

Приклад 1: Ввести масив X з N елементів та визначити, чи є в цьому масиві хоча б одна пара взаємно обернених чисел.

```
INPUT " Число елементів  в масиві N=",N
'Виділення пам'яті здійснюється під час виконання програми
DIM X(N)
FOR I = 1 TO N
PRINT "ВВЕДІТЬ X(";I;")=",
INPUT X(I)
NEXT I
Y$ = " Hi "
FOR I = 1 TO N-1
IF X(I)*X(I+1)=1 THEN
        Y$="Так":PRINT I;I+1;"Елемент":EXIT FOR
END IF
NEXT I
PRINT Y$
END
```

Приклад 2:

Відсортувати назви міст (символьні змінні) в алфавітному порядку.

```
REM Сортування масиву методом “Бульбашки”
DEFINT A
CONST FALSE=0,TRUE=NOT FALSE
DIM A$(10)
A$(1)="Москва"
A$(2)="Ленінград"
A$(3)="Харків"
A$(4)="Полтава"
A$(5)="Київ"
A$(6)="Мінськ"
A$(7)="Владивосток"
A$(8)="Донецьк"
```

```
A$(9)="Тула"
A$(10)="Бєлгород"
EXCHANGE=TRUE
'Сортувати, поки всі елементи не зміняться.
WHILE EXCHANGE=TRUE
EXCHANGE=FALSE
'Порівняння елементів масиву попарно. Якщо відбулася
'Заміна, присвоїти змінній EXCHANGE значення TRUE.
FOR I=2 TO 10
      IF A$(I-1)>A$(I) THEN
            EXCHANGE=TRUE
            SWAP A$(I-1),A$(I)
      END IF
NEXT I
WEND
CLS
FOR I = 1 TO 10
      PRINT A$(I)
NEXT I
```

Приклад 3

Десять студентів однієї групи склали по 5 іспитів. Результати сесії подати як двовимірний масив X(10,5). Кожний рядок містить оцінки одного студента. Необхідно:

- обчислити середній бал з кожного предмета;
- обчислити середній бал для кожного студента.

Підсумкові дані записати у двовимірний масив Y(11,6), який вивести у вигляді таблиці:

по вертикалі 10 рядків (студенти) і одинадцятий рядок - середній бал з кожного предмета;

по горизонталі 5 стовпців (оцінки) і шостий стовпець - середній бал для кожного студента.

№ з/п	Мате-матика	Фізика	Хімія	Історія	Гра-фіка	Середній бал по студенту
1	5	3	4	5	4	4.2
2	3	5	5	3	4	4
3	4	3	4	3	4	3.6
4	3	2	5	4	4	3.6
...	...	...	...	...	...	...
10	3	4	5	2	4	3.6
Середн. бал з пред-мета						

```
CLS
DIM X(10,5),Y(11,6)
LOCATE 1,10
PRINT "РЕЗУЛЬТАТИ СЕСІЇ"
LOCATE 2,10
PRINT " ОДИН РЯДОК-10 ОЦІНОК З ОДНОГО ПРЕДМЕТА "
FOR I= 1 TO 5
        FOR J = 1 TO 10
                LOCATE I + 2, J*7
                NPUT X(I,J)
        NEXT J
NEXT I
FOR I=1 TO 5
        S=0
        FOR J = 1 TO 10
                S=S+X(I,J)
        NEXT J
        Y(11,I)=S/10
NEXT I
PRINT
PRINT "ТАБЛИЦЯ ПО ВЕРТИКАЛІ - СТУДЕНТИ,";
PRINT "ПО ГОРИЗОНТАЛІ - ОЦІНКИ "
FOR I = 1 TO 10
        SUM=0
        FOR J = 1 TO 5
                Y(I,J)=X(J,I)
```

```
            SUM=SUM+Y(I,J)
        NEXT J
        Y(I,6)=SUM/5
NEXT I
PRINT "ВИВЕДЕННЯ СФОРМОВАНОЇ ТАБЛИЦІ:"
FOR I = 1 TO 10
        FOR J = 1 TO 6
            LOCATE I+12,J*7
            PRINT Y(I,J)
        NEXT J
NEXT I
FOR J = 1 TO 5
        LOCATE I+12,J*7
        PRINT Y(11,J)
NEXT J
END
```

5.8 Розробка програм з використанням табличних форм

При рішенні економічних, інженерних завдань, завдань управління доводиться працювати зі складними об'єктами, які складаються з елементів різних типів. Інформацію в цьому випадку доцільно подавати у вигляді таблиць (зручному для сприйняття), які підлягають обробці для одержання необхідних результатів.

Зонний формат виведення інформації за допомогою оператора PRINT не дозволяє одержувати якісні документи у формі звичайних таблиць через те, що відображувані значення притискаються до лівої границі зони. При цьому дані виявляються зміщеними, що приводить до скривлення стовпчиків таблиці.

Необхідний нестандартний формат виведення, що дозволяє самостійно вирішувати, у якому місці виведеного рядка помістити те або інше числове значення, скільки цифр варто відвести цілій і дробовій частинам і т.д.

Для цієї мети призначений оператор **PRINT USING**:

Формат оператора:

PRINT USING "<список форматів>"; <список_ *виведення* **>,**

де <список форматів>- один або кілька визначників формату (форматних полів);

<визначник формату> - текстовий вираз (у подвійних лапках), що складається із символів управління форматом *виведення*. Символи бувають звичайні й спеціальні.

<список *виведення* > - числові або символьні вирази, розділені крапкою з комою (проміжок і кома автоматично перетворяться в крапку з комою)

виведення **чисел**

Спеціальні символи для *виведення* числових даних:

#	^	$	.(крапка)	, (кома)

.

- Символ # резервує цифрову позицію. Група # задає числове поле *виведення*.
- Крапка в описі числового поля задає позицію десяткової крапки, що відокремлює цілу частину від дробової. Число вирівнюється по правому краю, а зайві позиції ліворуч заповнюються проміжками. При необхідності виконується округлення *(не усікання*). Зайві позиції після крапки заповнюються нулями. Якщо задане числове поле мале (не вміщується ціла частина або знак числа), то виводиться символ %, а після нього - повне значення числа.

Приклад:

```
PRINT USING "###.##"; 123.456        результат: 123.46
PRINT USING "##.###"; -2.54          результат: -2.540
PRINT USING "######"; 123456         результат: 123456
PRINT USING "S=##.#"; 13.247         результат: S=13.2
PRINT USING "###.##"; 12345.67       результат: %12345.67
PRINT USING "#.##"; 9.99999          результат: %10.00
```

- Чотири підряд знаки піднесення до степеня визначають місцезнаходження символу E або D, знака порядку й порядку числа, тобто число виводиться в плаваючому форматі.

Приклад

PRINT USING"##.###^^^^";123.456
результат:1.235E+02

- Комбінація $$ на початку шаблона числового поля викличе появу знака грошової одиниці $ перед старшою значущою цифрою числа.

Приклад

PRINT USING "$$###.##"; 123.456 результат: $123.46
PRINT USING *"$###.##";* 123.456 результат: $ 123.46)

- Кома у форматному полі використовується для відділення у великих числах тисяч і мільйонів.

Приклад

PRINT USING"$$###,###.##";2115.5
результат:$2,115.50

Виведення текстів

Спеціальні символи для *виведення* текстових даних:

- "&" - виводить текстовий вираз цілком.

Приклад

PRINT USING "&"; "ABCDEFGHIG" результат: ABCDEFGHIG

- \ n проміжків \ - переносять у виведений рядок (n+2) початкових символів текстового виразу.

Приклад

PRINT USING "\ \";"ABCDEFGHIG" результат: ABCDEFGH

- ! - переносить у виведений рядок тільки перший символ текстового виразу.

Приклад

PRINT USING "! "; "ABCDEFGHIG" результат: A

- Всі інші символи алфавіту в шаблоні (звичайні) без усякої зміни переносяться у відповідні позиції вивідного рядка й використовуються для оформлення результатів. Якщо в їхній склад необхідно включити символи, які використовуються для опису числових і символьних полів, то перед будь-яким спеціальним символом у шаблоні розміщають знак “підкреслення” ("_").

Приклад: PRINT USING "Номера _## _& _##"; 1; 2
результат: Номера #1 & #2.

- Якщо список форматів містить кілька визначників формату, а список виведення - кілька елементів, то перший елемент списку виведення виводиться по першому формату, другий - по другому й т.д. Елементи виводяться без пропусків, тому числові формати рекомендується розділяти символьними константами.

Приклад: PRINT USING "X=#.# F=####";.151,366.3
результат:X=0.2 F=366.

Коли список форматів буде вичерпаний, визначники формату почнуть вибиратися з початку списку.

Приклад: PRINT USING "##.##";10.1,-4.828
результат:10.10 -4.83

Можна оформити список форматів у вигляді символьної змінної (форматного рядка), а в операторі зробити посилання на цей рядок.

Приклад: a$ = "Курс #, група #" b = 1: c = 2
PRINT USING a$; b; c результат: Курс 1 група 2

Для зображення таблиць можливе використання символів, наведених на клавіатурі, а також символів псевдографіки. "-" "=" ", " "|" ┌ ╗ ╤ і т.д.

5.8.1 Приклад програмування завдання з виведенням результатів у вигляді табличної форми

Постановка завдання

Задано інформацію про успішність студентів групи з 10 чоловік.
Атрибути:

прізвище - до 15 символів, ім'я - 10 , по батькові - 10, рік народження - 4, стать - 5, група - 10,
оцінка з математики - 3, оцінка з фізики – 3, оцінка з хімії - 3

Необхідно:

Вивести вихідні дані у вигляді таблиці:

Список студентів групи 3

№	Прізвище	Ім'я	По батькові	Стать	Рік н.	Мат	Фіз	Хім
1	Токарєв	Іван	Борисович	м	1983	4	2	5
2	Рудь	Ганна	Сидорівна	ж	1984	4	3	4
…	…...	…...	…...	...	…...	…...	…...	…...

Вивести дані про наймолодшого студента:

Рудь А.С. 1984р.

Вивести інформацію про боржників, якщо їх немає - видати відповідне повідомлення:

Боржник: Токарєв Іван Борисович
оцінка з математики **- 4**
оцінка з фізики **- 2**
оцінка з хімії **- 5**
або
боржників немає

Підрахувати й вивести процентне співвідношення чоловіків і жінок у групі.

У групі 3 - 50% чоловіків і 50% жінок

Реалізація завдання

Таблиця ідентифікаторів

№	Іденти-фікатор	Опис (тип ідентифікатора)
1	N	Кількість записів у таблиці (числ.)
2	I	Поточний номер студента в списку (числ.)
3	max	Найбільший рік народження (числ.)
4	imax	Номер у списку студента з найбільшим роком народження (числ.)
5	PR	Ознака наявності боржників у групі (1- боржники є, 0 - немає) (числ.)
6	CM,CG	Число чоловіків і жінок (числ.)
7	PM,PG	Відсоток чоловіків і жінок (числ.)
8	NG$	Найменування групи (симв.)
9	L$	Горизонтальна лінія (симв.)
10	FORM$	Формат для виведення рядка таблиці (симв.)
11	FAM$(N)	Масив прізвищ (симв.)
12	IM$(N)	Масив імен (симв.)
13	OT$(N)	Масив по батькові (симв.)
14	POL$(N)	Масив ознак статі (симв.)
15	GOD(N)	Масив років народження (числ.)
16	OCM(N)	Масив оцінок з математики (числ.)
17	OCF(N)	Масив оцінок з фізики (числ.)
18	OCH(N)	Масив оцінок з хімії (числ.)

Схема алгоритму розв’язання завдання наведена на рисунку 5.5.

Рисунок 5.5, аркуш 1

Рисунок 5.5, аркуш 2

Програма

```
CLS:INPUT "Уведіть КІЛЬКІСТЬ ЗАПИСІВ  ТАБЛИЦІ: ", N
PRINT
DIM FAM$(N), IM$(N), OT$(N), POL$(N), GOD(N)
DIM OCM(N), OCF(N), OCH(N)
INPUT "Найменування групи: ", NG$
FOR I = 1 TO N
PRINT "Уведіть"; I; "- й ЗАПИС:"
INPUT "прізвище";FAM$(I)
INPUT "ім'я";IM$(I)
INPUT "по батькові";OT$(I)
INPUT "стать ";POL$(I)
INPUT "рік народження  ";GOD(I)
INPUT "оцінка з математики ";OCM(I)
INPUT "оцінка з фізики ";OCF(I)
INPUT "оцінка з хімії ";OCH(I)
PRINT
NEXT I
L$=STRING$(54,"-")
CLS:PRINT "Підсумки сесії групи ";NG$
PRINT
PRINT L$
PRINT "| № | Прізвище | Ім'я | По батькові | Стать | Рік н.
|Мат|Фіз|Хім|"
PRINT L$
FORM$="|###|\                    \|\          \|\             \|\
\|####|###|###|###|:"
FOR I = 1 TO N
    PRINT USING FORM$ ;I; FAM$(I), IM$(I), OT$(I), POL$(I),
GOD(I), OCM(I), OCF(I),OCH(I)
NEXT I
PRINT L$
PRINT
' Визначення наймолодшого студента
MAX=GOD(1)
FOR I=1 TO N
    IF GOD(I)>=MAX THEN MAX=GOD(I):IMAX=I
```

```
   NEXT I
   PRINT "НАЙМОЛОДШИЙ СТУДЕНТ"
   PRINT USING ="\                    \ !.!.  #### р.";FAM$(IMAX),
IM$(IMAX), OT$(IMAX), GOD(IMAX)
   ' визначення боржників
   PR=0
   FOR I=1 TO N
       IF OCM(I)<3 OR OCF(I)<3 OR OCH(I)<3 THEN
              PR=1
              PRINT "Боржники:"
   PRINT FAM$(I);" "; IM$(I));" " OT$(I));" ";POL$(I);"
";GOD(IMAX);" р.";" мат:"OCM(I);" фіз:"OCF(I);.";" хім:"OCH(I)
              PRINT
       END IF
   NEXT I
   IF PR=0 THEN PRINT "Боржників немає"
   ' Визначення відсотка чоловіків і жінок
   CM=0: CG=0
   FOR I=1 TO N
       IF POL$(I)="ж"   OR  POL$(I)="Ж" THEN CG=CG+1 ELSE
CM=CM+1
   NEXT I
   PM=(CM/N)*100: PG=(CG/N)*100
   PRINT "У групі ";NG$; " чоловіків ";PM; "%"; ", жінок ";PG;"%"
```

6 МОДУЛЬНЕ ПРОГРАМУВАННЯ

При рішенні багатьох практичних завдань виникає необхідність у багаторазовому обчисленні величин за тими ж самими алгоритмами при різних значеннях змінних.

Щоб програма була компактною, доцільно оформити ці обчислення у вигляді програмних модулів (процедур) і у відповідних місцях основної програми звертатися до них, задаючи необхідні для розрахунків значення параметрів.

Крім того, розбивка складного завдання на незалежні модулі є надзвичайно привабливою ідеєю, полегшуючи процеси створення й налагодження програми.

Самостійно модулі не виконуються. Спочатку виконується частина програми, що називається **основною (головною)** програмою. Інші модулі є **допоміжними** й викликаються в процесі виконання основної програми. Після виконання чергового модуля знову продовжує виконуватися основна програма (крім спеціальних випадків).

Модулі можуть викликати інші модулі. У програмі може бути будь-яка кількість модулів, які обмінюються значеннями не тільки з основною програмою, але й між собою.

Розглянемо, яку роль грають процедури в основній програмі, як їх створювати.

Процедури типу **subprogram** і **function** - є ефективним засобом модульного програмування для мов класу Basic.

Поряд із цим QBasic підтримує колишні засоби організації модулів - **subroutine** і **DEF FN**. Це дозволяє використовувати в QBasic програми, написані в старих версіях Basic.

6.1 Типи процедур. Основні визначення

Процедура - це закінчена частина програми, що призначена для рішення певного завдання. Будь-яка досить складна програма може бути розбита на основну програму й процедури, у яких вирішуються певні підзадачі.

QBasic дозволяє створювати два типи процедур - це **підпрограми** (**SUB**) і **функції** (**FUNCTION**). Різниця між ними полягає в способі виклику з головної програми й поверненні обчислених значень у головну програму.

Властивості процедур

Процедури характеризуються трьома основними властивостями:

- вони відділені від основної програми;
- оперують зі змінними, які використовуються тільки усередині процедури (локальні змінні);

- мають можливість одержувати інформацію з головної програми у вигляді параметрів або через загальну область і повертати обчислені значення назад.

Перша властивість робить становище процедури двояким: з одного боку, вона залишається частиною програми (якщо програма зберігається, процедура записується в той же файл), з іншого - існує як би незалежно: в *QBasic* для створення (редагування) різних процедур використовуються різні вікна.

Друга властивість означає, що, будучи незалежною частиною програми, процедура може використати свої власні змінні - вони не залежать від змінних основної програми й інші процедури. Так, змінні, які мають те саме ім'я (ідентифікатор) у головній програмі й у процедурі, - це різні змінні, які існують і використовуються незалежно одна від іншої.

Третя властивість процедур - це можливість вирішувати поставлене завдання з різними початковими даними, які їм передаються з головної програми.

Під **формальними** слід розуміти параметри, які вказуються в ***описі*** процедури. Вони є локальними змінними й з їхньою допомогою можна міняти значення змінних у процедурі.

Під **фактичними** слід розуміти параметри, які вказуються у ***виклику*** процедури. Вони є глобальними змінними, тобто тими реальними значеннями, з якими будуть зроблені обчислення.

6.1.1 Редагування процедур

Створення процедури

Існують два способи створення процедури:

- активізувати пункт головного меню «**Edit**» (Редагування).

Вибрати **New Sub**... або **New Function** (Нова Підпрограма або Нова Функція), при цьому відкриється діалогове вікно:

Увести ім'я нової процедури, наприклад, **Sub Sum(x,y**) і натиснути клавішу **[Enter]**

Після цього можна вводити оператори тіла процедури

Набрати текст основної програми;

• увести ключове слово **Sub** або **Function,** після якого вказати ім'я процедури й список формальних параметрів (якщо він є), наприклад, **Sub Sum(x,y**).

Відкриється нове вікно, у якому будуть два рядки: набрана Вами (**Sub Sum(x,y**)) і **End Sub** або **End Function.**

Після цього можна вводити оператори тіла процедури.

Ім'я процедури повинне бути унікальним і не повторювати імен інших процедур і змінних.

Збереження процедури

При збереженні програми (**Save** або **Save As**) *QBasic* записує тексти основної програми й усіх процедур в один файл. При цьому він автоматично додає в початок основної програми оператори **DECLARE**, які повідомляють про наявність процедур. Таким чином, основна програма завжди буде починатися з декларації всіх процедур (підпрограм і функцій), які викликаються нею.

Перегляд списку процедур

Для перегляду списку процедур, які використовуються головною програмою, треба активізувати пункт меню «**View**» (Перегляд), вибрати пункт «**SUBs**» і натиснути клавішу **Enter**. Більш просто це можна зробити, натиснувши функціональну клавішу **F2**. При цьому з'явиться діалогове вікно зі списком всіх програмних елементів, включаючи й головну програму.

Виконати багатомодульну програму можна, натиснувши клавішу **F5**, перебуваючи в кожній із процедур.

6.2 Процедури–функції або функції користувача (FUNCTION)

Поряд зі стандартними вбудованими функціями (SIN, COS і ін.), *QBasic* надає можливість користувачеві створювати свої функції виду **F(x)**, де **F** - функція; **x** - список аргументів.

Як і стандартні вбудовані функції, функції користувача, отримують аргументи через список параметрів і повертають в основну програму обчислене значення.

Функція повертає єдине значення в основну програму, тому всі функції повинні містити хоча б один оператор присвоювання якого-небудь значення імені функції. Присвоєння може відбуватися кілька разів при наявності, наприклад, умовного оператора, але при кожному виклику функція повертає тільки один результат.

Функція обчислюється в момент звертання до неї за іменем
ім'я [(фактичні параметри)]

Загальна форма запису
FUNCTION ім'я [(формальні параметри)] [**STATIC**]
[блок операторів]
ім'я = вираз
[EXIT FUNCTION] } тіло функції
[блок операторів]
END FUNCTION

де **ім'я** - ім'я функції користувача. Тип даних, які повертаються нею, може бути визначений явно суфіксом типу даних (%, &, !, # або $);

- ***формальні параметри*** - одна або кілька змінних, які вказують параметри, передані у функцію при її виклику.

Цей список має вигляд:

- ***змінна*[( )] [AS *тип*] [, *змінна*[( )] [AS *тип*]]...;**
- ***змінна*** - ім'я змінної *QBasic*;
- ***тип*** - тип змінної (*INTEGER, LONG, SINGLE, DOUBLE, STRING* або тип даних, які визначені користувачем);
- **STATIC** – вказує на те, що значення локальних змінних зберігаються між викликами функції;
- ***вираз*** – значення функції, що повертається; воно присвоюється імені функції як локальної змінної й повинне бути з ним одного типу (за замовчуванням це дійсний тип).

Пункти, узяті у квадратні дужки, необов'язкові.

Кожна процедура повинна мати початок і кінець. Оператором **END FUNCTION** завершується робота процедури, і управління передається оператору основної програми, розташованому за оператором виклику даної процедури.

Оператор **EXIT FUNCTION** дозволяє перервати виконання процедури й передати управління в основну програму.

При виклику функції **можна вказати**, що **значення аргументу (параметра) не повинне змінюватися функцією**. Для цього його треба взяти в круглі дужки.

Допоміжні змінні в процедурі мають локальний характер. Для них автоматично резервується пам'ять, і їх значення в основну програму не повертаються.

Команда **SHARED <*список змінних*>**

визначає зазначеним у списку змінним ті ж поля пам'яті, що були призначені головною програмою; у такий спосіб значення цих змінних стають доступними головній програмі (ці змінні не повинні бути параметрами).

Якщо в програмі зустрічається звертання до функції, то виконуються такі дії:

1) обчислення значень фактичних параметрів;
2) звертання до команд опису даної функції;
3) присвоювання формальним параметрам значень фактичних і виконання тіла функції;
4) присвоювання необхідного значення імені функції;
5) повернення обчисленого значення в програму, яка викликала функцію.

Приклад

Розглянемо використання функції, визначеної користувачем, на прикладі обчислення числа можливих сполучень.

$$C_n^m = \frac{n!}{m!(n-m)!}$$

Як видно з формули, треба тричі обчислювати значення факторіала, для чого визначимо функцію користувача Fact () і тричі використаємо її в основній програмі.

```
REM Основна програма
’Оголошення функції
DECLARE FUNCTION FACT% (K AS INTEGER)
DIM M AS INTEGER, N AS INTEGER: ’Оголошення змінних
INPUT ’’Задайте n, m’’,n,m :‘Завдання початкових даних
'Фактичними параметр. при виклику функції Fact() є n, m і n-m
C = Fact%(n) / (Fact%(m) * Fact%( n-m)):’Обчислення формули
PRINT ’’Кількість сполучень = ’’,C:’Друк результату
END
'Кінець основної програми

FUNCTION FACT% (K AS INTEGER): ’Опис функції
'K - формальний параметр функції
REM Функція обчислення факторіала
DIM I AS INTEGER, F AS INTEGER
IF K<0 THEN PRINT “факторіал ”; K; “ не існує”:STOP
IF K = 0 THEN
Fact%=1: ’Присвоювання значення, що повертається
ELSE
F = 1:'Обчислення факторіала
FOR I = 1 TO K
F =F * I
NEXT I
FACT% = F:          ’ Присвоювання значення, що повертається
END IF
END FUNCTION
```

6.3 Підпрограми (SUB)

Під підпрограмою розуміється процедура, що викликається з основної програми за допомогою оператора виклику **CALL**. У головну програму при цьому можна повертати не одне (як FUNCTION), а кілька обчислених значень.

Загальна форма опису підпрограми SUB:

SUB ім'я[(параметри)] **[STATIC**]
[блок операторів]
[EXIT SUB] } тіло підпрограми
[блок операторів]
END SUB

ім'я - ім'я підпрограми SUB, довжиною до 40 символів, без суфікса типу даних.

Приклад

З використанням процедури-підпрограми підраховується вартість телефонної розмови із щохвилинною оплатою: 0.22 грн+20% податку

```
DECLARE SUB CINA (K,C)
CLS
INPUT “Кількість хвилин розмови з містом Київ ===> “,K1
CALL CINA (K1,C)
PRINT “Вам необхідно сплатити за розмову з Києвом ==>”;C; “грн”
INPUT “Кількість хвилин розмови зі Львовом ===> “,K2
CALL CINA (K2,C)
PRINT “ Вам необхідно сплатити за розмову зі Львовом ===>”;C; “грн”
END
SUB CINA (K,C)
C=0.22*N
C=C+0.2*C
END SUB
```

Використання ключового слова **STATIC** робить всі локальні змінні статичними, тобто вони не будуть ініціалізуватися знову при наступному виклику даної підпрограми.

Приклад

```
DECLARE SUB CALC
FOR TIME%=1 TO 4
    CALL CALC
NEXT TIME%
END

SUB CALC STATIC
NUMBER%=NUMBER%+1
PRINT NUMBER%
END SUB
```

Результат виконання програми зі STATIC	Результат виконання програми без STATIC
1	1
2	1
3	1
4	1

На рисунку 6.1 основна програма викликає процедуру – підпрограму, а та у свою чергу - процедуру-функцію.

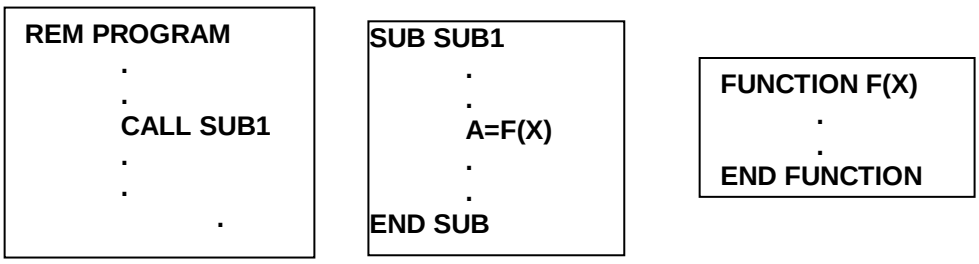


Рисунок 6.1

6.3.1 Передача параметрів у процедуру

Необхідно пам'ятати два основні правила передачі параметрів у процедури (SUB і FUNCTION):

- кількість фактичних параметрів при звертанні до процедури повинна дорівнювати кількості формальних параметрів у її оголошенні;
- відповідні параметри повинні бути одного типу.

У дужках при виклику процедури можна вказувати як константи (числові, строкові), так і значення змінних або самі змінні. Тому існують два основних засоби передачі параметрів - за значенням і за посиланням.

6.3.2 Передача параметра за посиланням

Як фактичний параметр передається не значення змінної, а її адреса. Якщо значення формального параметра змінюється в процедурі, то фактичний параметр набуває цього значення.

Таким чином, процедура отримує доступ до елемента пам'яті, у якому зберігається значення змінної, і може змінити її значення.

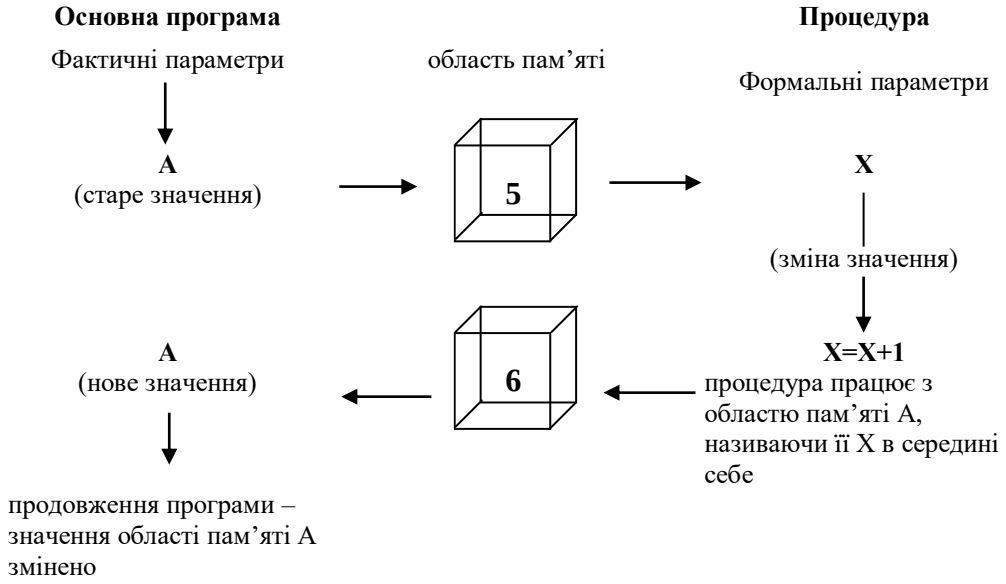


Рисунок 6.2

6.3.3 Передача параметра за значенням

Якщо процедура не повинна змінювати значення змінних, що передаються їй, то в цьому випадку параметри передаються за значенням (у процедуру передається копія значення змінної, яка зберігається в тимчасовій області пам'яті). Для цього відповідний параметр при виклику процедури береться в круглі дужки.

Якщо фактичний параметр є константою, QBasic не може передати його по посиланню, тому що він не має адреси.

Якщо за значенням передається змінна, то вона є локальною змінною процедури, якій присвоєне значення фактичного параметра. Тому зміна цієї локальної змінної не впливає на значення змінної в основній програмі (рисунок 6.3).

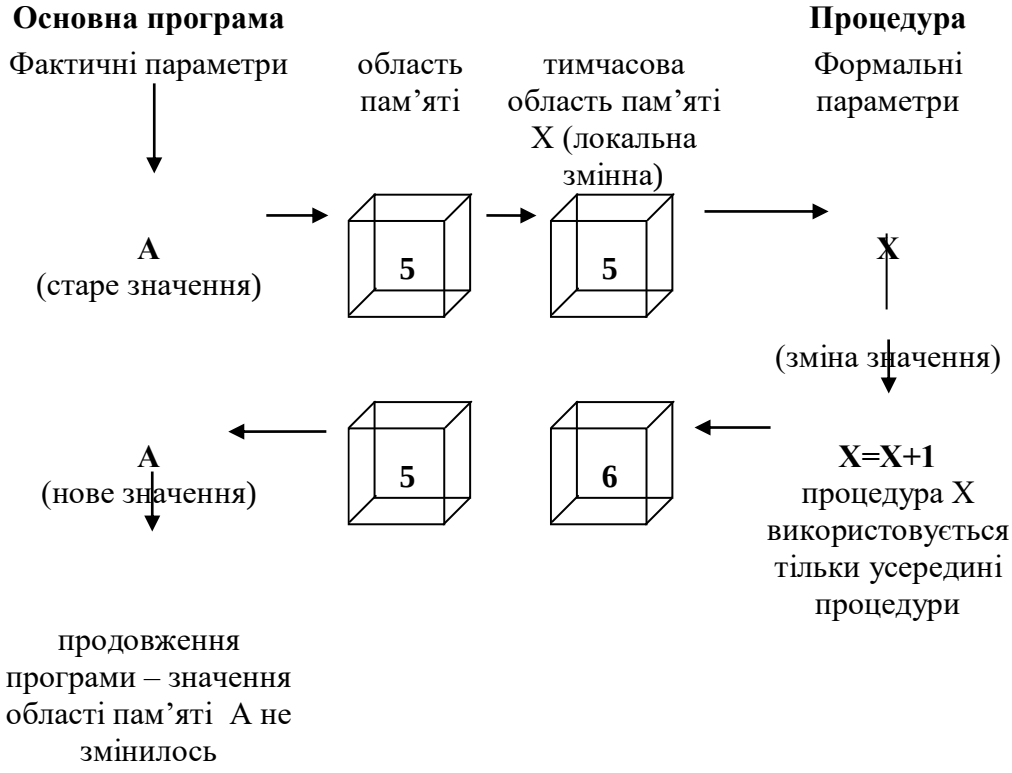


Рисунок 6.3

6.3.4 Передача масиву

Масив або елемент масиву завжди передаються в процедуру за посиланням. Ім'я масиву або елемента завжди вказує адресу фактичного параметра, що дозволяє процедурі змінювати їхні значення.

Наявність порожніх дужок після імені вказує, що передається масив, а не скалярна змінна. При цьому формальний параметр теж повинен містити круглі дужки.

Якщо потрібно передати частину масиву, то потрібно додати параметри, що вказують границі цієї частини.

Якщо потрібно передати елемент масиву - то вказати номер елемента в дужках після імені масиву.

Стандартні функції LBOUND(A) і UBOUND(A) визначають найбільше й найменше значення індексів масиву A.

Приклад

```
SUB P2(A(),S)
S=0:FOR I= LBOUND(A) TO UBOUND(A)
      S=S+A(I)
NEXT I
END SUB
```

Приклад

Знаходження мінімального й максимального елементів масиву.

У головній програмі здійснюється введення вихідного масиву, виклик підпрограми й друкування повернутих з підпрограми значень максимального й мінімального елементів масиву.

```
DECLARE SUB MINMAX (A!(), N AS INTEGER, MAX!, MIN!)
DIM K AS INTEGER, I AS INTEGER
CLS
INPUT “Кількість елементів К= “,K
DIM Y(K)
FOR I=1 TO K
PRINT “Y(“;I;”)=”;
INPUT “,Y(I)
NEXT I
CALL MINMAX(Y(), K, YMAX, YMIN)
PRINT “YMAX=”;YMAX, ”YMIN=”;YMIN
END
SUB MINMAX(A(),N AS INTEGER, MAX, MIN)
DIM I AS INTEGER
MAX=A(1)
MIN=A(1)
FOR I=1 TO N
IF A(I)>=MAX THEN MAX=A(I)
IF A(I)<=MIN THEN MIN=A(I)
NEXT I
END SUB
```

6.4 Підпрограми типу SUBROUTINE і DEF FN-ФУНКЦІЇ

Розглянемо форму організації підпрограм і функцій, що використовувалася в ранніх версіях Basic і яка підтримується QBasic.

6.4.1 Використання підпрограм типу SUBROUTINE – САМОСТІЙНО

Такі підпрограми

- є частиною основної програми;
- не мають локальних змінних;
- не можуть приймати параметрів.

У редакторі QBasic не мають спеціального вікна.

Їх скоріше можна розглядати як відособлену групу операторів усередині основної програми. Вони використовують тільки змінні основної програми. Іншими словами, поле пам'яті основної програми й підпрограми спільні й всі величини, визначені в основній програмі, можуть використовуватися й у підпрограмі. Необхідно стежити, щоб значення тих змінних, які не повинні змінюватися в основній програмі, не змінювалися в підпрограмі. Передача значень параметрів у підпрограму й повернення результатів або не вимагають ніяких дій, або здійснюються за допомогою операторів присвоювання.

Структура звертання до **Subroutine**
GOSUB мітка,

де мітка - мітка або номер рядка програми.

Якщо в рядку програми кілька операторів, то **GOSUB** повинен бути останнім. **Subroutine** повинна починатися з рядка, що має мітку або номер (рисунок 6.4).

Вона може бути викликана довільну кількість разів і з будь-якого місця програми, наприклад, із циклу або з іншої підпрограми типу **Subroutine** Логічний кінець **Subroutine позначається** оператором **RETURN, що** має формат:

RETURN
або
RETURN рядок,

де рядок - мітка або номер рядка програми.

Ця команда аналогічна **END SUB** для підпрограм. Якщо рядок в операторі **RETURN** не зазначений, то управління передається в точку виклику **Subroutine**.

При наявності після слова **RETURN** мітки або номера рядка управління передається на відповідний рядок.

Оператор **GOSUB** передає управління аналогічно **GOTO.** Однак при використанні **GOSUB** QBASIC відзначає точку програми, з якої відбувається звертання до підпрограми типу **Subroutine,** і може повернути в неї управління після виконання відповідного блока операторів.

Приклад

Розглянемо наведене раніше завдання обчислення кількості сполучень

```
INPUT “N=”; N: INPUT “M=”; M
K=N : GOSUB 200
Y=X
K=M : GOSUB 200
Z=X
K=N-M : GOSUB 200
C=Y / (Z*X)
PRINT “число сполучень із” ; N; “по” ; M; “дорівнює ”; C
STOP
200 REM підпрограма обчислення факторіала
IF K<0 THEN 280
IF K=0 THEN X=1 : GOTO 270
X=1
FOR I=2 TO K
        X=X*I
NEXT I
```

```
270 RETURN
280 PRINT “факторіал”; K; “не існує”
```

В основній програмі тричі звертаємося до підпрограми. При цьому параметру **K** присвоюються щоразу нові значення N, M, N-M. Щоб мати можливість використати результати роботи підпрограми після всіх трьох звертань, перед кожним новим звертанням до підпрограми результат попередніх звертань X міняємо на Y і Z.

На рисунку 6.4 наведена загальна структура програми, що містить підпрограми типу Subroutine.

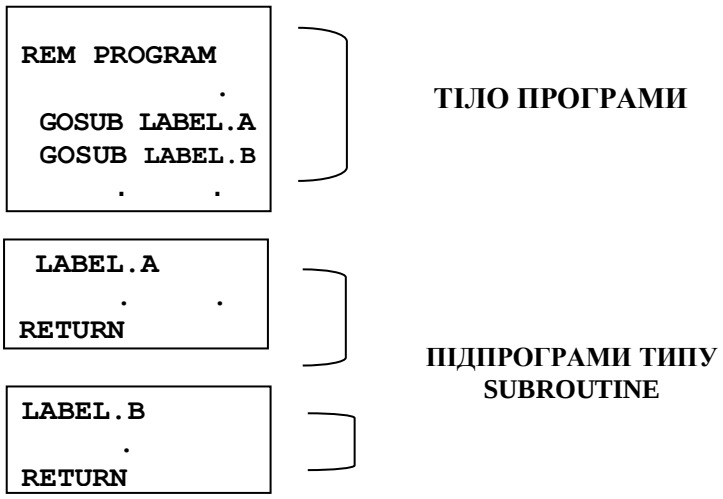


Рисунок 6.4

Додатковою можливістю виклику підпрограми є використання оператора типу

ON вираз GOSUB список,

де **вираз** – числовий вираз

список – мітки або номери рядків програми, розділені комами.

Цей оператор аналогічний операторові **ON GOTO** (якщо ціла частина виразу дорівнює 1, буде викликана підпрограма, номер рядка якої зазначений першим, якщо дорівнює 2, то другим і т.д.). Після закінчення роботи будь-якої підпрограми

основна програма виконується з рядка, що стоїть за **ON GOSUB,** як і у випадку, якщо ціла частина виразу більше числа міток у списку. Якщо ціла частина виразу менше 0, виникає помилкова ситуація.

6.4.2 Використання функцій типу DEF FN (НЕСТАНДАРТНІ ФУНКЦІЇ КОРИСТУВАЧА)

DEF FN є альтернативною формою функції, розробленої користувачем.

Ці функції можуть викликатися тільки в програмі, де вони описані. QBasic підтримує два різновиди функції типу **DEF FN:** однорядкову й багаторядкову.

Подібно FUNCTION, **DEF FN** повертає значення в основну програму. Однак при цьому є невід'ємною частиною основної програми.

Формат функції:
Однорядкова (проста форма):
DEF FNім’я (параметри) = вираз

Багаторядкова **(складна форма):**
DEF FNім’я (параметри)

оператори
FNім’я =вираз

оператори
END DEF,

де **ім'я** – ім'я функції;
параметри – список формальних параметрів;
вираз – вираз, значення якого присвоюється функції;
оператори – тіло функції.

FNім’я використовується для визначення функції й повинне характеризувати який-небудь тип даних. Параметри не можуть містити масиви або строкові константи. Тип виразу повинен відповідати типу функції.

У багаторядковій формі DEF FN для переривання виконання функції використовується оператор EXIT DEF, аналогічний за своєю дією операторові EXIT FUNCTION.

Звертання до нестандартної функції

FNім’я (параметри),

де **ім'я** – ім'я функції;
параметри – список фактичних параметрів

При звертанні до функції обчислюється значення виразу в операторі **DEF**, при цьому формальні параметри заміняються відповідними фактичними значеннями, які повинні відповідати формальним за кількістю, порядком слідування й типом.

Приклад

Обчислити $\omega = \sqrt{x^2 + y^2 + \sin^2 xy} + \sqrt{y^2 + z^2 + \sin^2 yz} + \sqrt{z^2 + x^2 + \sin^2 xz}$
при x=21,87 y=35,13 z=9.87

```
DEF FNF (A, B) =SQR(A*A+B*B+Sin (A*B)(2)
DATA 21.87, 35.13, 9.87
READ x, y, z
W=FNF (x,y)+FNF (y,z)+FNF (z,x)
```

Приклад, у якому однорядкова функція викликається в операторі PRINT.

```
DEF FNVS (A!, B!) = (A! + B!)/2
PRINT FNVS (8.5, 44.)
```

7 ФАЙЛИ НА МАГНІТНИХ НОСІЯХ

Багато програм використовують дані, що вводяться із клавіатури: обробляють їх, зберігають у масивах і змінних, виводять безпосередньо на екран. Однак у всіх програмах дані зникають із пам'яті комп'ютера, як тільки програма закінчує свою роботу.

Основним способом збереження інформації, отриманої в ході виконання програми, служить запис її на тверді або гнучкі диски. Це особливо важливо, якщо обсяг інформації великий, а дані передбачається використати неодноразово в цій або в інших програмах.

Використання файлів дає такі переваги:

1) можливість зберігати дані на зовнішніх носіях тривалий час, а не тільки під час виконання програми;

2) кількість даних може бути більшою, заздалегідь невідомою, тобто під час роботи з файлом не потрібно знати число записів у ньому;

3) дані в записі можуть бути різних типів;

4) дані зберігаються окремо від програм, які їх обробляють, і можуть бути використані різними програмами й користувачами;

5) між даними різних файлів може бути певна відповідність (релятивістські зв'язки), що дасть можливість не дублювати ті самі дані в різних файлах.

Інформація, розташована на зовнішньому носії, що має певну логічну структуру й ім'я, називається ***набором даних***. Він може містити як програми на вхідних мовах, так і оброблені дані (каталог книг бібліотеки, відомості про абітурієнтів ВНЗ й т.п.).

Набір даних складається з *фізичних записів*.

Фізичний запис – найменша порція даних, що переноситься із МД (магнітного диска) або на МД за одну фізичну операцію введення-виведення. Вона являє собою одиницю носія - сектор МД (512 Б).

Файл – символічне подання набору даних як сукупності *логічних записів* однакової структури.

Логічний запис – змістовна одиниця збереженої у файлі інформації, що є сукупністю відомостей про деякий об'єкт.

Приклад 1

Файл - список студентів групи, логічний запис містить поля: прізвище, ім'я студента, рік народження.

Розмір логічного запису залежить від розв'язуваного завдання. Файл зв'язується з набором даних при так званому **відкритті** файлів у програмі.

Кількість записів у файлі може бути довільною. За останнім записом знаходиться службова мітка кінця файла (код 26). Її записує у файл QBasic. Файл може бути порожнім, *тобто в ньому не буде жодного запису*, але ім'я файла й мітка кінця будуть обов'язково. Записи можуть містити дані різних типів.

Приклад 2 (див. приклад 1)

" Іванов", "Петро", 1986

Такий запис містить 3 поля-параметри:

- прізвище - текстовий тип;
- ім'я - текстовий тип;
- рік народження - цілий числовий тип.

Файли розрізняються відповідно до методу доступу до інформації в них, незалежно від типу збережених у них даних. Метод доступу визначає порядок, у якому дані записуються у файл або читаються з файла.

Підтримуються такі *методи доступу* до інформації у файлах:

- послідовний доступ (текстові файли);
- прямий (довільний) доступ.

Файли *послідовного доступу* дозволяють прочитати деякий запис, тільки прочитавши всі попередні. Їх можна порівняти з аудіо- або відеозаписами на магнітофонній стрічці: для пошуку потрібного місця. Ви змушені перемотувати стрічку й послідовно її переглядати або прослуховувати.

Інформація в них зберігається в текстовому форматі (у вигляді ланцюжка кодів ASCII).

Записи в такому файлі легко проглядаються просто з екрана. Такі файли можуть оброблятися будь-якими текстовими редакторами так само, як і самим QBasic. Але, оскільки інформація зчитується й зберігається послідовно, це робить процес обробки інформації в них більш тривалим у порівнянні з обробкою файлів інших типів.

Файли *прямого доступу* зберігають інформацію в спеціальному форматі. Кожен запис займає певну (однакову для всіх записів!) довжину, тому файли з таким доступом можуть займати більше місця на диску, зате робота з ними відбувається швидше. Звертання до записів у таких файлах здійснюється **"прямо**" - без читання попередніх даних, за номером запису (шифр книги в бібліотеці). Записи можуть бути будь-якої довжини до 32767 байт.

Будь-який файл можна організувати так, щоб мати або прямий, або послідовний доступ, але не обидва разом.

Якщо при кожному звертанні до файла використовуються майже всі дані, а міняти їхній зміст часто не передбачається, то доцільно вибрати послідовний доступ. Коли потрібно часто міняти вміст записів і переглядати їх у довільному порядку - прямий доступ.

Крім цього, QBasic дозволяє звертатися до файла як до послідовності байтів і оперувати безпосередньо байтами. Такий спосіб доступу до інформації називається **бінарним** (теж прямого доступу, обіг за номером байта).

Будь-які дії з файлами, які виконуються в програмі, містять у собі такі обов'язкові кроки:

- відкриття файла;
- читання й запис оброблюваних даних;
- закриття файла.

7.1 Файли послідовного доступу (текстові файли)

Щоб **створити** послідовний файл, необхідно:

1) відкрити файл оператором **OPEN** для виведення або дозапису в нього даних;
2) писати дані у файл операторами **PRINT** або **WRITE;**
3) закрити файл оператором **CLOSE**.

Щоб **прочитати** послідовний файл необхідно:

1) для доступу до даних, відкрити його оператором **OPEN** для виведення з нього даних;
2) читати дані з файла в програму операторами **INPUT**, **LINE INPUT** або **INPUT$**;
3) закрити файл оператором **CLOSE**.

Можна відкрити кілька файлів одночасно, щоб читати, наприклад, інформацію з одного файла, обробляти її й потім зберігати в іншому. Але не можна використовувати відкритий файл і для читання, і для запису інформації одночасно, тому що ці дії пов'язані з різними режимами відкриття файла.

Для **відкриття** файла використовується оператор **OPEN**, що має такий формат:

OPEN *файл$* FOR *режим* AS *#номер_файла %*,

де ***файл$*** - ім'я файла;
режим – режим роботи з даним файлом;
- необов'язковий знак, що випереджає номер файла;
номер_файла % (дескриптор) - ціле число від 1 до 255.

Приклад 3

OPEN "Рік народження.txt " FOR OUTPUT AS #5

Параметри оператора **OPEN**

• Параметр **файл$.** Кожний файл повинен мати ім'я, складене за певними правилами, щоб операційна система

комп'ютера знала, де шукати той або інший файл. Тому повне ім'я файла включає не тільки безпосередньо назву, але й найменування дискового пристрою й директорії, де знаходиться або буде створений файл.

Якщо файл, необхідний вам у програмі, знаходиться на тому ж диску, що й QBasic, але в іншій директорії, необхідно однаково вказувати повний шлях до нього.

Приклад 4

Якщо файл Рік народження**.txt** знаходиться в піддиректорії **MY_DATA** директорії **DOS** диска **З:** він може бути відкритий у програмі за допомогою такого оператора:

OPEN "C:\DOS\MY_DATA Рік народження**.txt " FOR INPUT AS #2**

Найпростіший спосіб роботи з файлами - зробити поточною ту піддиректорію, у якій перебувають файли, що цікавлять вас, і потім просто вказувати їхні імена.

Як параметр **файл$** можна використати змінну текстового типу, якій присвоєне ім'я файла. Це дозволяє вводити ім'я файла з клавіатури й, таким чином, використовувати програму для роботи з різними файлами. У цьому випадку оператор відкриття файла набуває виду:

Приклад 5

```
INPUT "Введіть ім'я файла: ", FileNames$
OPEN FileNames$ FOR APPEND AS #1
```

Такий метод відкриття файлів є найбільш універсальним, оскільки він простий й зручний для користувача.

- Параметр ***режим*** визначає режим роботи з файлом, тобто вказує QBasic, як ви збираєтеся цей файл використати: для читання або запису в нього інформації. Цей параметр може мати одне з таких значень:

OUTPUT – відкрити новий файл для запису в нього інформації. Якщо відкрити в цьому режимі вже існуючий файл, його вміст буде стерто;

APPEND – відкриває існуючий файл для додавання в нього нової інформації. У такому режимі нова інформація завжди поміщається в кінець файла, після останнього запису. Якщо вказати ім'я файла, якого ще не існує, буде відкритий новий файл, як це робиться в режимі OUTPUT;

INPUT – відкриває існуючий файл для читання збереженої в ньому інформації. У цьому випадку, якщо вказати ім'я неіснуючого файла, піде повідомлення про помилку File not found (Файл не знайдений).

- Параметр **номер_файла%** присвоює файлу, що, відкривається певний номер для більш зручного використання в програмі.

У прикладі 4 присвоюється файлу з ім'ям **Рік народження.txt** номер 2. Тепер даний номер не можна буде присвоїти ніякому іншому файлу, поки **Рік народження.txt** відкритий.

Щоб працювати з файлами, потрібно розуміти, як зв'язується операційна система з файлом. Для цього є канал введення-виведення. При відкритті файла ставиться у відповідність канал з певним номером. Отже, при роботі з файлом має значення не ім'я файла, а номер каналу. Операційна система повинна мати відомості про наявність вільних каналів.

Максимальне число одночасно відкритих файлів у програмі задається і залежить від конфігурації операційної системи. Можна змінити число файлів, що відкриваються, за допомогою **команди** FILES у **файлі CONFIG.SYS**, що визначає конфігурацію вашої системи.

Для запобігання хаосу не слід відкривати занадто багато файлів одночасно. Як тільки закінчена робота з яким-небудь файлом, відразу треба його закрити.

Для **занесення даних** у файл (формування запису на зовнішньому носії) використовують команди **PRINT** і **WRITE**.

Формат операторів:

PRINT #номер_файла%, [список значень]
WRITE # номер_файлу%, [список значень]

список значень - список констант або/і змінних, значення яких записуються у файл. Якщо список значень відсутній, у файл буде занесений порожній рядок.

Логіка роботи цих операторів різна.

Оператор **WRITE**

Роздільником у списку значень є **кома.** Елементи списку заносяться у файл в один текстовий рядок. Символьні дані (String) записують у лапках. Ця команда сама заносить коми між полями й більш ощадливо розташовує дані на носії. Після запису останнього елемента рядка автоматично записуються символи “повернення каретки” (13) і “перехід на новий рядок” (10). Дані можна прочитати оператором INPUT

Оператор **PRINT**

Роздільниками в списку значень є:

кома - значення записуються в 14-символьні зони виведення (після кожного даного автоматично заноситься символ табуляції);

крапка з комою - значення записуються підряд, без проміжків між ними.

Оператор **PRINT** зручний для ретельного редагування тексту вихідного файла, а оператор **WRITE** краще застосовувати в тому випадку, коли вихідний файл буде використовуватися надалі як вхідний для інших програм.

Приклад 6

Програма запису у файл із ім'ям Рікнародження.txt таких даних (рядків) (див. приклад 1, 2)

Іванов	Петро	1986
Сурина	Марина	1987

```
Dim Fam As String, Im As String, God As Integer
Open “ Рікнародження.txt” for Output As#1
For i=1 to 2
        Input “Прізвище”; Fam
        Input “Ім'я”; Im
        Input “Рік народження”; God
        Write #1, Fam, Im, God
Next i
Close #1
```

“Іванов”, “Петро”, 1986
“Сурина”, “Марина”, 1987

Замінимо **Write #1, Fam, Im, God** оператором **Print #1, Fam, Im, God**

```
1               15              30
Іванов          Петро           1986
Сурина          Марина          1987
```

А тепер оператором **Print #1, Fam; Im; God**

```
1       7       12
ИвановПетро 1986
СуринаМарина 1987
```

Можна вставити коми в лапках у список значень **PRINT**, щоб розбити запис на більш дрібні частини й забезпечити доступ до цих частин (у майбутньому) командою читання.

Дані можна записувати не тільки в змінні, але й у заданий блок оперативної пам'яті.

Для **закриття** файла використовується оператор:

CLOSE [#номер_файла]

Якщо вказується номер файла, то буде закритий саме цей файл.

Оператор **CLOSE без параметра закриває всі файли**, відкриті в цей момент у програмі.

Читати дані можна різними способами:

- за допомогою операторів **INPUT** і **LINE INPUT;**
- за допомогою функції **INPUT$.**

Формат цих операторів

INPUT# номер_файла%, Список змінних

Список змінних – записані через кому змінні будь-якого типу. Змінні списку набувають відповідних значень полів запису файла.

LINE INPUT# номер_файла%, змінна

змінна – змінна типу String

Зчитує рядок до 255 символів з файла (поки не зустріне символ "ВК") і присвоює з**мінній** це значення

INPUT$(n[,#]номер_файла%])

n – число символів (байтів) для читання

змінна – змінна типу String

Зчитується певна кількість символів із зазначеного файла

Приклад 7: (див. приклад 6) **INPUT** Do Until EOF(1) Input #1, Fam, Im, God Loop Close #1	***Приклад*: LINE INPUT** Do Until EOF(1) Line Input #1, text$ Loop Close #1	***Приклад:*** **INPUT$** Input$ (LOF(1),#1) Close #1

EOF *(номер_файла)* - повертає "ІСТИНА" при досягненні кінця файла.

LOF *(номер_файла)* **–** повертає довжину файла.

7.2 Файли довільного доступу

Щоб створити довільний файл, необхідно:

1) відкрити файл оператором **OPEN**, указуючи довжину запису.

Формат

OPEN файл$ FOR Random [Access Доступ][блокування] AS #номер_файла % Len=Довжина_запису

На відміну від текстових файлів тут не робиться розходження між файлами для запису й для читання: всі вони відкриваються в одному режимі, що визначається ключовим словом **Random.**

Крім цього, потрібно вказати довжину запису атрибутом **Len:** якщо це значення менше реальної довжини запису, то виникає помилка, якщо більше, то при записі файла використовується більше дискового простору, ніж необхідно.

Існує можливість відкрити файл із видом доступу “тільки читання” або забороною читання й запису в нього інформації за допомогою програм інших комп'ютерів, або встановити захист на окремий запис за номером.

2) оператором **FIELD** розподілити простір у буфері прямого доступу для змінних, значення яких будуть записані у файл. При цьому запис буде мати необхідну структуру.

Формат оператора:

FIELD #*номер_файла* pole1 As Змінна1[,pole2 As Змінна2...]

Сумарна довжина всіх перерахованих полів повинна збігатися з обсягом буфера, оголошеним при відкритті файла.

3) всі числові дані, які записуються у файл довільного доступу, попередньо повинні бути перетворені до символьного типу. Цей переклад виконують функції **MKI$, MKL$, MKS$, MKD$** (Make...). Значення аргументів не повинні виходити за діапазон відповідних числових типів. Довжина рядків, що

повертаються, 2, 4, 4, 8 відповідно. Тільки після такого перекладу дані можна помістити в буфер.

LSET і **RSET** переміщають дані в буфер прямого доступу, підготовляючи його до запису у файл, і здійснюють відповідно ліве або праве вирівнювання строкових значень;

4) після заповнення буфера його вміст можна перенести у файл оператором **PUT**.

Формат оператора:

PUT *#номер_файла% [. номер запису][,змінна]*

#номер_файла% - номер відкритого файла;

номер запису - номер запису для запису;

змінна - містить дані для запису у файл;

5) закрити файл оператором CLOSE.

Приклад 8

Для прикладу 6, у припущенні, що прізвище містить до 15 символів, ім’я - до 10 символів, рік народження -4 цифри

```
Dim Fam As String, Im As String, God As Integer
Dim Fam1As String, Im1 As String, God1 As String
Open “Рікнародження.txt” for FOR Random AS #1 % Len=27
    FIELD #1 15 As Fam1, 10 As Im1, 2 As God1
For i=1 to 2
    Input “Прізвище”; Fam
    Input “Ім'я”; Im
    Input “Рік народження”; God
    Write #1, Fam, Im, God
        LSET Fam1, Fam
        LSET Im1, Im
        LSET God1, MKI$(God)
    PUT #1, Fam1,Im1,God1
Next i
    CLOSE #1
```

Щоб прочитати довільний файл, необхідно:

1) відкрити файл оператором **OPEN** вказуючи довжину запису;
2) оператором **FIELD** розподілити простір у буфері прямого доступу для змінних, значення яких будуть читатися з файла;
3) помістити необхідний запис у буфер оператором **GET**.

Формат оператора **GET:**
GET *#номер_файла% [. номер запису][,змінна]*
#номер_файла% - номер відкритого файла;
номер запису - номер запису для читання;
змінна - містить дані для прийому введення з файла.

Дані в буфері доступні програмі. Строкові дані при необхідності повинні бути перетворені в числові функціями **CVI, CVL, CVS, CVD (Convert...)**;
4) закрити файл оператором **CLOSE.**

Відзначимо такі корисні оператори:

Оператор **перейменування** файла **NAME *Старе Ім'я* AS *Нове ім'я***
Оператор **видалення** файла **KILL Ім'я файла, що видаляється**

8 ОБРОБКА СИМВОЛЬНИХ ДАНИХ

Для розуміння символьних і текстових функцій необхідно попередньо згадати кодову систему, яка використовується комп'ютерами.

8.1 Одержання ASCII – коду символу й одержання символу, що відповідає ASCII- КОДУ

Для виконання цих перетворень в QBasic передбачені такі функції:

ASC (*текст***)** – повертає код ASCII для першого символу текстового значення;

CHR$ (*код***)** – повертає символ, що відповідає коду ASCII.

Параметр *код* повинен мати значення від 0 до 255 і бути одним з ASCII - кодів.

Приклад	***Результат***
PRINT "Символ з кодом ASCII 81 - це "; CHR$(81) PRINT "Код ASCII символу "Q" - це "; ASC ("Q")	Символ з кодом ASCII 81- це Q Код ASCII символу "Q" - це 81

8.2 Введення символів у програму

Оператор **INPUT** дозволяє ввести символьні вирази будь-якої припустимої довжини (у лапках або без них), при цьому завершення введення відбуваються при натисканні клавіші **ENTER**.

При використанні функції **INPUT$** ви не повинні натискати клавішу **ENTER** після введення символу – введення завершується відразу після одержання символу.

Функцію **INPUT$** можна застосовувати й для введення в програму декількох символів.

Її загальний вид:

INPUT$ (*число*),
де *число* – число символів для введення.

Символи, що вводяться, не відображаються на екрані.

Приклад

```
PRINT "Введіть 2 літери"
COD$=INPUT$(2)
PRINT "Введені літери";COD$
```

Оператор **LINE INPUT** дозволяє ввести довільний текстовий рядок до 256 символів, включаючи коми. Можна вводити будь-які символи, не перевіряючи, чи є вони літерами.

Приклад

```
PRINT "Введіть будь-який текст"
LINE INPUT COD$
PRINT "Введений  текст";COD$
Введіть будь-який текст
Нг89б;;.?fs
Введено  текст Нг89б;;.?fs
```

8.3 Підтримка інтерфейсу між програмою й клавіатурою (INKEY$)

Припустимо, необхідно створити паузу очікування в певному місці вашої програми, наприклад, щоб мати час для прочитання великого повідомлення (це може бути вікно з рекомендаціями з користування програмою). Звичайно в подібних ситуаціях програма чекає натискання певної або довільної клавіші.

Функція INKEY$ аналізує системну інформацію про натиснуті клавіші й може бути використана для створення паузи довільної тривалості. Вона повертає порожній рядок, якщо не була натиснута жодна клавіша, або рядок з одного символу при натисканні звичайної клавіші, або рядок із двох символів при натисканні клавіші з розширеним кодом (128-255). При цьому символ, що вводиться із клавіатури, на екран не виводиться.

Щоб організувати таку паузу, потрібно використати умовний цикл, у якому перевіряється значення, що повертається функцією INKEY$.

У наведеному нижче прикладі програма буде перебувати в режимі очікування до моменту натискання довільної клавіші:

Приклад

```
PRINT "Press any key to continue"
WHILE INKEY$ = ""
WEND
```

Однак можна задати й конкретну клавішу, при натисканні на яку виконання програми буде продовжено.

Приклад

```
PRINT "Press ' ESC ' to continue"
WHILE INKEY$ <> CHR$(027)
WEND
```

8.4 Визначення довжини текстового виразу

Функція **LEN** призначена для визначення довжини символьного виразу.

LEN (*текст*),
де ***текст*** – текстовий вираз.

Функція обчислює довжину тексту, тобто число символів (включаючи проміжки).

Приклад

```
PRINT LEN("ВАСЯ")
```
результат: 4

8.5 Вибір підрядка (виділення частини тексту)

В мові QBasic існує кілька функцій, що дозволяють виділяти символи з текстового виразу. Ці функції мають такі формати:

LEFT$ (*текст, число*)
RIGHT$ (*текст, число*)
MID$ (*текст, позиція*[,*число*]),

де ***текст*** – текстовий вираз;
число – число виділюваних символів (від 0 до 32767);
позиція – номер символу, з якого починається виділення.

Функція **LEFT$** повертає підрядок з довжиною, рівною параметру ***число***, і, що починається з першого символу вхідного рядка (з лівого краю тексту).

Функція **RIGHT$** повертає зазначене ***число*** символів, починаючи із правого краю тексту, аналогічно функції **LEFT$.**

Функція **MID$** повертає підрядок довжиною ***число***, починаючи з ***позиції*** від лівого краю тексту. Для функції **MID$** параметр ***число*** не є обов'язковим. Якщо він не зазначений, то функція повертає правий залишок рядка від заданої ***позиції*** до кінця текстового виразу.

Якщо параметр ***число*** дорівнює 0, то всі функції повертають порожній рядок. Якщо значення даного параметра більше, ніж реальна довжина тексту, буде повернуто весь текстовий вираз.

Приклад	***Результат***
CLS EX$ = "Бажаю все знати" PRINT LEFT$ (EX$, 4) PRINT RIGHT$ (EX$, 9) PRINT MID$ (EX$, 6 , 3)	Бажаю все знати все

8.6 Пошук підрядка в символьному виразі

Припустимо, що маємо великий текстовий файл і бажаємо знайти місце розташування певної фрази в ньому. Програма істотно спроститься при використанні функції INSTR, що спеціально призначена для пошуку певного підрядка в символьному виразі.

Функція **INSTR** повертає числове значення, що вказує позицію, з якої починається шуканий підрядок, відносно початку тексту.

INSTR ([*позиція,*] *текст,* підрядок**),**
де ***позиція*** – номер символу, з якого починається пошук підрядка;
текст – текстовий вираз, у якому відбувається пошук;
підрядок –текстовий вираз, що шукається.

Параметр ***позиція*** необов'язковий, якщо його опустити, пошук буде виконуватися з першого символу текстового виразу.
Функція INSTR повертає 0, якщо:

- підрядок не знайдено;
- значення параметра ***позиція*** більше довжини тексту;
- довжина тексту нульова.

Якщо підрядок - порожній, функцією буде повернуте значення ***позиція*** (якщо параметр невизначений, то буде повернута 1). Пошук припиняється при першому ж знаходженні підрядка, тому друге входження підрядка знайдене не буде.

Приклад

```
CLS
EX$ = "АНЯ,ЯН,АН-ДРУЗІ"
PRINT EX$
PRINT "ІМ'Я ' ЯН ' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ"; INSTR (EX$, "ЯН")
PRINT "ІМ'Я 'АНЯ' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ";INSTR (EX$, "АНЯ")
PRINT "ІМ'Я 'АН' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ";INSTR (EX$,"АН")
```

Результат

```
АНЯ,ЯН,АН-ДРУЗІ
ІМ'Я ' ЯН ' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ 5
ІМ'Я 'АНЯ' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ 1
ІМ'Я 'АН' ПОЧИНАЄТЬСЯ З ПОЗИЦІЇ 8
```

8.7 Приведення тексту до різних варіантів написання

Якщо текст написаний маленькими літерами, а ви хочете вивести його на екран і роздрукувати, використовуючи тільки великі, вам не треба вводити текст знову - скористайтеся першою з таких функцій:

UCASE$ (*текст*)
LCASE$ (*текст*),

де ***текст*** – текстовий вираз, призначений для зміни регістра.

Функція LCASE$ перетворить всі літери рядка в маленькі, функція UCASE$ -у великі.

Приклад	***Результат***
BAL$="рахунок:" GAL$="Нуль" PRINT BAL$;GAL$ PRINT PRINT UCASE$ (BAL$); PRINT LCASE$ (GAL$)	рахунок: Нуль РАХУНОК: нуль

Ці функції корисні при перевірці відповідей на питання програми типу "Бажаєте повторити? Y/N ".

У тому випадку, коли невідомо, маленьку або велику літеру введе користувач, кожна із цих функцій замінить подвійну перевірку.

Приклад

```
BEGIN:
…....
INPUT "Бажаєте повторити? Y/N", IFLAG$
IF UCASE$(IFLAG$)= "Y" THEN BEGIN
```

8.8 Формування тексту й видалення початкових або кінцевих проміжків

Функція **SPACE$(*N*)** повертає рядок, що містить *N* проміжків.

Приклад	***Результат***
FOR I=1 TO 5 PRINT SPASE$(I);I NEXT I	1 2 3 4 5

Функція **STRING$(*N, символ*)** повертає рядок, що містить ***N*** одиночних символів ***символ***

(замість символу можна вказати його код від 0 до 255).

Приклад	***Результат***
PRINT STRING$(5,42);"QBASIC";STRING$(5,"/")	*****QBASIC/////

Функція **LTRIM$ (*текст*)** повертає рядок, отриманий з рядка *текст* видаленням з нього початкових (лівих) проміжків.

Функція **RTRIM$ (*текст*)** повертає рядок, отриманий з рядка *текст* видаленням з нього кінцевих (правих) проміжків.

Приклад	***Результат***
A$=" QBasic " B$=LTRIM$(A$) PRINT A$;"*" PRINT B$;"*" B$=RTRIM$(A$) PRINT B$;"*"	 QBasic * QBasic * QBasic*

8.9 Перетворення текстових значень у числові, й навпаки

Функція **VAL(*текст*)** повертає числове значення аргументу ***текст***, що має символьне значення арифметичного виразу.

Приклад	**Результат**
PRINT VAL ("3.14")	3.14

Функція **STR$(*число*)** повертає символьне подання числа або числового виразу.

Приклад	***Результат***
PRINT STR$ (123)	"123"

9 ГРАФІЧНІ МОЖЛИВОСТІ QBASIC

9.1 Графічні режими екрана

Існують два режими роботи з екраном - текстовий і графічний.

Текстовий режим дозволяє виводити 80 символів в одному рядку й містить на екрані 25 рядків (один - службовий). Такі параметри екрана встановлюються при вмиканні комп'ютера й зберігаються при роботі QBasic.

В QBasic існують спеціальні графічні оператори для створення зображень. Вони вимагають перемикання в графічний режим роботи екрана оператором **SCREEN**.

Графічні режими характеризуються кількістю точок по вертикальній і горизонтальній осях екрана.

SCREEN режим% [,[перемик_кольору] [,активн_стор][,видима _стор]]]

Вибір графічного режиму визначається параметром

- *режим%*, що супроводжує цей оператор.

0-текстовий; 1,7,13 – *320*200*; 8 – *640*200*; 9 – *640*350*; 11,12 – *640*480*

- *перемик_кольору* – 0/1 – встановлює монохромний або кольоровий режим;
- *активн_стор* – сторінка екрана, у яку записується виведення тексту або графіки;
- *видима_стор* - – сторінка екрана, відображувана на екрані в цей момент.

Приклад

SCREEN 2

9.1.1 Кодування графічних зображень. Принципи подання зображень

Комп'ютерні графіки – розділ інформатики, що вивчає роботу на комп'ютері із графічними зображеннями (рисунками, кресленнями, фотографіями, відеокадрами та ін.)

Найпоширенішими є два принципи подання зображень: векторний і растровий.

Растрова графіка – засоби й методи комп'ютерної графіки, що використовують растрове подання графічної інформації у вигляді сукупності кодів пікселів, складових зображення.

Піксель – найменший елемент зображення на екрані. Чим щільніше розташовані пікселі, тим краще виглядає зображення на екрані. Щільність пікселів виміряється як їх кількість на одиницю довжини; найпоширеніша одиниця – dpi – кількість точок на дюйм (як правило, 72 або 96 dpi).

Растр – прямокутна сітка пікселів на екрані.

Розв'язувальна здатність екрана – розмір сітки растра (**M** x **N**- добуток числа точок по горизонталі на число точок по вертикалі).

Відеоінформація – інформація про зображення, яка відтворюється на екрані комп'ютера і зберігається в пам'яті.

Відеопам'ять – оперативна пам'ять, що зберігає відеоінформацію під час її відтворення в зображенні на екрані.

Сторінка – частина відеопам'яті, що містить інформацію про один образ екрана (однієї "картинки" на екрані). У відеопам'яті можуть одночасно розміщатися кілька сторінок.

Графічний файл – файл, що зберігає інформацію про графічне зображення (.bmp, .jpeg)

Число кольорів, відтворених на екрані дисплея (**H**) можна підрахувати, знаючи число біт, що відводяться у відеопам'яті під кожний піксель (**n**)

$$H=2^n$$

де **n - бітова глибина** (довжина коду пікселя)

n=1- монохромний монітор, 2 кольор;

n=24 - кольоровий монітор, 16777216 кольорів

Приклад

Для "маленького монітора" з растровою сіткою 10 x10 і чорно-білим зображенням представимо літеру "К" у вигляді бітової матриці (рисунок 9.1).

	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	1	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	0	1	1	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	0	1	0	0	1	0	0	0
0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Рисунок 9.1

Все різноманіття фарб на екрані отримують шляхом змішування трьох базових кольорів: червоного, синього й зеленого.

Це змішування аналогічно змішуванню акварельних фарб на папері, але з однією відмінністю: колір акварельних фарб виходить у результаті відбиття падаючого на них світла, у той час, як колір на екрані формується в результаті випромінювання світла. Тому, коли ви змішуєте на папері три основні фарби, то одержуєте чорний колір, а при цих же кольорах максимальної яскравості на екрані виходить білий колір.

Кожний піксель на екрані складається із трьох близько розташованих елементів, що світяться цими кольорами. Кольорові дисплеї, що використовують такий принцип, називаються RGB - моніторами. Код кольору пікселя містить інформацію про частку кожного базового кольору.

Якщо всі три складові мають однакову інтенсивність (яскравість), то з їхніх сполучень можна одержати 2^3=8 різних кольорів. Таблиця 9.1 показує кодування 8-кольорової палітри за допомогою 3-розрядного двійкового коду. У ній наявність базового кольору позначена 1, а відсутність - 0.

Таблиця 9.1

Двійковий код 8-кольорової палітри			
к	**з**	**с**	**Колір**
0	0	0	Чорний
0	0	1	Синій
0	1	0	Зелений
0	1	1	Блакитний
1	0	0	Червоний
1	0	1	Рожевий
1	1	0	Коричневий
1	1	1	Білий

Таблиця 9.2 показує кодування 16-кольорової палітри за допомогою 4-розрядного двійкового коду..

Таблиця 9.2

Двійковий код 16-кольорової палітри					
И	**к**	**з**	**с**	**10-й код**	**Колір**
0	0	0	0	0	Чорний
0	0	0	1	1	Синій
0	0	1	0	2	Зелений
0	0	1	1	3	Блакитний
0	1	0	0	4	Червоний
0	1	0	1	5	Рожевий
0	1	1	0	6	Коричневий
0	1	1	1	7	Сірий
1	0	0	0	8	Темно-сірий
1	0	0	1	9	Яскраво-синій
1	0	1	0	10	Яскраво-зелений
1	0	1	1	11	Яскраво-блакитний
1	1	0	0	12	Яскраво-червоний
1	1	0	1	13	Яскраво-рожевий
1	1	1	0	14	Яскраво-жовтий
1	1	1	1	15	Білий

Перший стовпець- код інтенсивності, він управляє яскравістю всіх трьох кольорів одночасно

Ця таблиця відповідає номерам кольорів при використанні графіки в QBasic. Наприклад, жовтому відповідає шістнадцяткове число 1110H або 14 в десятковій системі.

Більша кількість кольорів виходить при роздільному управлінні інтенсивністю базових кольорів. Причому інтенсивність може мати більше 2 рівнів, якщо для кодування кожного з базових кольорів виділяти більше одного біта.

Приклад

При використанні бітової глибини 8 біт/піксель кількість кольорів 2^8=256. Біти такого коду розподілені в такий спосіб:

ККК333СС.

Червоний і зелений компоненти мають 2^3=8 рівнів яскравості, а синій – 4.

Відзначимо, що растрова графіка працює з реалістичними (фото) зображеннями.

Графічні растрові файли мають великий обсяг (потрібен стиск при зберіганні).

При зміні розмірів, обертанні й інших перетвореннях рисунка відбувається його перекручування.

Векторна графіка – засоби й методи комп'ютерної графіки, що використовують таке подання графічної інформації у вигляді графічних примітивів - сукупності простих елементів: прямих ліній, дуг, кіл, прямокутників, зафарбувань та ін. Положення й форма графічних примітивів задаються в системі графічних координат, пов'язаних з екраном. Звичайно початок координат розташований у верхньому лівому куті екрана.

Сітка пікселів збігається з координатною сіткою. Горизонтальна вісь X спрямована ліворуч праворуч, вертикальна вісь Y - зверху вниз.

Відрізок прямої лінії однозначно визначається вказівкою координат його кінців; коло - координатами центра й радіусом; багатокутник - координатами його кутів; зафарбована область - граничною лінією й кольором зафарбування та ін.

Приклад

Літера “К”, зображена растровими засобами (див. вище), у векторному поданні може бути подана трьома лініями. Якщо формат команди, що задає лінію, умовно має вигляд

ЛІНІЯ(X1,Y1,X2,Y2),

то наша літера описується в такий спосіб:

ЛІНІЯ(4,2,4,8)
ЛІНІЯ(5,5,8,2)
ЛІНІЯ(5,5,8,8)

Ці три оператори й подають векторний код літери “К”.

Графічні файли векторного типу мають невеликий обсяг. Векторні зображення легко масштабуються без втрати якості.

Варто пам'ятати:

- при введенні зображень у комп'ютер за допомогою сканера формуються графічні файли растрового типу;
- при виведенні на екран монітора будь-якого зображення у відеопам'яті формується інформація растрового типу;
- розходження в поданні графічної інформації в растровому й векторному форматах існує лише для графічних файлів.

Крім мов програмування, графіка використовується в графічних редакторах - прикладних програмах, призначених для створення й обробки графічних зображень на екрані.

Приклади векторних редакторів: CorelDraw, AdobeIllustrator.

Приклади растрових редакторів: Paint, CorelPHOTO-PAINT, AdobePhotoshop.

9.2 Графічні примітиви в мові QBASIC

Для зображення графічних примітивів в QBasic використовуються відповідні оператори:

- PSET, PRESET - рисування точки;
- LINE - рисування відрізка;
- CIRCLE - рисування кола.

Існують *три різних типи координат графічного екрана*.

- абсолютні координати;

- координати на основі Точки Останнього Посилання, у скороченні — ТОС (*Last Point Referenced- LPR);*
- відносні координати.

Абсолютні координати

З огляду на систему координат екрана, просто вказується місце, у якому треба нарисувати точку, наприклад, PSET(100,120).

Значення в дужках (100 і 120) вказують на положення точки, що рисується, по осях x і y відповідно.

Точка Останнього Посилання (ТОП)

Координати точки, що була нарисована останньою, зберігаються в пам'яті комп'ютера. Ця точка й називається *Точкою Останнього Посилання* й часто використовується в графічних операторах.

Приклад: при рисуванні лінії за допомогою оператора LINE - (300,120) досить указати координати тільки однієї точки, і на екрані буде проведений відрізок від ТОП до зазначеної точки, що після цього стане ТОП.

Відразу після включення графічного режиму ТОП є точка в центрі екрана.

Відносні координати

Ці координати показують величину переміщення щодо положення ТОП. Щоб нарисувати нову точку, використовуючи відносні координати, буде потрібно ключове слово STEP.

Приклад

PSET STEP (-5,8)

При цьому з'явиться точка, положення якої буде ліворуч на 5 і нижче на 8 точок відносно ТОП. Інакше кажучи, якщо ТОП має координати (100,100), то даний оператор означає рисування точки з координатами (95,108).

Абсолютні координати повинні бути завжди додатними, а відносні можуть бути як додатними, так і від’ємними.

9.2.1 Оператори PSET І PRESET

Оператор PSET призначений для рисування точки на екрані шляхом зміни її кольору з фонового (чорного) на білий.

Оператор може мати такі форми:

PSET ***(X, Y)*** — абсолютна форма;
PSET STEP ***(X, Y)*** - відносна форма;
PRESET ***(X,Y)*** — абсолютна форма;
PRESET STEP ***(X,Y)*** - відносна форма,

де ***X, Y*** - абсолютні координати або зсув точки відносно ТОП.

Оператор PRESET призначений для зміни кольору відповідної точки на фоновий, тобто виконує дію, зворотну PSET.

9.2.2 Прямі лінії, відрізки, прямокутники

Оператор LINE призначений для рисування відрізка, що з'єднує дві довільні точки екрана.

Загальна форма запису:

LINE [(X_початок, B_початок)] - (X_кінець, B_кінець) [,[колір][,[B/BF],[,стиль%]]],

де **X_початок, B_початок** — координати початку відрізка (необов'язкові параметри);
X_кінець, B_кінець — координати кінця відрізка (обов'язкові параметри);
колір – колір лінії;
B/BF – рисує прямокутник або зафарбований прямокутник замість лінії;
стиль% - чи будуть рисуватися точки растра.

Якщо координати початку відрізка не зазначені, то відрізок буде починатися в ТОП. В операторі LINE можна використати відносні координати початку й/або кінця відрізка.

Для рисування прямокутників можна вибрати більш простий шлях.

Приклад

LINE (50,50)-(150,155) , ,B

У випадку пропуску якого-небудь параметра або параметрів потрібно зберегти необхідну кількість розділовими комами.

Команда DRAW дозволяє:

- рисувати лінії;
- розфарбовувати області;
- обертати зображення;
- змінювати розміри (масштаб).

Загальний вид команди:

DRAW *текстова постійна*

текстова постійна - послідовність параметрів-кодів, які означають напрямок руху (команди). Після кожного параметра стоїть число, що вказує довжину лінії. Команда визначається однією або двома літерами, що задають конкретну дію.

Приклад: Ln - перемістити вліво P n, m - зафарбувати область

а) переміщення

DRAW "M 20,20" - рисує лінію від ТОП до точки з координатами (20,20)

(Аналог Line -(20,20)

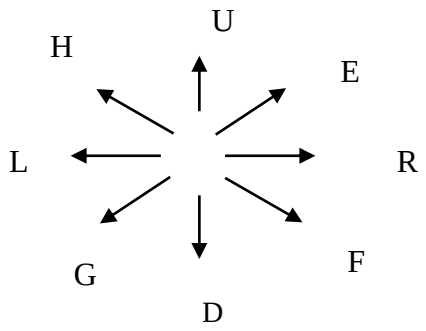
DRAW "M +25,-40" - рисує лінію від ТОП до точки на 25 точок праворуч й на 40 вище ТОП;

б) відносний рух

використовується для рисування ліній різних кольорів у вісьмох припустимих напрямках або переміщення в кожному із цих напрямків.

Кожна з команд супроводжується цілочисловим аргументом, що вказує довжину лінії в точках

DRAW "R16"



Корисними також є такі параметри:

- **B** - переміщення без рисування;
- **N** - переміщення без зміни ТОП;
- **C** - якщо він знаходиться в середині рядка символів разом із числовим значенням кольору, то колір ліній зміниться на зазначений.

Приклад

```
REM Рисування літери "B"
SCREEN 1
DRAW "BM+10, +0 R20 E10 U10 H5 E5 U10 H10 L20 D50"
DRAW "BM+10, -10 R10  U10 L10 D10"
    DRAW "BM+0, -20 R10  U10 L10 D10"
```

в) обертання

Команда An обертає зображення на кут, кратний 90 градусам, де n=0,1,2,3

Команда TAn дозволяє повертати зображення на довільний кут від -360 до +360 (- означає поворот за годинниковою стрілкою; + проти годинникової)

Приклад

```
REM Рисування літери "B" з поворотом на 90 градусів
DRAW "A1"
```

г) масштабування

Команда Sn змінює розмір зображення, збільшуючи або зменшуючи його залежно від значення n.

Зображення масштабується в n/4 разів, тому команда "S12" збільшує лінійний розмір зображення в 3 рази, а команда "S2" - зменшує вдвічі (n може мати значення від 1 до 225).

Приклад

```
REM Рисування літери "B", збільшеної вдвічі
DRAW "S8";
```

д) колір в операторі DRAW

Колір може бути визначений за допомогою команди Cn

```
REM Рисування літери "B" рожевого кольору
DRAW "C2"
```

DRAW P ***колір, контур***

зафарбовує замкнуту область.

9.2.3 Оператор CIRCLE

Дозволяє рисувати коло у будь-якому місці екрана:

CIRCLE ***(X_центр,B_центр), радіус*** **—** абсолютна форма;

CIRCLE STEP ***(X_центр,B_центр), радіус*** **—** відносна форма,

де ***X_центр, B_центр*** — координати або зсув центра кола;
радіус — радіус кола.

Приклад

```
REM Рисування кола
SCREEN  2
CLS
CIRCLE   (100,100),25
```

Коло із центром у точці з координатами (100,100) має радіус 25 точок.

9.2.4 Використання кольору

Команда COLOR

COLOR A [,B,C]

A - задає колір зображення, **B** - задає колір тіла, **C** - задає колір границі.

Колір, що використовується як параметр в операторах PSET, PRESET, LINE і CIRCLE, впливає тільки на зображення, залишаючи тіло без змін.

У графічних операторах QBasic колір визначається в такий спосіб:

PSET (X, Y) , колір;

PRESET (X,Y) , колір;

LINE (X_початок,B_початок) - (X_кінець,B_кінець) , колір;

CIRCLE (X_центр, B__центр), радіус, колір, де колір - значення колірного параметра.

У режимі екрана, що задається оператором SCREEN 2, можливі тільки два кольори чорний і білий, тому даний параметр марний.

Режим SCREEN 1 підтримує 4 кольори, яким відповідають значення від 0 до 3

Щоб “стерти” який-небудь елемент зображення без очищення всього екрана, можна просто перерисувати цей елемент кольором тіла.

Команда PAINT

використовується для розфарбовування замкнутих ділянок (не тільки прямокутних).

PAINT (X,Y),D,C

X,Y - координати будь-якої точки в середині замкнутої ділянки, що розфарбовують; **D** - колір розфарбовування; **C** - колір границі ділянки.

9.2.5 Дуга, еліпс і сектор

Щоб нарисувати дугу, еліпс або сектор кола, необхідно додати нові параметри в оператор **CIRCLE**, повна форма якого має такий вигляд:

CIRCLE (X,Y), радіус, колір, початок, кінець, коефіцієнт

- **X, Y** — координати центра кола;
- **радіус** — радіус кола;
- **колір** — його колір;
- **початок** — початкова точка дуги, задана в радіанах;
- **кінець** — кінцева точка дуги, задана в радіанах;
- **коефіцієнт** — відношення значень Y-радіуса й X - радіуса.

Для рисування кола використовуються тільки параметри X, Y і радіус.

Для рисування дуги необхідно додати значення параметрів початкової й кінцевої точок. Дуга визначається кутом, що вирізається з відповідного кола. Значення параметрів **початок** і **кінець** задаються в радіанах і повинні мати значення між 0 і 2π.

QBasic при рисуванні дуг веде відлік від початкової точки дуги до кінцевої проти годинникової стрілки.

Приклад

```
REM Рисування кола, дуги й сектора
SCREEN 2
CLS
CIRCLE (100,100), 30
CIRCLE (180,100), 30, 3, 1, 2
CIRCLE (260,100), 30, 3, -2, -1
```

Якщо одному з параметрів *(**початок або кінець**)* значення не присвоюється, то воно вважається рівним нулю.

При від’ємних значеннях цих параметрів QBasic з'єднує початкові кінцеві точки дуги із центром відповідного кола. Таким чином, на екрані виходить зображення сектора кола. Якщо від’ємним є значення тільки одного параметра, то й з'єднуватися із центром кола буде тільки одна точка дуги.

Для рисування еліпса потрібно задати **коефіцієнт** відношень радіусів по осях Y і X. Цей параметр визначає ступінь стиску еліпса й може мати будь-яке додатне значення.

Якщо параметр **коефіцієнт** опущений або дорівнює 1, ви одержуєте зображення кола. При від’ємному значенні параметра ви одержите повідомлення про помилку.

Приклад

```
REM Рисування еліпсів
SCREEN 2
CLS
CIRCLE (50,90), 30
CIRCLE (150,90),30, , , ,0.3
CIRCLE (250,90,30, , , ,1.5
```

9.3 Імітація руху на екрані

Для імітації руху зображення об'єкта на екрані необхідно виконати такий алгоритм:

- нарисувати об'єкт у заданій точці;
- знищити об'єкт, зафарбувавши його кольором тіла;
- змінити координати об'єкта;
- перейти до п.1.

Приклад

```
REM рух униз зеленого кола на чорному тілі
SCREEN 8
COLOR 2,1
Y=10: SY=4
50 CIRCLE (120,Y),40,2     'рисування кола
CIRCLE (120,Y),40,1        'стирання кола
Y=Y+SY                     ' зміна координати центра
IF Y<10 OR Y>190 THEN SY=-SY
GOTO 50
```

Варто організовувати штучні паузи між рисуванням і стиранням об'єктів для підтримки правильних зображень на екрані ("холості" цикли)

Оператори GET і PUT

З їхньою допомогою можна домогтися ефекту мультиплікації: створювати багаторазові копії зображення або рухати графічну картинку.

GET дає можливість зберегти будь-яку прямокутну область екрана в числовому масиві, а **PUT** відтворює зображення в довільному місці екрана.

Загальна форма запису

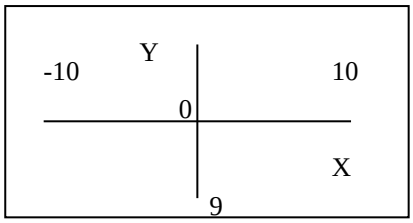
GET [STEP] (x1!,y1!) - [STEP] (x2!,y2!), *ім'я* [(*індекс%*)]
PUT[STEP] (x1!,y1!), *ім'я* [(*індекс%*)] [,*режим*]

- **STEP** - задає відносну форму графічних координат;
- **x1!,y1!** - координати верхньої лівої точки зображення, що зберігається оператором GET або точка на екрані, у якій оператор PUT поміщає це зображення;
- **x2!,y2!** - координати нижньої правої точки зображення, що зберігається;
- **ім'я_** - ім'я масиву, у якому зберігається зображення;
- **індекс%** - індекс елемента масиву, починаючи з якого зберігається зображення;

- **режим** - ключове слово, що визначає режим відтворення збереженого зображення. За замовчуванням - *XOR*;
- *AND* - об'єднання збереженого зображення з існуючого;
- *OR* - накладення збереженого зображення на існуюче;
- *PSET* - рисування збереженого зображення, знищуючи існуюче;
- *PRESET* - рисування збереженого зображення кольором тіла, знищуючи існуюче;
- *XOR*- рисування збереженого зображення або знищення попереднього, зі збереженням і відновленням тіла, відтворюючи ефект анімації.

Побудова графіків

Приклад



Знайдемо формули для переходу від математичних координат (X,Y) до графічного (XD,YD):

1 Різний початок координат:

XD=x+100:YD=Y+90;

2 Масштаб:1 до 10

XD=x*10+100:YD=Y*10+90;

3 Осі Y і YD протинаправлені

Будемо змінювати X від -10 до 10 з маленьким кроком, а також додамо побудову осей (LINE).

```
'Програма - графік
CLS
SCREEN 12
LINE (0, 90)-(200, 90)
LINE (110, 0)-(100, 180)
FOR x = -10 TO 10 STEP .01
y = x * x
xd = x * 10 + 100
yd = y * 10 + 90
IF yd > 0 AND yd < 180 THEN PSET (xd, yd)
NEXT x
```

СПИСОК ЛІТЕРАТУРИ

1 Інформатика, комп’ютерна техніка, комп’ютерні технології: Посібник/ За ред. О.І. Пушкаря - К.: Академія, 2001. – 696 с.

2 Смирнов Н.Н. Программные средства ПЭВМ. - Л.: Машиностроение ,1990.

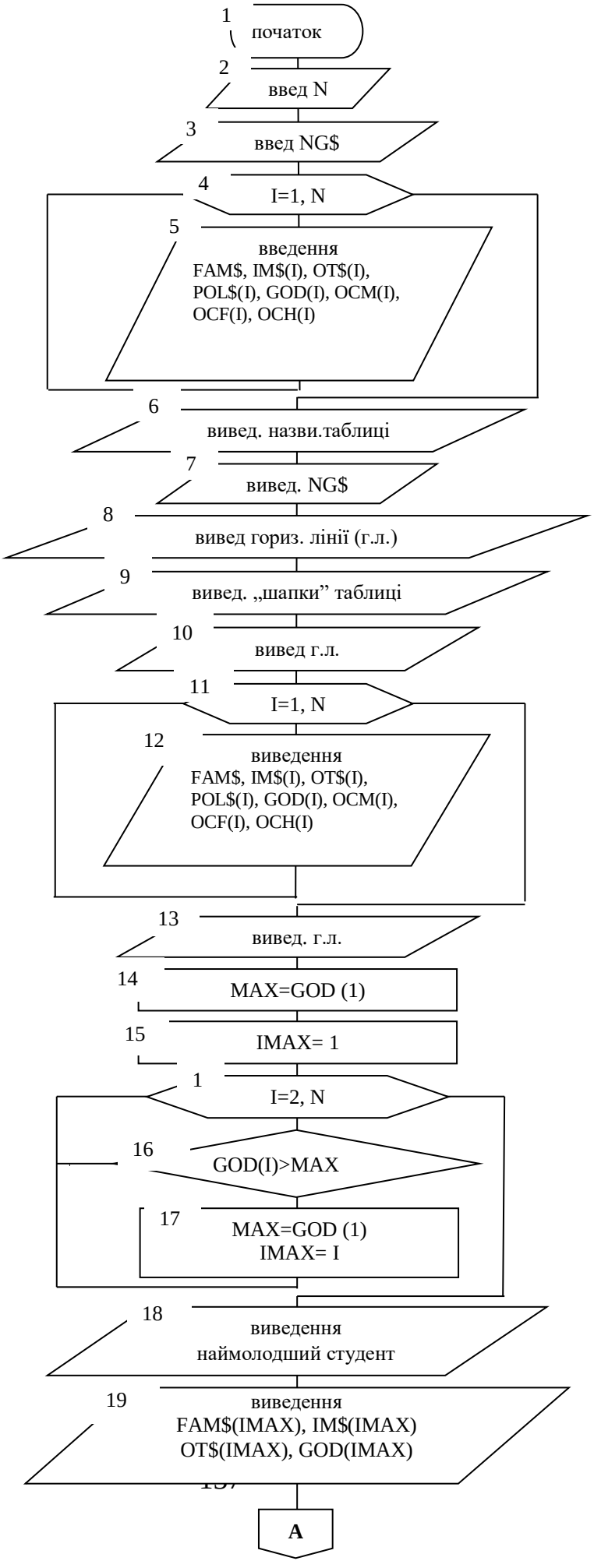
3 Філіппенко І.Г., Гончаров В.О., Меркулов В.С. Основи побудови ПК: Конспект лекцій з курсу "Обчислювальна техніка та програмування”. – Харків: УкрДАЗТ, 2005. –Ч. 1.

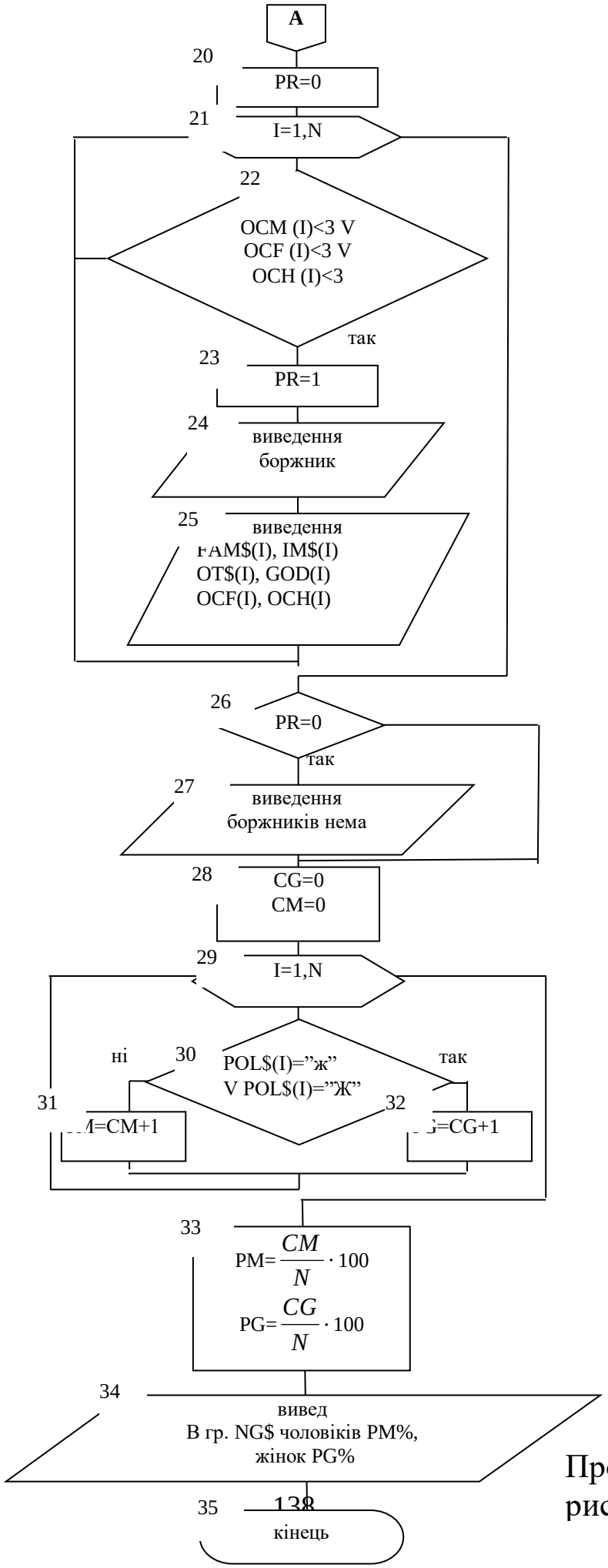
4 Філіппенко І.Г., Гончаров В.О., Меркулов В.С. Основи алгоритмізації: Конспект лекцій з курсу "Обчислювальна техніка та програмування”. – Харків: УкрДАЗТ, 2005. –Ч. 2.

5 Методичні вказівки до лабораторних робіт з дисциплін “Основи інформатики”, „Обчислювальна техніка та програмування” (основи проектування алгоритмів). – Харків: ХарДАЗТ, 2000. – Ч. 1.

6 Методичні вказівки до лабораторних робіт з дисциплін “Основи інформатики”, „Обчислювальна техніка та програмування” (програмування мовою BASIC). – Харків: ХарДАЗТ, 2000. –Ч. 2.

7 Методичні вказівки до лабораторних робіт з дисциплін “Основи інформатики”, „Обчислювальна техніка та програмування” (програмування мовою BASIC). – Харків: ХарДАЗТ, 2000. –Ч. 3.


1
початок
2
введ N
3
введ NG$
4
I=1, N
5
введення
FAM$, IM$(I), OT$(I),
POL$(I), GOD(I), OCM(I),
OCF(I), OCH(I)
6
вивед. назви.таблиці
7
вивед. NG$
8
вивед гориз. лінії (г.л.)
9
вивед. „шапки” таблиці
10
вивед г.л.
11
I=1, N
12
виведення
FAM$, IM$(I), OT$(I),
POL$(I), GOD(I), OCM(I),
OCF(I), OCH(I)
13
вивед. г.л.
14
MAX=GOD (1)
15
IMAX= 1
1
I=2, N
16
GOD(I)>MAX
17
MAX=GOD (1)
IMAX= I
18
виведення
наймолодший студент
19
виведення
FAM$(IMAX), IM$(IMAX)
OT$(IMAX), GOD(IMAX)
A



Продовження рисунку 5.5