ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра обчислювальної техніки і систем управління

ПРОГРАМУВАННЯ В СЕРЕДОВИЩІ VISUAL BASIC 6.0

КОНСПЕКТ ЛЕКЦІЙ

з дисциплін

«ОБЧИСЛЮВАЛЬНА ТЕХНІКА, ПРОГРАМУВАННЯ, МОДЕЛЮВАННЯ СИСТЕМ»
«ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ»

Частина 3

Харків – 2014

Програмування в середовищі Visual Basic 6.0: Конспект лекцій / В.С. Меркулов, І.Г. Бізюк, Н.М. Завгородня, О.В. Чаленко. – Харків: УкрДАЗТ, 2014. – Ч. 3. – 57 с.

Конспект лекцій розроблено у відповідності до програм з дисциплін «Обчислювальна техніка, програмування, моделювання систем» та «Обчислювальна техніка та програмування».

Метою лекцій є опанування студентами алгоритмічної мови високого рівня, здобуття необхідних знань для організації обчислень в середовищі Visual Basic 6.0, використанні теоретичних засад об'єктно-орієнтованого підходу до розробки програмних проектів.

Рекомендується для студентів спеціальностей 6.100501 та 6.092200 всіх форм навчання.

Іл. 13, табл. 14, бібліогр. 25 назв.

Конспект лекцій розглянуто та рекомендовано до друку на засіданні кафедри обчислювальної техніки та систем управління 26 лютого 2013 р., протокол № 8.

Рецензент

проф. В.І. Мойсеєнко

ПРОГРАМУВАННЯ В СЕРЕДОВИЩІ VISUAL BASIC 6.0

КОНСПЕКТ ЛЕКЦІЙ

з дисциплін

«ОБЧИСЛЮВАЛЬНА ТЕХНІКА, ПРОГРАМУВАННЯ, МОДЕЛЮВАННЯ СИСТЕМ»
«ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ»

Частина 3

Відповідальний за випуск Бізюк І.Г.

Редактор Еткало О.О.

Підписано до друку 25.03.13 р.

Формат паперу 60x84 1/16. Папір писальний.

Умовн.-друк.арк. 2,00. Тираж 100. Замовлення №

Видавець та виготовлювач Українська державна академія залізничного транспорту, 61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 2874 від 12.06.2007 р.

Програмування в середовищі Visual Basic 6.0: Конспект лекцій / В.С. Меркулов, І.Г. Бізюк, Н.М. Завгородня, О.В. Чаленко. – Харків: УкрДАЗТ, 2014. – Ч. 3. – 57 с.

Конспект лекцій розроблено у відповідності до програм з дисциплін «Обчислювальна техніка, програмування, моделювання систем» та «Обчислювальна техніка та програмування».

Метою лекцій є опанування студентами алгоритмічної мови високого рівня, здобуття необхідних знань для організації обчислень в середовищі Visual Basic 6.0, використанні теоретичних засад об’єктно-орієнтованого підходу до розробки програмних проектів.

Рекомендується для студентів спеціальностей 6.100501 та 6.092200 всіх форм навчання.

Іл. 13, табл. 14, бібліогр. 25 назв.

Конспект лекцій розглянуто та рекомендовано до друку на засіданні кафедри обчислювальної техніки та систем управління 26 лютого 2013 р., протокол № 8.

Рецензент
проф. В.І. Мойсеєнко

ЗМІСТ

8 ОСНОВНІ ЕЛЕМЕНТИ ПРОГРАМУВАННЯ VB 6.0. 4

8.1 Елементарні конструкції 4

8.2 Методи введення-виведення. Діалогові вікна 37

8.3 Оператори управління 42

БІБЛІОГРАФІЧНИЙ СПИСОК 56

8 ОСНОВНІ ЕЛЕМЕНТИ ПРОГРАМУВАННЯ VB 6.0

8.1 Елементарні конструкції

8.1.1 Склад мови

У тексті будь-якою природною мовою можна виділити чотири основних елементи: символи, слова, словосполучення та речення.

Подібні елементи містить і алгоритмічна мова, тільки слова називають ***лексемами*** (елементарними конструкціями), словосполучення — ***виразими***, а речення — ***операторами***.

Лексеми утворюються із символів, вирази — з лексем і символів, а оператори — із символів, виразів і лексем (рисунок 8.1).

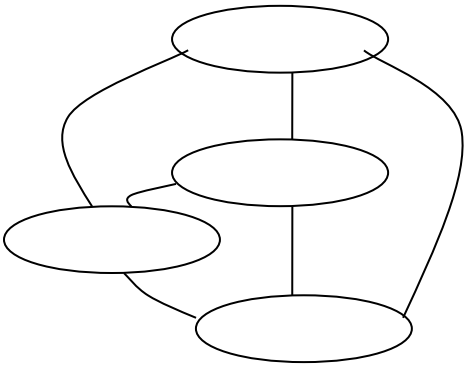


Рисунок 8.1 — Склад алгоритмічної мови

8.1.2 Алфавіт і лексеми мови

Основою будь-якої мови є *алфавіт. Алфавіт мови*, або її символи — це основні неподільні знаки, за допомогою яких пишуться всі тексти цією мовою.

Алфавіт мови програмування — набір припустимих знаків, які можна використати для запису програм.

Алфавіт VB6.0 включає:

- ***великі і малі латинські букви, букви кирилиці;***
- ***арабські цифри від 0 до 9;***
- ***розділові символи (обмежники):***

.	**десяткова крапка**	**,**	**кома**
:	**двокрапка**	**« “**	**лапки**

( ) круглі дужки	**‘ одинарні лапки (апостроф)**
; крапка з комою	

- ***узагальнені пробільні символи:***

символ табуляції	**символ переходу на новий рядок**
пропуск	

- ***спеціальні знаки:***

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

- ***знаки арифметичних операцій:***

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

- ***знаки операцій відношення:***

***основні*:**	***похідні* (складені символи, сприймані як один символ):**
> знак більше ніж	**>= знак більше або дорівнює**
< знак менше ніж	**<= знак менше або дорівнює**
= знак дорівнює	**<> знак не дорівнює**

Кожному символу відповідає індивідуальний числовий ASCII-код у діапазоні від 0 до 255 (0-127 — основна таблиця, 128-255 — розширена).

Лексема — мінімальна одиниця мови, що формується із символів алфавіту та має самостійний вміст.

Лексеми мови мають певний сенс для компілятора і не можуть бути розбиті надалі.

Границі лексем визначаються іншими лексемами, такими, як роздільники або знаки операцій. В VB 6.0 розрізняють ***6 класів лексем***:

- вільно обирані й використовувані ідентифікатори;
- службові (*зарезервовані*) слова;

- константи;
- рядки (*рядкові константи*);
- знаки операцій;
- роздільники (*знаки пунктуації: дужки, крапка, кома, пробільні символи*).

Сукупність лексем і правил їх формування утворюють ***лексику мови.***

Вираз задає правило обчислення деякого значення.

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.

Оператори бувають такі, ***що виконуються й не виконуються***:

- *оператори, що виконуються,* задають дії над даними;
- *оператори, що не виконуються*, служать для опису даних, тому їх часто називають *операторами опису*.

Групи операторів ***за призначенням*** наведені в таблиці 8.1.

Таблиця 8.1

Група	***Призначення***
Описові оператори	Для опису даних, типів змінних, розмірів масивів, нестандартних функцій і т.ін.
Оператори присвоювання	Для задавання початкового значення або зміни поточного значення змінної
Оператори введення-виведення даних	Для введення вихідних даних та виведення результатів
Оператори управління процесом обробки інформації	Для організації галужень, циклів і т.ін.
Інші оператори, що забезпечують додаткові можливості	Для роботи з файлами даних, для графічних побудов, для одержання звукових ефектів і т.ін.

Кожний елемент мови визначається ***синтаксисом*** і ***семантикою***:

- *синтаксичні визначення* встановлюють правила побудови елементів мови;
- *семантика* визначає їх вміст і правила використання.

Програмний код VB 6.0 являє собою послідовність лексичних одиниць (лексем), записаних відповідно до прийнятих синтаксичних правил, що реалізує деяку семантичну конструкцію.

Об'єднана єдиним алгоритмом сукупність описів і операторів утворює ***програму алгоритмічною мовою.*** Для того, щоб виконати програму, потрібно перекласти її на мову, зрозумілу процесору, — **машинні коди**. Цей процес складається з декількох **етапів:**

- спочатку програма передається препроцесору, що виконує директиви, які містяться в її тексті (наприклад, включення в текст так званих заголовних файлів — текстових файлів з описами використовуваних у програмі елементів);
- отриманий повний текст програми надходить на вхід компілятора, що виділяє лексеми, а потім на основі граматики мови розпізнає вирази й оператори, побудовані із цих лексем. При цьому компілятор виявляє синтаксичні помилки й у випадку їхньої відсутності будує об'єктний модуль;
- компоновник, або редактор зв'язків, формує модуль програми, що виконується. Він підключає до об'єктного модуля інші об'єктні модулі, у тому числі такі, що містять функції бібліотек, звертання до яких є в будь-якій програмі (наприклад, для здійснення виведення на екран). Якщо програма складається з декількох вихідних файлів, вони компілюються окремо й поєднуються на етапі компонування;
- модуль, що виконується, має розширення ***.exe*** і запускається на виконання звичайним способом.

8.1.3 Дані

Концепція типу даних

Дані характеризуються ***типом*** і ***організацією***. *Тип даних* визначає:

- внутрішнє подання даних у пам'яті комп'ютера;
- безліч значень, які можуть набувати величини цього типу;
- операції й функції, які можна застосовувати до величин цього типу.

Тип даних узагальнює такі поняття, як:

- розміщення в ЕОМ;
- спосіб подання;
- прикладний зміст.

Розміщення. Дані й програма в момент виконання розміщаються в оперативній пам'яті ЕОМ, що складається із пронумерованих комірок, кожна з яких уміщає 1 байт даних. Номер певної комірки називається ***адресою***. Одного байта для зберігання числового даного, як правило, недостатньо, і дане займає безперервну послідовність байтів (2, 4, 8 байтів).

Спосіб подання. Послідовність бітів у байтах даного кодує всю необхідну інформацію, що визначає дане. Правила кодування і їхня реалізація в різних ЕОМ визначають спосіб подання даного. Так, для числових даних існує спосіб подання з фіксованою крапкою і з плаваючою крапкою.

Для кодування символів досить одного байта, щоб закодувати 256 символів (з десятковими кодами від 0 до 255).

Прикладний зміст. Визначає можливість використання даних для тих або інших цілей (з арифметичними даними можна робити обчислення, із символів можна будувати слова й речення).

Під ***організацією даних*** розуміється незалежність окремих даних (зберігаються в окремих непослідовних комірках пам'яті) або їхня зв'язаність (зберігаються у зв'язаній послідовності комірок пам'яті).

Зв'язаними даними в VB 6.0 є ***масиви*** (сукупність зв'язаних даних одного типу) і ***записи*** (сукупність зв'язаних даних різних типів).

Внутрішнє подання значень

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

- **з *фіксованою крапкою;***
- **з *плаваючою крапкою.***

У форматі *з фіксованою крапкою* предаються тільки цілі числа, а у *форматі із плаваючою крапкою* — речовинні числа (цілі і дробові). *Під крапкою тут мається на увазі знак-роздільник цілої й дробової частини числа.*

***Цілі** числа*

У *форматі з фіксованою крапкою* 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 $-2^{k-1} = -2^{15}(-32768)$ до $2^{k-1} - 1 = 2^{15} - 1$ (32767);
- в 32-розрядній — від -2147483648 до 2147483647.

*Вихід результатів обчислень за межі припустимого діапазону називається **переповненням**. Переповнення при обчисленнях з фіксованою крапкою не викликає переривання роботи процесора. Обчислення продовжуються, але результати можуть виявитися неправильними.*

Старший байт								**Молодший байт**							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
знак	14 біт	13 біт	12 біт	11 біт	10 біт	9 біт	8 біт	7 біт	6 біт	5 біт	4 біт	3 біт	2 біт	1 біт	0 біт

Такий вигляд має в пам'яті комп'ютера число +1. *Знак числа зберігається в старшому біті (крайньому ліворуч):*

0 *відповідає знаку* **+**

1 *відповідає знаку* **-**

Додатні числа зберігаються в ***прямому коді.***

Від'ємні числа зберігаються в ***додатковому коді.***

***Речовинні** числа*

Числові величини, які можуть набувати будь-які значення (цілі й дробові), називаються речовинними числами. *В математиці також використовується термін «дійсні числа».*

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

$$R=m*n^p$$

Приклад: m = 0.3, **n** = 10, а **p** візьмемо різним:
$0.3 \bullet 10^{-1}=0.03$ $0.3 \bullet 10^{-2}=0.003$ $0.3 \bullet 10^{2}=30$ $0.3 \bullet 10^{3}=300$

З наведеного прикладу видно, що завдяки зміненню порядку крапка переміщується (плаває) по мантисі. При цьому, якщо порядок від'ємний, крапка зміщується по мантисі ліворуч, а якщо додатний, то праворуч.

Число із плаваючою крапкою розбивається на три частини (набори окремих розрядів), умовно розділених на:

- знак числа;
- порядок (позиція коми);
- мантиса (самі цифри).

Мантиса містить разом і дробову й цілу частини. *Порядок* і *мантиса* — цілі числа, які разом зі знаком дають подання числа із плаваючою крапкою в такому вигляді:

Приклад: число 123.45 зберігається в мантисі як 12345, а де перебуває крапка, визначається порядком**.**

Класифікація даних

В VB 6.0 розрізняють дві групи типів даних: **основні** (іноді їх називають базові або вбудовані) і типи даних, **обумовлені користувачем** (користувальницький або власний тип).

Застосовуються такі **базові типи даних**:

- *числовий;*
- *рядковий (**String**);*
- *типу дата (**Date**);*
- *байтовий (**Byte**);*
- *логічний (**Boolean**);*
- *довільний (**Variant**);*
- *об'єктний (**Object**).*

Для зберігання чисел використовуються такі **типи**:

- *ціле (**Integer**);*
- *довге ціле (**Long**);*
- *десяткове звичайної точності (**Single**);*
- *десяткове подвійної точності (**Double**);*
- *десяткове довге (**Currency**);*
- *однобайтне ціле (**Byte**).*

Для ефективного використання пам'яті необхідно правильно вибирати тип даного. У таблиці 8.2 наведені базові типи даних VB 6.0, необхідна для їхнього розміщення пам'ять, діапазон можливих значень, префікси та суфікси, використовувані в позначеннях, та приклади.

Таблиця 8.2

Тип даних	***О'бєм занятої пам'яті, байт***	***Діапазон значень***	***Префікс***	***Суфікс***	***Приклад***
1	**2**	**3**	**4**	**5**	**6**
Цілі числа					
Byte	**1**	**Додатне число**	**byt**		**bytImage**

(однобайтне ціле число)		**від 0 до 255**			
Integer (коротке ціле число)	**2**	**Від -32768 до 32767**	**int**	**%**	**intQuantity**
Long (довге ціле число)	**4**	**Від -2147483648 до 2 147483648**	**lng**	**&**	**ingTotal**
Boolean (логічне значення)	**2**	**True (ІСТИНА) або False (ХИБНІСТЬ)**	**bin**		**binSuccess**
Числа з плаваючою крапкою					
Single (речовинне число із плаваючою крапкою (нормальне, одинарне))	**4**	**Від'ємні числа: від -3.4E+38 до -1.4E-45 Додатні числа: від 1.4E-45 до 3.4E+38**	**sng**	**!**	**sngLength**
Double (речовинне число із плаваючої комою подвійної точності (довге))	**8**	**Від'ємні числа: від -1.8D+308 до -4.9D-324 Додатні числа: від 4.9D-324 до 1.8D+308**	**dbl**	**#**	**dblSum**

Продовження таблиці 8.2

1	2	3	4	5	6
String (рядок змінної довжини)	**10 байт + довжина рядка (1 байт на кожний символ)**	**Від 0 до 2 мільярдів символів**	**str**	**$**	**strLastname**
String *довжина	**довжина рядка**	**Від 1 до ~65400**		**$**	

(рядок фіксованої (постійної) довжини))	**(1 байт на кожний символ**				
Об’єктні типи					
Object (посилання на об'єкт)	**4**				
Невизначені типи					
Variant (числові типи)	**16**	**Довільне числове значення**	**vnt**		**vntValue**
Variant (символьні типи)	**22 байта+ довжина рядка**	**Довільне символьне значення**	**vnt**		
Інші типи					
Currency (грошова величина— число з фіксованою крапкою)	**8**	**Ціла частина числа до 15 цифр, дробова — до 4 Від -922337203685477. 5808 до 922337203685477.5807**	**cur**	**@**	**curPrice**
Date (дата/час)	**8**	**Діапазон дат від 1 січня 100 року до 31 грудня 9999 року. Діапазон часу від 00:00:00 до 23:59:59.**	**dtm**		**dtmFinish**

Byte

Тип Byte — це найменший із трьох цілих типів даних VB 6.0, призначений для збереження цілих чисел від 0 до 255. Зберігати від'ємні числа в типі Byte не можна. Його використання дає змогу значно заощаджувати оперативну пам'ять. Звичайно типи Byte використовуються для збереження двійкових даних (графічних, звукових файлів і т. д.).

Boolean

Логічний або булевий тип називають також типом Boolean — тип даних може зберігати тільки два значення: TRUE або FALSE (істина або хибність) — булеві значення. Його використання в операціях порівняння замість цілочислових є гарним стилем програмування. Якщо відображається тип Boolean на екрані, VB 6.0 автоматично перетворить його в рядок, що містить слово TRUE або FALSE.

Variant

Тип даних **Variant** — це особливий тип даних, що може зберігати будь-які типи даних і виконувати операції, не піклуючись про тип даних, що міститься в них, за винятком типу **Object**. VB 6.0 використовує тип **Variant** для всіх змінних, для яких тип не визначений заздалегідь. Змінна типу **Variant** підтримує внутрішнє подання даних, що зберігаються в ній. VB 6.0 використовує для даних цього типу найбільш компактне продання, можливе для конкретних значень, що містяться в даних. Типи **Variant** потребують більшого обсягу пам'яті, чим будь-який інший тип даних, за винятком великих рядків. Дані типу **Variant** набувають характеристик визначеного типу, що зберігаються в даний момент.

Приклад: якщо дані типу **Variant** містять символьні дані, вони набувають характеристик типу **String** (байт/символ). Якщо дані типу

Variant містять числові дані — характеристики якого — небудь числового типу, звичайно — **double.**

Currency

Тип **Currency** — це число з фіксованою крапкою, тобто десяткова крапка завжди міститься в тому самому місці — праворуч від неї завжди містяться чотири цифри. Він створений для того, щоб уникнути помилок при перетворенні десяткових чисел у двійкову форму й навпаки. Використовується тип **Currency** для збереження чисел, коли точність вкрай важлива, що буває при «грошових» обчисленнях. Математичні операції над числами типу **Currency** мають невеликі або зовсім не мають помилок округлення, тому вони точніші, ніж операції над числами з плаваючою крапкою.

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

```
TYPE grupa
    name1 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
```

Деякі угоди щодо створення ідентифікаторів

Існує **угорська нотація** (*оскільки запропонував її співробітник компанії Microsoft, угорець*), за якою кожне слово, що становить ідентифікатор, починається із великої букви, а спочатку ставиться префікс, що відповідає типу величини та області її дії.

Приклад: dblMaxLength

Інша традиція — розділяти слова, що становлять ім'я, знаками підкреслення.

Приклад: max_length

Імена із великих літер використовуються для визначення констант.

Приклад: conPi, conVersion

Гарним тоном при програмуванні будь-якою мовою є осмислений вибір імен для об'єктів програми (присвоювати об'єктам імена, що відповідають контексту й несуть описове навантаження). Це правило сприяє розумінню програми.

Як **приклад**, *можна навести ім'я процедури обробки події, пов'язаної із клацанням миші по командній кнопці, що запускає програму* ***btnStart_ Click****: перша частина імені складається зі скорочення слова кнопка (****button — btn****) і слова* ***Start****, друга частина визначає подію —* ***Click****.*

Декларація змінних може бути **явною** або **неявно**ю**.**

Явне визначення змінних *має такі переваги:*

▪ прискорюється виконання коду. VB 6.0 створює всі оголошені явно змінні в модулі або процедурі перед виконанням коду процедури;

▪ зменшується кількість помилок у результаті правильного написання імені змінної;

▪ код стає більш читабельним і зрозумілим. За всіма оголошеннями змінних на початку модуля або процедури користувач може легко визначити, які змінні використовуються в цьому модулі або процедурі;

▪ за допомогою явного оголошення змінних можна нормалізувати виділення великих букв в імені змінної. Якщо явно повідомляється про змінну, VB 6.0 завжди змінює великі букви в імені змінної в операторі відповідно до імені в оголошенні змінної.

Для явного визначення змінних існують два способи:

1) *використання* ***операторів оголошення****:*

Формат:

Static / Public / Private / Dim *Ім'я_змінної* [As *Тип_змінної*] [, *Ім'я_змінної* [As *Тип_змінної*]]….

де *Dim (Розмір)* — ключове слово, яке повідомляє VB 6.0, що декларується змінна й резервується область пам'яті для її зберігання;

Ім'я_змінної — ім'я змінної (ідентифікатор, що не входить у перелік ключових слів VB 6.0);

As (як) — ключове слово, яке повідомляє VB 6.0, що визначається тип даних для змінної;

Тип_ змінної — тип даних для повідомлення змінної;

Private (Приватний), Public (Загальний) — ключові слова, що визначають область видимості змінної;

Оголошення типу цими операторами записується на початку тексту програмного коду.

Приклад:

Private binSuccess As Boolean

Dim strLastname As String, dblSum As Double

При підготовці коду програми середовище програмування надає допомогу користувачеві: після набору ключового слова As розкривається список, у якому поряд з іншими типами об'єктів зазначені й базові типи змінних. Тип змінної можна встановити, двічі клацнувши на типі в цьому списку або натиснувши клавішу [*Tab*].

2) *використання* ***спеціального символу-суфіксу***

Тип змінної визначається за допомогою додавання в кінець її імені спеціального символу опису типу — суфікса: %, &, !, #, $, @.

Ключове слово As не потрібне.

Формат:

Static / Public / Private / Dim Ім'я_змінної [суфікс] [, Ім'я_змінної [суфікс]]….

Приклад:

Dim strInputMsg$ — *оголошується змінна типу "рядок" (String);*

Static sngCalcAverage! — *оголошується змінна типу Single;*

Private intNumVal% — *оголошується змінна типу "ціле" (Integer)*

Неявне визначення змінних здійснюється двома способами:

1) із використанням ***оператора DEF***

DEF тип_даних діапазон_символів [, діапазон_символів] ….

де *тип_даних* — скорочена назва типу даних:

INT — 2-байтовий цілочисловий;

LNG — 4-байтовий цілочисловий;

SNG — 4-байтовий речовинний;

DBL — 8-байтовий речовинний;

STR — символьний.

діапазон_символів — указує границі діапазону імен, для яких задається тип даних за замовчуванням.

Цей оператор встановлює тип даних для змінних, параметрів процедур і тип значення, що повертається, для процедур, імена яких починаються з певних символів, тобто *за* ***приналежністю першого символу ідентифікатора заданому діапазону символів*.**

Приклад:

Defstr L-Z 'змінні, імена яких починаються із літер L-Z, будуть мати символьний тип.

Defint A, P-S, Z 'змінні, імена яких починаються з A,P,Q,R,S,Z вважаються 2-байтовими цілого типу.

2) за замовчанням

Змінна декларується просто вказівкою її імені у тексті програми.

Приклад: MyVal

На змінні, описані явним способом, неявне оголошення типів змінних не поширюється. У цьому випадку тип змінної

визначається при першому операторі присвоювання, і в цей же момент змінній буде виділена пам'ять. Якщо не використовуються спеціальні кінцеві символи, що визначають тип змінної, їй буде присвоєний тип *variant*.

Масиви даних.

Розрізняють ***статичні*** та ***динамічні*** масиви.

Границі *статичного* масиву встановлюються під час розробки проекту і можуть змінюватися тільки в новій версії програми.

Динамічні масиви змінюють свої границі під час виконання програми. З їх допомогою можна динамічно задавати розмір масиву відповідно до конкретних умов. *Треба пам'ятати, що робота з динамічними масивами потребує додаткових витрат на програмування.*

Статичні масиви

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

Декларація масиву

Формат:

[Public/ Private] Dim ім'я_масиву(індекси) **As** тип_даних

де ***Dim*** — ключове слово, яке вказує, що оголошується змінна*;*

ім'я_масиву — ідентифікатор, що визначає ім'я масиву;

As тип_даних — будь-який тип даних VB 6.0 — базовий або створений користувачем

індекси — список вимірностей у вигляді:

[низ TO] верх[,[низ TO] верх]...;

низ — нижня границя індексу масиву. За замовчуванням нижня границя дорівнює нулю;

верх — верхня границя. При декларації багатовимірного масиву вказується кілька індексів, відповідно до вимірності масиву.

Максимальне значення індексу **2147483647** (тип **long**), мінімальне (за замовчуванням) — 0. При необхідності за допомогою оператора **Option Base n** (**n=0/1**) воно може бути встановлено рівним 1.

Зауваження: *надалі у прикладах вважатимемо, що* ***n*** *= 1*

Застосувавши оператор **DIM,** ви одночасно:

- визначаєте ім'я масиву;
- описуєте тип елементів масиву або вказуєте тип даних для скалярних змінних;
- обнуляєте всі елементи числових масивів, а символьним — присвоюєте значення порожнього рядка (" ");
- резервуєте комірки пам'яті для елементів масивів.

Багатовимірні масиви. Під час декларації багатовимірного масиву верхні границі кожного виміру розподіляються комами.

Приклад:

```
Dim aName (10,25) as String
Private Sub Coommand1_Click()
 aName(1,3)=”X”
```

End Sub

*Масив з ім'ям aName може містити 250 різних значень (10*25=250).*

Динамічні масиви (змінного розміру)

Іноді при декларації масиву його розмір невідомий. У цьому випадку доцільно оголосити динамічний масив, що дає змогу змінювати його розмір або вимір під час виконання додатка.

Застосування динамічних масивів дає змогу ефективно управляти пам'яттю, виділяючи пам'ять під великий масив лише на той час, коли цей масив використовується, а потім звільняючи її.

***Створюється динамічний масив** у 2 етапи:*

1-й *етап*. Оголошенням за допомогою операторів оголошення, використовуваних при створенні масиву фіксованого розміру. Список вимірностей масиву залишається порожнім.

[Public/ Private] Dim ім'я_масиву()[**As** тип_даних]

2 *етап*. За допомогою виконуваного оператора **ReDim** у процесі виконання програми вказується фактична кількість елементів у динамічному масиві і його вимір.

Синтаксична конструкція перевизначення масиву:

ReDim [Preserve] ім'я_масиву **(**індекси**) [As** тип_даних**]**

де ***ReDim**—ключове слово, яке вказує, що перевизначаються розміри масиву;*

***індекси** — вимірності масиву (до 60);*

***Preserve** — ключове слово, що використовується для збереження даних в існуючому масиві при зміні значення його розміру.*

На відміну від масивів статичних розмірів, коли звертатися до елементів можна відразу після його оголошення, до елементів динамічного масиву відразу звертатися не можна, тому що вони ще не ініціалізовані. При зміні виміру виникає небезпека втратити вміст масиву (дані, розміщені в масиві раніше, втрачаються), бо після зміни виміру елементам масиву присвоюються значення за замовчуванням. Це зручно в тому

випадку, якщо дані вам більше не потрібні і ви хочете перевизначити розмір масиву та підготувати його для розміщення нових даних.

Присвоювання масивів

Немає необхідності створювати цикл ***For...Next*** для присвоювання одного масиву іншому за кожним елементом.

Досить написати оператор

NewMassive=OldMassive

і вміст масиву *OldMassive* присвоїться масиву *NewMassive*. Для виключення помилок при такому присвоєнні бажано дотримуватися однакової вимірності і типу масивів.

Область дії змінних

Область дії визначає можливість доступу до даних в окремих процедурах однієї форми або в процедурах, що належать до різних форм однієї програми. ***Область дії*** або ***область видимості*** змінної — це область, де використовується змінна.

Ви можете *оголосити змінну* для роботи:

- *в межах однієї процедури;*
- *в будь-якій процедурі даної форми;*
- *для роботи у всій програмі.*

Коли Ви оголошуєте змінну, зона її видимості задається одним з ключових слів: ***Dim, Private, Public***. Проте зона видимості змінної залежить і від того, де вона оголошена.

Dim. У такий спосіб оголошують *локальні* змінні, які існують тільки під час виклику процедури, де вони оголошені. Але якщо змінна за допомогою *Dim* оголошена в розділі глобальних оголошень форми або модуля, то вона буде доступна у всіх процедурах цієї форми або модуля, але для інших форм і модулів така змінна буде «невидимою».

Private. Відрізняється від *Dim* тим, що не може оголошувати змінні всередині процедури або функції. При оголошенні ж у розділі глобальних оголошень форми або модуля *Dim* і *Private* рівнозначні.

Public. Якщо змінна оголошена з використанням цього ключового слова, то вона є *глобальною* й доступна із всіх форм і модулів проекту. Якщо змінна оголошена як *Public* у коді форми, то з інших форм і модулів доступ до неї повинен здійснюватися через таку конструкцію: *Ім'я форми.Ім'я змінної.*

Якщо змінна оголошена як *Public* у розділі оголошень програмного модуля, то доступ до неї можливий просто через її ім'я.

Static. Змінні *Static* оголошуються всередині процедур і функцій і поза ними недоступні, але після завершення роботи процедури або функції, у якій оголошені, зберігають своє значення. *За замовчуванням оголошена змінна є локальною, діючою тільки в межах тієї процедури, у якій вона була створена.*

Модулі коду

Форма й весь пов'язаний з нею код зберігаються в окремому файлі з розширенням ***frm.*** Крім того, ви можете розробляти програми, що містять тільки код. Він розміщується у файлах модулів коду з розширенням ***bas.***

Формат запису імен змінних з урахуванням їх області дії

Таблиця 8.3

Область дії змінної	Префікс	Приклад
Глобальна	**g**	**gdtmFinish**
Локальна всередині модуля	**m**	**msngLength**
Локальна всередині процедури	**Немає префікса**	**strLastnam**

Оператори, вирази й операції

Програмний оператор являє собою будь-яку комбінацію ключових слів, властивостей, функцій, операцій і символів, сукупність яких являє собою коректну конструкцію, розпізнавану VB 6.0.

Приклади:

Beep

Label1.Caption = Time

Призначення будь-якого виразу — одержання деякого значення.

Приклади:

(3.14159 * D^2)/4 Pi = 3.14159

Якщо значеннями виразу є цілі й речовинні числа, то говорять про **арифметичні вирази.** Залежно від типу формованих значень визначаються типи виразів.

Операція присвоювання

При декларації змінної відбувається зв'язування імені змінної з областю пам'яті, у якій буде зберігатися її значення. Перш ніж використовувати змінну в програмі, їй необхідно присвоїти значення**.**

Формат:

[Let] ***змінна = вираз***

де символ “=” — знак операції присвоювання;

аргумент *змінна* задає ім'я змінної, якій буде присвоєне значення виразу, розташованого праворуч від знака рівності.

Приклади:

sngFirst = 10

strLastname = "Іванов"

sngResult = sngFirst + 255

strName = "Іванов" & ": " & strTeam

P = X+ 2 >=K And A + B > 3 Or A*X + B*Y=D And (H<=Y And Y<=Q)

Sesia(Current). Name Exam(2)= “Фізика“

Massiv_J(3,4) =Y ^ 2 - 4*A*3 + D

int = 6 **'** *змінній* int *присвоюється значення 6*

Службовець. Прізвище = “Іванов” **'** *Структурній складовій Прізвище користувальницького*

типу Службовець присвоюється прізвище Іванов

TxtFirstName.Text = FirstName **'** *значення змінної FirstName присвоюється елементу Text користувальницького типу даних з ім'ям TxtFirstName.*

Правила використання оператора присвоювання:

▪ *у правій і лівій частинах повинні бути дані одного типу;*

▪ *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#]=2.0399999618530273;

4) при обчисленні виразів операнди двомісцевих арифметичних операцій перетворяться до точності операнда з більш високою точністю, а результат — до точності змінної, якій він присвоюється;

*5) змінній або елементу масиву типу **Variant** у лівій частині може відповідати будь-який тип виразу в правій частині (у комірці пам'яті для зберігання даних типу **Variant** зберігається не тільки значення, але і його тип);*

*6) опція **Let** в операторі використовується для присвоєння значення одного даного користувальницького типу іншому, за*

умови, що типи елементів обох користувальницьких даних збігаються.

Арифметичні операції. Для формування й наступного обчислення виразів служать операції. Кожна операція має свій **пріоритет**:

1-й ранг — обчислення числових функцій;
2-й ранг — унарний мінус;
3-й ранг — піднесення до степеня;
4-й ранг — множення / ділення;
5-й ранг — ділення націло;
6-й ранг — обчислення залишку;
7-й ранг — додавання / віднімання.

Операції одного рангу у виразах виконуються відповідно до правил асоціативності (зліва направо). Для зміни цього порядку можна використати круглі дужки. Вирази, що містяться в дужках, обчислюються в першу чергу.

Для позначення арифметичних операцій використовуються знаки **+ - * / \ MOD.** *Операція \ означає ділення націло (дробова частина відкидається). Операція* ***MOD*** *означає обчислення залишку (ділення по модулю).*

Приклад: 17\2=8 25MOD8=1

Якщо операнди цих операцій речовинні, то вони попередньо округлюються.

Логічні операції та операції відношення. Відношення може мати тільки два результуючі значення — **True** (ІСТИНА) і **False** (ХИБНІСТЬ)

Таблиця 8.4

Операція	***Опис***
<Операнд1>= <Операнд2>	Дорівнює
<Операнд1> <> <Операнд2>	Не дорівнює

<Операнд1> > <Операнд2>	Більше
<Операнд1><<Операнд2>	Менше
<Операнд1> => <Операнд2>	Більше або дорівнює
<Операнд1> <= <Операнд2>	Менше або дорівнює
<Рядок> Like <Маска>	Відповідність масці. Результат —True, якщо рядок відповідає масці
<Операнд1> Is <Операнд2>	Посилання на об'єкт. Результат —True, якщо обидві змінні посилаються на той самий об'єкт

Приклад:

b2-4ac>0 ~ B^2-4*A*C>0; i≠j ~ I<>J; рядок a = рядок b ~ A$=B$

*В **VB 6.0** використовуються такі **логічні операції:***

- ***Not —** логічне заперечення —інверсія **(НІ);***
- ***And —** логічне множення — кон'юнкція **(І);***
- ***Or —** логічне додавання — диз'юнкція **(АБО);***
- ***Xor —** АБО, що виключає**;***
- ***Eqv —** логічна еквівалентність**;***
- ***Imp —** логічна імплікація**.***

Логічні операції** поєднують логічні величини, які можуть набувати два значення**: True —** ІСТИНА (не-нуль) або **False —**ХИБНІСТЬ (нуль).* Результат *логічних операцій* набуває одне із двох значень ***True або ***False.*** Він використовується при ухваленні рішення про подальший хід обчислювального процесу і визначається таблицею 8.5.

Таблиця 8.5

Значення		**Результат виконання**						
Операнд1 (A)	***Операнд2 (B)***	***Not A***	***Not B***	***A And B***	***A Or B***	***A Xor B***	***A EqvB***	***A Imp B***
True	True	False	False	True	True	False	True	True
False	True	True	False	False	True	True	False	True
True	False	False	True	False	True	True	False	False
False	False	True	True	False	False	False	True	True

Установлений такий ***пріоритет*** виконання *логічних операцій* і *операцій відношення*:

1) спочатку виконуються операції відношення **(<, >, =, <>, >=, <=)** в порядку прямування зліва направо.

2) далі *логічні операції* (показані у порядку убування пріоритету): **Not, And, Or, Xor, Eqv, Imp.**

При перетворенні арифметичного типу до логічного 0 перетвориться в False, а інші значення перетворяться в True.

При перетворенні логічного типу до арифметичного False перетвориться в 0, а True в (-1).

Операції для роботи з рядками

В VB 6.0 є тільки одна операція для роботи з рядками — це операція ***конкатенації.*** Конкатенація дає змогу об'єднати значення двох або декількох строкових змінних або строкових констант. Знаком операції конкатенації є символ амперсанд (&) або (+). При конкатенації рядків значення другого рядка додається в кінець першого рядка. Результатом операції є більш довгий рядок, складений з вихідних рядків.

Приклад:

t$=“ЯКІ”+ “ ВАШІ” +“ ДОКАЗИ ?”

Константи

Константи — це об'єкти, значення яких однозначно визначаються написанням, залишаються постійними й не можуть бути змінені під час виконання програми. Константи можуть бути ***іменованими*** й ***неіменованими***. Синтаксис мови визначає **3 типи** констант: ***числові*** (цілі й речовинні)*,* ***логічні****,* ***символьні****.*

Приклади:

75.07 ' числова константа ;

2.7E+6 ' числова константа (дорівнює 2 700 000) ;

«Помилка доступу до бази даних" **'** символьна константа;

#8/12/1999# ' константа типу дата ;
False ' логічна константа.

Числові цілі константи. ***Цілі константи*** — послідовність цифр і спеціальних символів зі знаком або без нього. Передбачені нижчеподані цілі константи:

- *Цілі константи в десятковому поданні*:
  - *одинарної точності* (тип INTEGER) — належать діапазону від -32768 до 32767 і зображуються в пам'яті 2-байтовими двійковими числами. У програмі запис цих констант завершується символом % (цілі короткі).

Приклад: 3%, -19%

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

Приклад: 123897890&, -78989&

- *Цілі константи у вісімковому поданні* — послідовність вісімкових цифр без знака, яким передує &0.

Приклад: &0113, &02

- *Цілі константи в шістнадцятковому поданні* — послідовність шістнадцяткових цифр без знака, яким передує &H.

Приклад: &H123, &HA56E

Числові речовинні константи. Для запису *речовинних констант* використовують дві форми: ***природну*** й ***експоненційну****:*

- Речовинна константа в *природній формі* — послідовність десяткових цифр зі знаком і крапкою, що розділяє цілу і дробові частини.

Приклад: 0.775; -15.04; +11.08

- Речовинна константа в *експоненційній* (показовій) формі (з плаваючою крапкою) має вигляд — 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. 61E+12, 16. 333D-2

0.11E+12 або 1.1E+11 відповідає 0.11*1012.

Класифікація речовинних констант

За обсягом зайнятої пам'яті виділяють:

- Речовинна константа *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!

- Речовинна константа *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, 3241.6 , -1.0965434D12

Логічні константи можуть мати тільки два фіксованих значення: TRUE (істина) або FALSE (хибність).

Символьні (рядкова, текстова) константи служать для зображення окремих знаків і являють собою лексему, що складається із символу (або будь-якої послідовності символів), укладеного в лапки (тип STRING).

Приклад:

"P", "Program", "3.14", "+" ' неіменовані символьні константи.

Максимальна довжина символьної константи — 32767 символів. Якщо заздалегідь відомо, що число символів константи не перевищує n, то тип задається, як STRING* n.

Константи VB 6.0 поділяються:

- на ***символічні*** *або* ***визначені користувачем*** константи — оголошуються оператором Const;
- ***вбудовані (внутрішні)*** *або* ***системні***, що надаються додатками або елементами управління (рисунок 8.2). Вбудовані константи мають префікс VB 6.0.

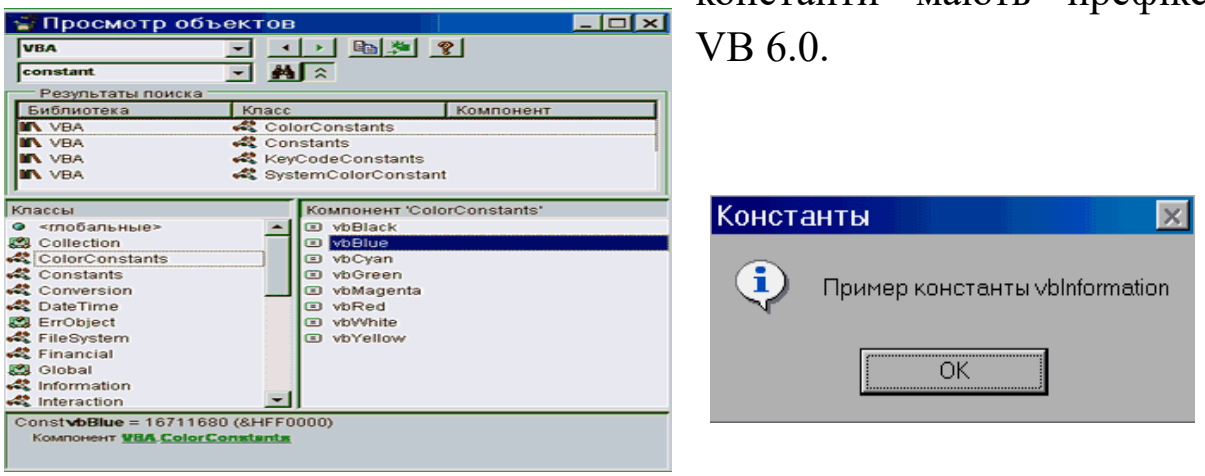


Рисунок 8.2 — Системні константи

Вбудовані константи включаються в тексти програм без попереднього опису. При найменуванні вбудованих констант використовується стандартна угода, що дає змогу визначити, до об’єктів якого додатка належить константа.

Приклад: вбудовані константи, що належать до об’єктів

Access, починаються з префікса ac,

Excel — з префікса xl,

Word — з префикса wd,

VB 6.0 — з префикса vb.

Приклад: використання вбудованих констант**:**

```
MsgBox "Текстове повідомлення", vbInformation
                ‘ константа vbInformation вказує, що у
```

вікні виведення

‘ повинен бути значок “Інформація“

vbYesNoCancel = 3

‘ аргумент функції MsgBox для виведення в діалоговому

‘ вікні командних кнопок Yes, No, Cancel

Для декларування констант служить ключове слово Const.

Формат:

[Public/Private] CONST і*м.’я_константи*[As *тип_даних*]

=*вираз* [,*ім.’я_константи* [As *тип_даних*]=*вираз*],

…

Обчислюється значення *виразу,* і результат присвоюється константі.

Приклад:

Const *PI* AS SINGLE =3.1415926, *conA* = «програмування»

Public Const *conMY_NAME* = “Юра“

Тип константи можна не повідомляти (автоматично одержить тип Integer). За замовчуванням приймається тип, що займає найменший обсяг пам'яті. *Краще явно вказувати тип спеціальними символами в операторах оголошення констант.*

Приклад:

Const ONE& = 1 ‘резервує 4 байти

Const ONE# = 1 ‘резервує 8 байтів, зберігається у вигляді числа подвійної точності із плаваючою крапкою

Перерахування

Перерахування — це список констант. Перед використанням такого списку його необхідно визначити в програмі. Оператор Enum оголошує *Перерахування* та визначає значення його елементів.

Приклад: розглянемо перерахування оцінок, що отримали студенти

```
Enum Ocinka
    Nezadov = 2
    Zadov = 3
    Dobre = 4
    Vidminno = 5
End Enum
```

Функції

Як операнди у програмах поряд з константами, змінними й обумовленими користувачем функціями можна застосовувати функції, визначені в самій мові — *вбудовані функції*. Загальний вид функції:

Ім'я функції (аргумент_1, аргумент_2,.... , аргумент_n)

Функція обчислює й повертає результат залежно від вихідних даних (аргументів). Функції можуть використовуватися *в арифметичних виразах з оператором присвоювання*:

Приклад:

y = Sin(x) ‘ x — аргумент, а обчислений синус від x — результат

y = 2 * Sin(4 * x) + 2

або з *оператором порівняння*:

Sin(x) > Cos(y) + 1

Sin(Pi) = 0

Звертання до вбудованої функції, що повертає значення того або іншого типу, повинно відповідати виразу, у якому до неї звертаються.

Приклад: *в арифметичному виразі можна звертатися до функцій, що повертають значення арифметичних типів, у символьному — символьного типу.*

Звертання до функцій, які не повертають обчислене значення, є окремими операторами програми.

Для звертання до деяких вбудованих функцій потрібно задавати значення аргументу (наприклад, Sin(X+2), де X+2

вираз, що визначає значення аргументу). Для інших вбудованих функцій аргумент задавати не потрібно (наприклад. Now).

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) *рядкові (символьні) функції;*
4) *функції перевірки типів;*
5) *функції перетворення форматів;*
6) *функції для перетворення типів даних;*
7) *функції для роботи з датою й часом.*

Стандартні математичні функції

Таблиця 8.6

Запис на VB 6.0	***Функція***	***Приклад***
SIN(X)	***Синус X***	*sinX, аргумент в радіанах*
COS(X)	***Косинус X***	*CosX, аргумент в радіанах*
TAN(X)	***Тангенс X***	*tgX, аргумент в радіанах*
ATN(X)	***Арктангенс X***	*ArctgX, -π/2<X<π/2,*
ABS(X)	***Модуль X***	*|X|*
SQR(X)	***Квадратний корінь X***	*√X, X>0*
EXP(X)	***Експонента X***	*eX*
LOG(X)	***Натуральний логарифм X***	*lnX, X>0*
FIX(X)	***Усікання до цілого (відкидання дробової частини)***	*FIX(-5.6)=-5 FIX(24.07)=24.00*
INT(X)	***Знаходження найбільшого цілого,***	*INT(15.5)=15 INT(-6.2)=-7*

	що не перевищує X	*INT(-6.7)=-7*
RND(X)	***Генерація псевдовипадкових чисел від 0 до 1***	
ROUND	***Округлення до заданого числа десяткових знаків***	*Round(3.44,1)=3.4* *Round(3.44)=3.4*
SGN(X)	***Знак числа X***	*SGN(X) = -1, якщо X < 0* *SGN(0) = 0, якщо X = 0* *SGN(X) = +1, якщо X > 0*

Формули, для обчислення деяких математичних функцій

Таблиця 8.7

Функція	***Розрахункова формула***
Секанс	***Sec(x)=1/Cos(x)***
Косеканс	***Cosec(x)=1/Sin(x)***
Котангенс	***Cotan(x)=1/Tan(x)***
Арксинус	***Arcsin(x)=Atn(x/Sqr(-x*x+1))***
Арккосинус	***Arccos(x)=Atn(-x/Sqr(-x*x+1))+2* Atn(1)***
Логарифм на підставі N	***LogN(x)=Log(x)/Log(N)***

Стандартні функції обробки символьної інформації

Числові функції символьних аргументів. Значеннями цих функцій є числа, а аргументами — символьні вирази або функції. *Рядкові функції.* Значеннями цих функцій є ланцюжки символів.

Таблиця 8.8

Функція	***Призначення***
Asc	*Повертає ASCII-Код символу*
Chr	*Перетворює ASCII-Код у символ*
InStr, InStrRev	*Здійснюють пошук одного рядка в іншому*
LCase	*Змінює регістр букв вихідного рядка на нижній*
Left	*Повертає зазначену кількість символів з початку рядка*
Len	*Повертає кількість символів у рядку*
LTrim, RTrim,	*Видаляють пробіли, розташовані відповідно на початку,*

Trim	*наприкінці й по обидва боки символьного рядка*
Mid	*Повертає задану кількість символів з довільного місця рядка*
Right	*Повертає зазначену кількість символів з кінця рядка*
Str, CStr	*Перетворюють числовий вираз у рядок*
StrReverse	*Змінює порядок проходження символів у рядку на зворотний*
StrConv	*Змінює регістр букв символьного рядка*
Val	*Перетворюють рядок у числовий вираз*
UCase	*Змінює регістр букв вихідного рядка на верхній*
Space	*Повертає рядок, що складається із зазначеного числа пробілів*
Split	*Перетворює рядок в одновимірний масив, що нумерується з нуля*
Join	*Перетворює масив у рядок*
String	*Повертає рядок, що складається із зазначеного числа повторень того самого символу*
StrComp	*Повертає результат порівняння двох рядків*
Replase	*Знаходить і заміняє в рядку підрядок іншим підрядком*

Усі ці функції в основному використовуються для перетворення даних, що поставляють оператори введення-виведення й обробки символьної інформації. Якщо необхідно, щоб результатом виразу було число, то й всі вихідні дані в цьому виразі повинні бути числами. І також якщо результат — рядок, то й всі вихідні дані повинні бути рядками. У противному випадку можливі або помилки в програмі, або результат не буде відповідати очікуваному.

Функції перетворення типів даних

Таблиця 8.9

Функція	***Опис***	***Приклад***	
		Аргументи	***Результат***
Val(рядок)	*Перетворює рядок у число. Якщо у рядку немає цифр, то результат буде 0*	*Val(“25”)* *Val(“Мир”)* *Val(“01рахунок*	*25* *0* *1*
Str(число)	*Перетворює число в рядок*	*Str(5)*	*5*

		Str(-5)	*-5*

Функції для роботи з датою й часом

Таблиця 8.10

Функція	***Призначення***
Date	*Повертає поточну системну дату, як* ***дд.мм.гг.***
DateAdd	*Додає або віднімає зазначений інтервал до дати. Позитивні значення додають, негативні — віднімають*
DateDiff	*Повертає число заданих інтервалів часу, що лежать між двома зазначеними датами*
Now	*Як і Date повертає поточну дату, але при цьому вказується й поточний час*
Time	*Повертає поточний системний час як* ***hh.mm.ss.***

Форматування чисел, дат і часу

Функція ***Format*** *перетворює числові значення в текстовий рядок і надає можливість управляти зовнішнім виглядом (появою) рядка.*

Format (expression[, format [, firstdayofweek [,firstweekofyear]]])

- ***expression*** *(змінна) задає число, що перетворюється;*
- ***format*** *— рядок, складений із символів, що визначають форматування числа;*
- ***firstdayofweek*** *є константою, що задає перший день тижня;*
- ***firstweekofyear*** *є константою, що задає перший тиждень року.*

Найбільш часто використовувані символи параметру format

Таблиця 8.11

Символ	***Опис***
0	*Цифровий символ-заповнювач; друкує замикаючий або провідний нуль у поточній позиції*
#	*Цифровий символ-заповнювач; ніколи не друкує замикаючих або провідних нулів*
.	*Символ-Заповнювач десяткової крапки*
/	*Символ-Заповнювач для роздільника тисяч*
- + $ ( ) space	*Буквальний символ; символи відображаються точно так, як вони набрані у форматному рядку*

Числові формати

Числові перетворення, подані в таблиці припускають, що в Панелі керування Windows на вкладці ***Регіон і мова*** *обрана опція* ***Англійський (США).***

Таблиця 8.12

Формат	***Результат***
Format(8315.4, "00000.00")	*08315.40*
Format(8315.4, "#####.##")	*8315.4*
Format(8315.4, "##,##0.00")	*8,315.40*
Format(315.4,"$##0.00")	*$315.40*

Символ для десяткової крапки — крапка (.), а символ для роздільника тисяч — кома (,). Однак дійсний символ роздільника визначається установками в Панелі керування Windows.

Вирази

Нагадаємо, що в алгоритмічній мові вираз — це послідовність констант, змінних, індексних змінних (елементів масивів), функцій або будь-яких їхніх комбінацій, утворена за допомогою знаків арифметичних і логічних операцій, операцій відношень, операцій над рядками символів.

Розрізняють *вирази:*

- *арифметичні;*
- *логічні;*
- *символьні.*

Арифметичний вираз — послідовність констант, змінних, функцій, з'єднаних знаками арифметичних операцій. *Результатом його є* ***число****. Арифметичні вирази відповідають загальноприйнятим алгебраїчним виразам. Число або змінна також вважаються арифметичним виразом.*

Приклад:

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

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

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

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

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

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

Усі ці функції в основному використовуються для перетворення даних, що поставляють оператори введення-виведення й обробки символьної інформації.

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 одна за одною зліва направо. Винятком є піднесення до степеня *xyz=**X^(Y^Z).***

Для обчислення кореня довільного степеня використовується еквівалентний вираз

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

Логарифм з довільною основою обчислюється за формулою

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

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

Від'ємні значення підносити до дробового степеня забороняється. Пояснюється це тим, що вираз перед обчисленням автоматично перетворюється як $x^a=e^{a\ln x}$.

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

Особливі ситуації при обчисленні арифметичних виразів:

- ділення на нуль у цілочисловій арифметиці не допускається;
- ділення на нуль у плаваючій арифметиці може давати нескінченність або NAN (англ. Not-a-Number) — один з особливих станів числа з плаваючою крапкою).

Переповнення — видається відповідне повідомлення, і виконання програми припиняється (рисунок 8.3)

.

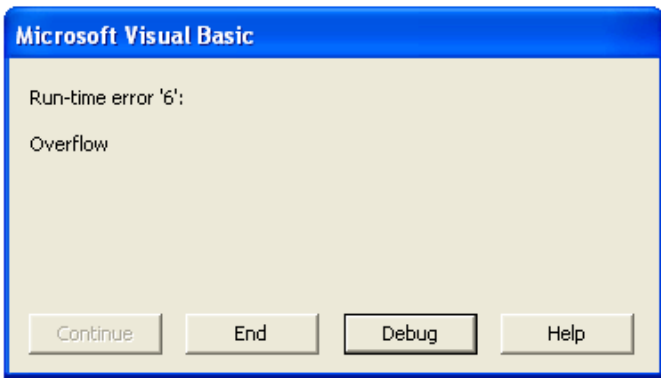


Рисунок 8.3 — Системне повідомлення при переповненні

Символьні вирази

Символьний (рядковий) вираз — послідовність символьних констант, змінних, функцій або будь-яких їхніх комбінацій, з'єднаних знаком конкатенації (англ.Concatenate — зчіплювати).

Формат:

strвираз_1 & strвираз_2 [. . . & strвираз_N]

де **strвираз —** будь-який рядок (рядкова змінна, рядкова константа або функція обробки рядків).

& (знак амперсанда) між рядковими виразами — вказує, що здійснюється конкатенація цих виразів. Знак амперсанда відділяється від виразів пробільним символом.

В одному операторі можна поєднувати будь-яку кількість рядкових виразів.

Приклад:

```
strResult = “ Студент” & “ Іванов”
        ‘Поєднуються дві неіменовані рядкові константи.
        ‘Результатом операції конкатенації
        ‘буде значення “Студент Іванов”.
```

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

Логічні вирази

Логічний (умовний) вираз — послідовність арифметичних або символьних виразів, з'єднаних знаками відношень та/або логічних операцій. Результат логічного виразу набуває одне з двох значень: True або False. Якщо порівняння здійснюється над числами, то припустиме порівняння цілого і речовинного типів. Символьні величини можна порівнювати тільки із символьними.

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)

8.2 Методи введення-виведення. Діалогові вікна

Одержання даних від користувача, збереження їх у змінній і відображення результатів дій, виконаних над ними, є основними елементами, необхідними для написання інтерактивних процедур. Для реалізації інтерактивних процедур у проектах VB 6.0 застосовуються два різновиди діалогових вікон:

- *вікна повідомлень* MsgBox, що виводять повідомлення для користувача;
- *вікна введення* InputBox, що дають змогу користувачеві вводити інформацію.

8.2.1 Одержання даних від користувача (діалогове вікно InputBox)

Щоб одержати вихідні дані від користувача процедури, використовується функція ***InputBox***. Вона відображає *діалогове вікно*, яке *містить* ***текст,*** *що запитує користувач*, *щоб ввести деяке значення,* і ***текстове вікно*** *для введення цього значення*.

Це вікно містить також стандарт: елементи діалогу — **командні кнопки ОК (**підтвердження дії**)** і **Cancel (**скасування дії**).**

Формат:

String_var InputBox (*символьний вираз1*

[,*символьний вираз2*] [,*символьний вираз3*

[,*число1*[,*число2*]

[,*символьний вираз4, символьний вираз5*])

де **String_var** — будь-яка змінна, яка може зберігати рядок (або змінну типу **String** чи **Variant**);

символьний вираз1 (Prompt) — текст (повідомлення), який відображається в діалоговому вікні як запит. Виконання програми припиняється до введення повідомлення (текстової або числової константи) і натискання кнопки Ok;

символьний вираз2 (*Title)* — текст, який відображається в заголовку діалогового вікна. Максимальна довжина тексту 1024 символи. У цей текст можна вставити як роздільники рядків переведення каретки Chr(13), переведення рядка Chr(l0) або їхню комбінацію. Якщо опустити цей аргумент, VB 6.0 відображає в рядку заголовка діалогового вікна InputBox слово "Input";

символьний вираз3 (*Default*) — текст, який відображається в полі введення як повідомлення за попереднім визначенням (за замовчуванням). Якщо параметр відсутній, рядок залишається порожній;

число1 (*xpos*) — відстань по горизонталі у твіпах між лівою межею кута вікна додатка й лівим краєм екрана;

число2 (*ypos*) — відстань по вертикалі у твіпах між верхньою межею вікна додатка й верхнім краєм екрана. Якщо *число1* відсутнє, то вікно діалогу вирівнюється щодо центра екрана. Якщо *число2* відсутнє, то вікно діалогу розміщується по вертикалі приблизно на 1/3 висоти екрана;

символьний вираз4 (Helpfile) — посилання на файл довідкової системи*;*

символьний вираз5 (Context) — посилання на зміст у файлі довідкової системи.

Приклад: *наведено код модуля з однією процедурою. У коді є оголошення константи і змінної модульного рівня. Ця*

процедура обчислює площу кола; значення радіуса кола одержує від користувача процедури (рисунок 8.4.)

```
' Одержання вхідних даних за допомогою оператора InputBox
Const Pi As Single =3.14 ' значення Pi
Dim CircleArea As Single ' зберігає обчислену площу кола
Sub Calc_CircleArea()
Const BoxTitle = "Площа кругу"
Dim Radius As Single, Temp As String
Temp = InputBox("Введіть радіус кола", BoxTitle)
Radius = CSng (Temp)
CircleArea = Pi * (Radius * Radius)
MsgBox CircleArea, , BoxTitle
End Sub
```

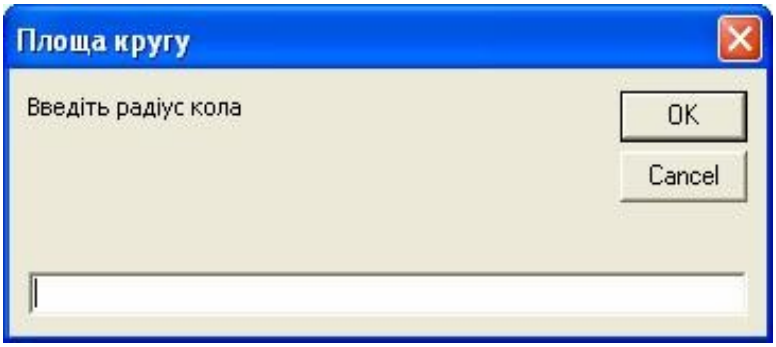


Рисунок 8.4 — Діалогове вікно введення,
що відображає функція InputBox

8.2.2 Відображення результатів дій (діалогове вікно *MsgBox*)

Діалогове вікно ***MsgBox*** дає можливість виводити на екран прості повідомлення. ***MsgBox*** відображає повідомлення в діалоговому вікні, очікує натискання на кнопку користувачем і повертає значення (тип Integer), що вказує, яку кнопку вибрав користувач. Діалогове вікно повідомлення не вимагає проектування й викликається із програми ***командою MsgBox.***

Формат:

MsgBox (*символьний вираз1*[,*число* [,*символьний вираз2*]
[,символьний вираз3, символьний вираз4)]

де *символьний вираз1 (Prompt)* — текст, який відображається як повідомлення в діалоговому вікні;

число (Buttons) — вид додаткових елементів, що вводять автоматично у вікно діалогу;

символьний вираз2 (Title) — текст, який відображається в заголовку діалогового вікна;

символьний вираз3 (Helpfile), символьний вираз4 (Contex) — тексти контекстно-залежної допомоги.

Вікно повідомлення може також викликатися за допомогою ***функції MsgBox.*** Значення, що повертається, залежить від кнопки, натиснутої користувачем (таблиця 8.14). Формат функції повністю збігається з форматом відповідної команди.

Приклад:

```
Dim Rc as integer     'код, що повертається функцією
Rc=MsgBox("Вітаємо " + YourName + "! Ви згодні тестуватися?",
vbYesNo + vbQuestion," Тестування!!!")
```

Аргумент **Buttons** може набувати значення-константи, наведені у таблиці 8.13.

Таблиця 8.13

Значок	***Константа***	***Значення***	***Опис***
	vbOKOnly	*0*	*Відображати тільки кнопку ОК*
	vbOKCancel	*1*	*Відображати кнопки ОК і Cancel*
	vbAbortRetryIgnore	*2*	*Відображати кнопки Abort, Retry і Ignore*
	vbYesNoCancel	*3*	*Відображати кнопки Yes, No і Cancel*
	vbYesNo	*4*	*Відображати кнопки Yes і No*
	vbRetryCancel	*5*	*Відображати кнопки Retry і Cancel*
	***vbCritical***	*16*	*Відображати значок Critical Message — помилка*
	***vbQuestion***	*32*	*Відображати значок Warning Query — запит*
	***vbExclamation***	*48*	*Відображати значок Warning Message — попередження*
	***vbInformation***	*64*	*Відображати значок Information*

			Message —інформація

Приклад: *Введіть у командному вікні середовища проектування Immediate таку команду й натисніть клавішу <Enter>:*

MsgBox“Вітаємо Вас!", vbYesNo+ vbExclamation, “Вікно повідомлень".

У відповідь одержите діалогове вікно, показане на рисунку 8.5.

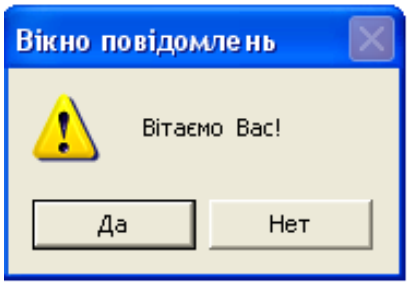


Рисунок 8.5 — Діалогове вікно,
що відображає функція MsgBox

Приклад: *значення констант можна додавати одне до одного, щоб досягти бажаного результату. Наприклад, щоб у вікні були кнопки OK та Cancel, а також значок інформаційного повідомлення, можна скористатися будь-яким з наведених варіантів*

DigDef= vbOKCancel

DigDef =1+64

Далі значення змінної DigDef використовується в MsgBox.

Залежно від вибору кнопки діалогове вікно ***MsgBox*** повертає одне зі значень, заданих системними константами. Це необхідно для аналізу натиснутої кнопки й виконання відповідних дій у програмі. У вихідному коді для цього можна використовувати константи, зазначені в таблиці 8.14.

Зверніть увагу — функція ***InputBox*** повертає рядок, а функція ***MsgBox*** — числове значення цілого типу.

Таблиця 8.14

Кнопка	Константа	Значення при натисканні на кнопку
OK	vb0k	1
Cancel **(***Скасування****)***	vbCancel	2
Abort (*Стоп* ***)***	vbAbort	3
Retry (*Повтор****)***	vbRetry	4
Ignore ***(****Пропустити****)***	vbIgnore	5
Yes (*Так* ***)***	vbYes	6
No(*Ні* ***)***	vbNo	7

8.3 Оператори управління

Найчастіше в певному місці програми необхідно виконувати ті або інші оператори, залежно від деяких умов. В VB6.0 це можливо реалізувати за допомогою управляючих конструкцій (або структур), які у свою чергу складаються зі структур прийняття рішень і циклів, які містять:

- оператори передачі управління;
- оператори вибору;
- оператори організації циклів.

8.3.1 Оператор безумовного переходу

Оператори безумовного переходу застосовуються в програмі для реалізації безумовних алгоритмічних конструкцій. Вони виконують перехід з однієї ділянки програми на будь-яку іншу без будь-якої умови.

Формат:

GoTo *мітка*

де **GoTo** — службове слово;

мітка — поміщається ліворуч від програмного оператора й відділена від нього двокрапкою.

Приклад:

```
GoTo m1   'Оператор безумовного переходу до оператора з міткою m1
m1: Text1.Text = "це мітка"   'Оператор з міткою m1
```

*Варто вказати, що для одержання гарного стилю програмування варто уникати застосування оператора **GoTo**, тому що в цьому випадку погіршується читабельність і розуміння програми.*

Як ви вже знаєте, конструкції виконуються в тій послідовності, у якій вони записані в програмі, однак досить часто вам потрібно змінити порядок виконання команд залежно від виконання (або невиконання) певної умови.

Існують конструкції, призначені для управління порядком виконання команд.

IF...THEN...ELSE
SELECT CASE

8.3.2 Оператор умовного переходу (IF...THEN...ELSE)

Конструкція використовується в тому випадку, якщо необхідно, щоб група операторів виконувалася при дотриманні певних умов. Здійснює "розгалуження" програми, тобто передає управління на ту або іншу "гілку" обчислювального процесу залежно від результату виконання умови. Підставою для прийняття рішень в управляючих конструкціях є ***умовні вирази***.

Оператор**,** що відповідає конструкції

IF...THEN...ELSE (*якщо...то...інакше*)

можна записати в ***лінійній*** *або* ***блоковій*** формі:

а) ***лінійна форма*** — доцільно використовувати, якщо перевіряється одна умова.

IF<*умова*>THEN<*оператори_так*>[ELSE<*оператори_ні*>]

де *умова* — логічний вираз, результат якого набуває значення ХИБНІСТЬ або ІСТИНА. *Умовою може бути навіть арифметичний вираз. У цьому випадку результат набуває значення ХИБНІСТЬ, якщо арифметичний вираз — нуль, і значення ІСТИНА — в інших випадках;*

оператори_так — виконуються, якщо результат перевірки умови — ІСТИНА;

оператори_ні — виконуються, якщо результат перевірки умови — ХИБНІСТЬ.

Якщо конструкція *оператори* складається з декількох операторів VB6.0, то вони розділяються двокрапкою і повинні обов'язково розміщатися в одному програмному рядку.

Лінійна форма запису може бути ***повною*** або ***скороченою***.

Повна лінійна форма запису (альтернативна) припускає наявність у записі гілки ELSE (рисунок 8.6).

Приклад:

IF<*умова*>THEN<*дія_1*>
ELSE<*дія_2*> <*дія_3*>

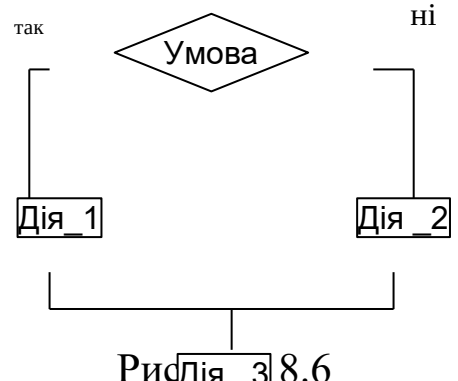


Рис. 8.6

Якщо результат перевірки умови — ІСТИНА, то виконується дія_1, а далі — дія_3 (наступний рядок).

Якщо результат перевірки умови — ХИБНІСТЬ, то виконується дія_2, а далі — дія_3 (наступний рядок).

Скорочена лінійна форма запису (безальтернативна) є найпростішою формою (рисунок 8.7). Вона зручна, коли частина ELSE відсутня.

Приклад:

IF<*умова*>THEN<*дія_1*>

<*дія_3*>

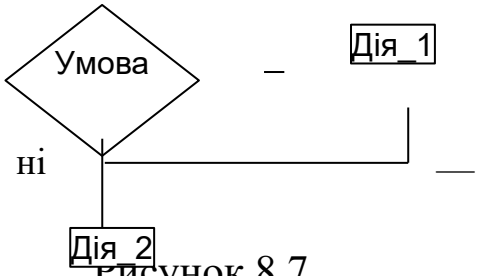


Рисунок 8.7

Якщо результат перевірки умови ІСТИНА, то виконується дія_1, а далі дія_2 (наступний рядок).

Якщо результат перевірки умови ХИБНІСТЬ, то виконується тільки дія_2.

Приклад: *ввести числа (x,y,z). Визначити, чи буде найбільше з них парним. Вивести відповідне повідомлення.*

```
CLS
x= Val(InputBox("Введіть число x "))
y= Val(InputBox("Введіть число y "))
z= Val(InputBox("Введіть число z "))
If x>y Then max=x Else max=y
If z>max Then max=z
Print " максимальне число "; max
If max mod 2 =0 Then Print " парне" Else Print "непарне"
```

б) блокова форма

IF <*умова1*> THEN

[*блок_операторів1*]

[ELSEIF < *умова 2*> THEN

[*блок_ операторів*2]].

...
[ELSE
[*блок операторівn*]]
END IF

де *умова1, умова2* — будь-які вирази, що можуть бути оцінені як ІСТИНА або ХИБНІСТЬ;

блок_ операторів1, блок_ операторів2,.... — один або декілька операторів в одному або декількох рядках.

Перевіряється <*умова1*>, якщо результат перевірки умови ІСТИНА, виконуються оператори блока **THEN**; якщо ХИБНІСТЬ — аналізується кожна умова **ELSEIF** (рисунок 8.8).

При виконанні будь-якої із умов буде реалізований відповідний *блок_операторів*. Якщо жодна з умов **ELSEIF** не виконується, управління передається блоку **ELSE**, а якщо його немає — операторові, розташованому за **END IF**. Блокова структура може мати будь-яку кількість умов **ELSEIF**. Виконання кожного з альтернативних блоків автоматично завершується переходом на **END IF** (рисунок 8.8).

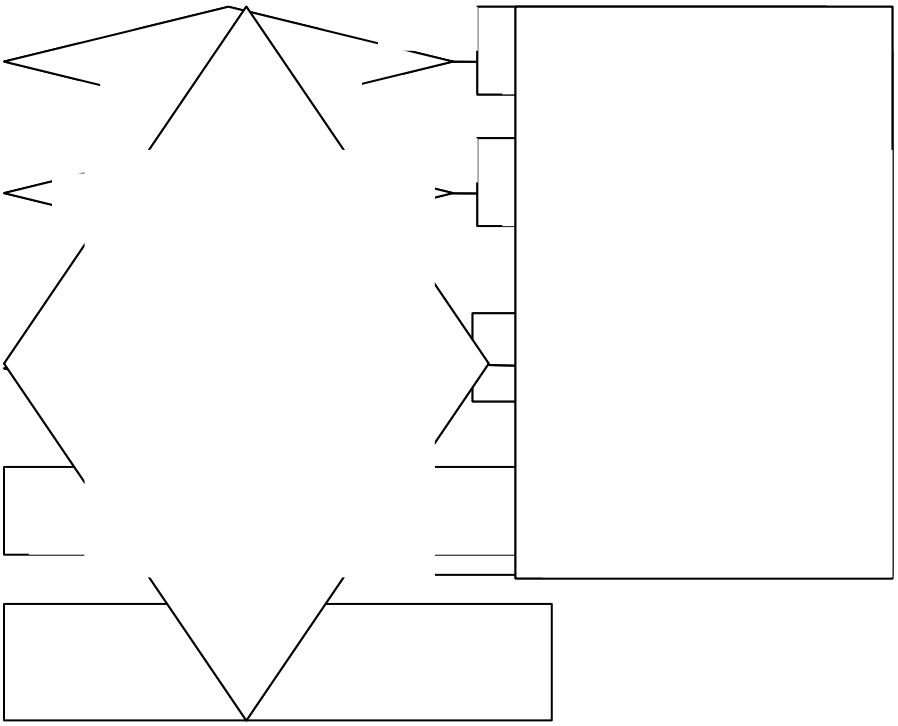


Рисунок 8.8

Приклад: *наведений код програми визначає кількість правильних відповідей і виставляє оцінку*

```
If CorrectAnsver.Text >=8 Then
    Ball.Text = ”Відмінно”
ElseIf CorrectAnsver.Text >=6 Then
    Ball.Text = “Добре”
ElseIf CorrectAnsver.Text >=4 Then
    Ball.Text = “Задовільно”
Else
    Ball.Text = “Незадовільно”
End If
```

8.3.3 Оператор вибору (SELECT CASE)

Оператор **SELECT CASE** застосовується, коли одна величина бере участь у декількох логічних порівняннях і визначає, який блок операторів буде виконуватися. Його доцільно використовувати при наявності складних умов, коли необхідно виконати один з декількох *блоків_операторів* залежно від результатів перевірки цих умов.

Формат:

```
SELECT CASE <вираз вибору> (<тест-вираз>)
    CASE <список_виразів1>
        [ блок_операторів1 ]
    [ CASE <список_виразів2>
        [ блок_операторів2 ] ]...
    [ CASE ELSE
        [ блок_операторівn ] ]
END SELECT
```

Способи запису умовних виразів у блоках CASE

Аргументи списку виразів можуть набувати кожну з таких форм або їх комбінацію і повинні розділятися комами:

- *значення* — вираз і значення перевіряються на рівність один одному;
- *значення1 TO значення 2* — перевіряється, чи належить *тест-вираз* області [*значення2 — значення1*] (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.

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

- Якщо *вираз_ вибору* потрапляє в діапазон, зазначений в CASE, програма виконує відповідний *блок_операторів*.
- 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 SELEC**T.
- Якщо *вираз_вибору* задовольняє декільк… умовам **CASE**, виконується *блок_операторів*, що й…
- Блоки **… CASE** мож… жний блок пов…

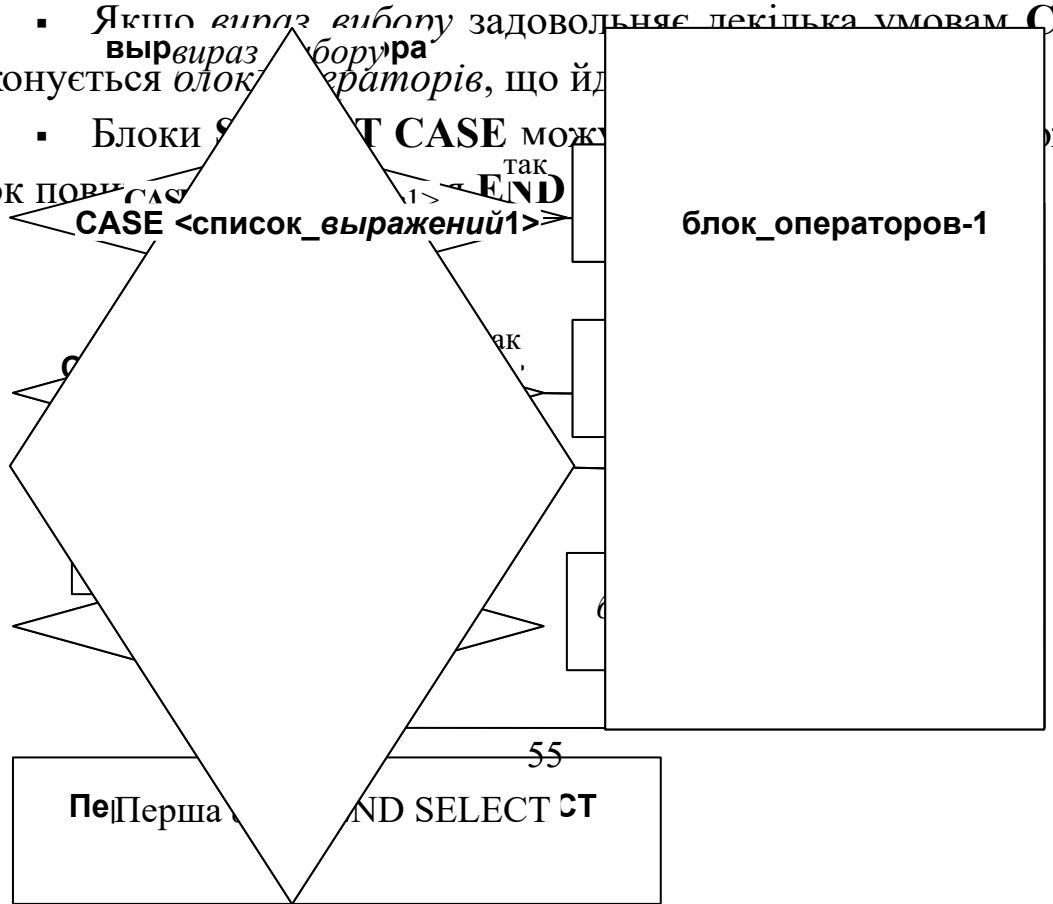


Рисунок 8.9

Приклад: *програма підраховує кількість цифр у введеному числі. Результатом повинно бути ціле число від 1 до 999*

```
CLS: x = Val(InputBox("Введіть число x "))
Select Case x
 Case 1, 2, 3, 4, 5, 6, 7, 8, 9
 Label1.Caption = "Одна цифра"
 Case 10 To 99
 Label1.Caption = "Дві цифри"
 Case Is > 999
 Label1.Caption = "Введено більше число"
 Case Is <= 0
 Label1.Caption = "Введено менше число"
 Case Else
 Label1.Caption = "Три цифри"
End Select
```

8.3.4 Оператори організації циклів

Існують два основних типи операторів циклів — для реалізації ***циклів з лічильником*** (з відомим числом повторень) і ***циклів з умовою*****:**

- *цикли з лічильником* використовують у тих випадках, коли необхідно виконати деякі дії певне число раз;
- *цикли з умовою* застосовуються тоді, коли деякі дії в програмі повинні повторюватися доти, поки виконується певна умова або доти, поки не буде виконана певна умова.

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

Оператори циклу:

FOR... NEXT

FOR EACH...NEXT

DO ... LOOP

Цикли можна реалізувати за допомогою умовних операторів, але це досить незручно.

Цикли з лічильником

Оператор **FOR ... NEXT** дає змогу 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:

For <*параметр циклу***> =<***початок***> To <***кінець***>[Step <***збільшення***>]**

[*тіло циклу***]**

Next [*параметр циклу***]**

де **For** (*для*) — визначає початок циклу;

To (*до*) — визначає кінець циклу;

Step (*крок*) — збільшення параметру циклу;

Next (*наступний*) — кінець;

параметр циклу — числова змінна, яка визначає число повторень циклу;

початок, кінець — арифметичні вирази, які визначають початкове і кінцеве значення параметра циклу;

збільшення — числова константа або арифметичний вираз, що визначає зміну параметра циклу на кожному кроці циклу. Якщо *збільшення* не задано, то за замовчуванням воно приймається рівним одиниці;

тіло циклу (блок операторів) — набір операторів, що виконуються багаторазово. *Тіло циклу* повторюється стільки разів, скільки визначено значеннями *початок, кінець, збільшення.*

У процесі виконання 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 *збільшення* >0 і *параметр циклу* ≤ *кінець* або *збільшення* <0 і *параметр циклу* ≥ *кінець*, то виконується *тіло циклу*; значення *параметра циклу* змінюється на величину *збільшення* і повторюється пункт 3. У протилежному випадку виконуються програмні рядки, що розташовані після **NEXT**. *При збільшенні = 0 цикл стає нескінченним*.

Правила застосування оператора **FOR ... NEXT:**

- *параметр циклу*, зазначений у **FOR**, ***повинен збігатися*** з параметром, зазначеним у **NEXT**;
- *параметр циклу* — арифметична змінна, він ***не може бути елементом масиву або елементом користувальницького типу даних;***
- ***не можна передавати управління усередину циклу*** — у цьому випадку *параметр циклу* не визначений;

- *параметр циклу*, *початок*, *кінець*, *збільшення* ***не можна змінювати усередині циклу***;
- *можна* ***виходити із циклу природно*** (тобто після виконання його задане число разів), а також ***за допомогою управляючих операторів*** з будь-якого оператора *тіла циклу*;
- можна 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 циклу*** *дорівнює кінцевому значенню плюс збільшення*;
- із циклу ***припустиме звертання до підпрограми*** з наступним поверненням у нього.

Приклад: *програма обчислення суми натурального ряду чисел від 1 до N*

```
Private Sub btnStart_Click ()
Dim int_nCounter, int_nCountVar, int_nCounterSum As Integer
int_nCounter = Val(InputBox("Введіть кількість чисел "))
int_nCounterSum = 0

For int_nCountVar = 1 To int_nCounter Step 1
    int_nCounterSum = int_nCounterSum + int_nCountVar
Next int_nCountVar

MsgBox " Сума дорівнює " + Str (int_nCounterSum)
End Sub
```

Цикли **FOR ... NEXT** можуть бути ***вкладеними***. Кожний вкладений цикл повинен мати свій ідентифікатор 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 VB.

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
```

Оператор **FOR EACH...NEXT** — специфічна форма циклу, схожа на цикл For...Next, призначена для виконання деякої операції з кожним об'єктом, що входить до складу деякої колекції об'єктів (наприклад, елементами на формі). Такою операцією, наприклад, може бути виклик методу або присвоювання значення властивості, обробка всіх елементів деякого масиву.

Вона особливо зручна в тому випадку, коли кількість оброблюваних елементів заздалегідь не відома.

Формат:

For Each *Ім'яОб'єкта* In *Ім'яКолекції*
операції над об'єктами
Next *Ім'яОб'єкта*

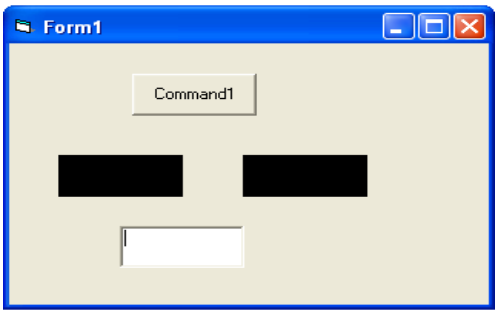
Приклад: *у циклі пробігаємо всі елементи в масиві A. Якщо елемент масиву більше 10, виводимо його на екран.*

```
Dim z As Integer
Dim A(10) As Integer
 'Заповнюємо масив A
 ……..
 For Each z In A
 If z > 10 Then
```

```
Console.WriteLine(z)
End If
Next
```

Приклад: *показано, як змінити властивість BackColor у всіх написів (Label), що знаходяться на формі. Me тут — поточна форма. Тобто не обов'язково використовувати повне ім'я форми для доступу до її властивостей.*

```
Dim x As Object
 For Each x In Me.Controls
If TypeName(x) = "Label" Then
x.BackColor = 0
End If
 Next x
```



При використанні конструкції ***For Each...Next*** *необхідно мати на увазі, що для набору об'єктів параметр* ***Ім'яОб'єкта*** *може бути тільки змінною типу Variant, загальною змінною типу Object або об'єктом, перерахованим в Object Browser. Для масивів параметр елемента може бути тільки змінною типу Variant.*

Цикли з умовою

Головною особливістю таких циклів є наявність умови, яка може бути будь-яким логічним виразом, що набуває значення True або False.

Оператор **DO ... LOOP** має 4 форми запису:

Формат:

Цикл із передумовою	**Цикл із післяумовою**
Do While | Until *умова*	**Do**
тіло циклу	*тіло циклу*
[Exit Do]	**[Exit Do]**
Loop	**Loop While | Until** *умова*

де **DO** — виконати;

WHILE — *поки*, повторюється тіло циклу, поки *умова* виконується (правильна);

UNTIL — *поки не*, повторюється тіло циклу, поки *умова* не стане виконуватися (не стане правильною);

LOOP — *цикл*;

умова — набуває значення ІСТИНА або ХИБНІСТЬ.

У **циклі з передумовою**, який ще називається “перевірка угорі”, у*мова* перевіряється на початку циклу, тому *тіло циклу* може взагалі жодного разу не виконатися, якщо результат першої перевірки — ХИБНІСТЬ при застосуванні **WHILE** (рисунок 8.10) або ІСТИНА при застосуванні **UNTIL** (рисунок 8.11).

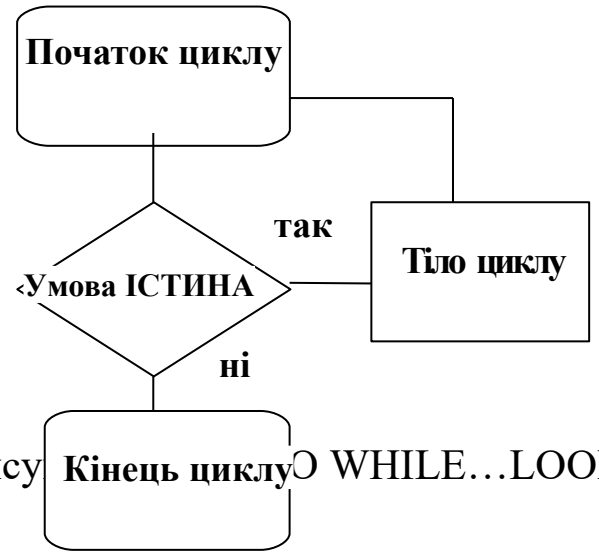


Рисунок 8.10 — Схема WHILE…LOOP

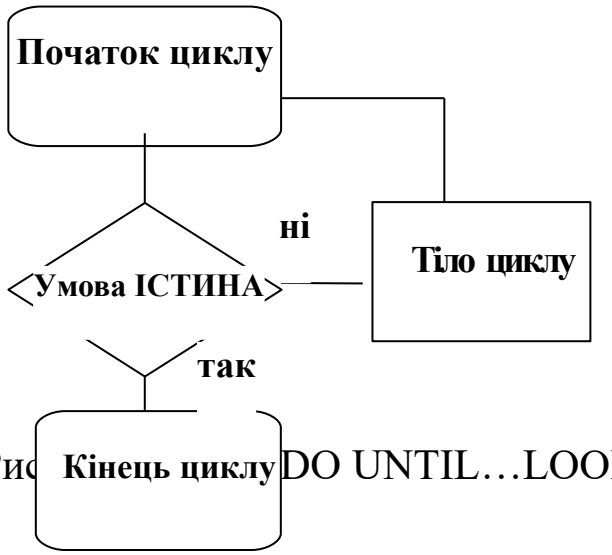
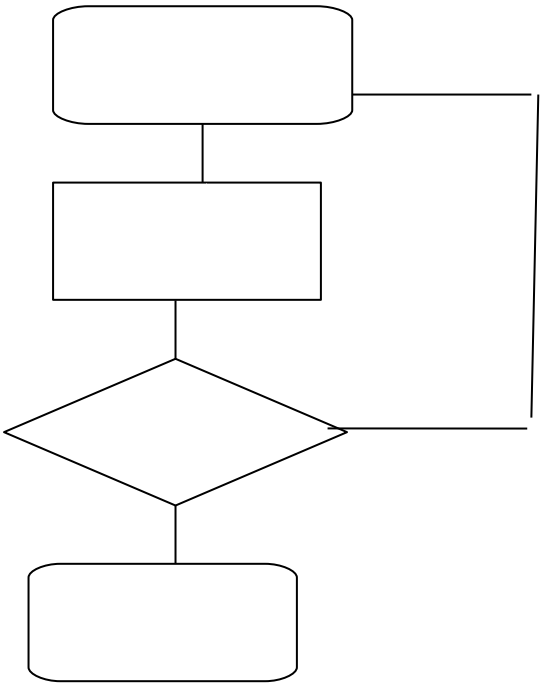
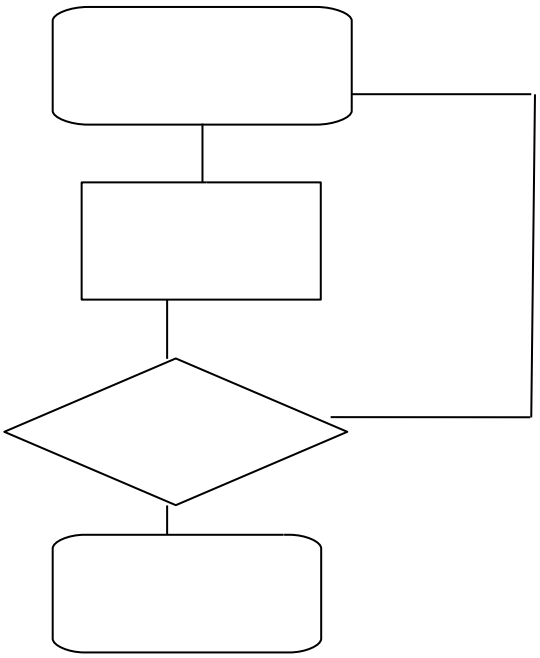


Рисунок 8.11 — Схема DO UNTIL…LOOP

У **циклі з післяумовою**, який ще називається “перевірка унизу”, перевірка умови виконується наприкінці циклу, у зв'язку із цим тіло циклу завжди виконується хоча б один раз (рисунки 8.12, 8.13).





Наведемо приклади реалізації задачі обчислення суми натурального ряду чисел за допомогою цього оператора.

Приклад: *обчислення суми натурального ряду чисел*

Цикл Do Loop з передумовою

```
Private Sub btnStart_Click ()
Dim int_nCounter, int_nCountVar, int_nCounterSum As Integer
int_nCounter = Val(InputBox("Введіть кількість чисел "))
int_nCounterSum = 0
int_nCountVar = 1

Do While int_nCountVar <= int_nCounter
int_nCounterSum = int_nCounterSum + int_nCountVar
int_nCountVar = int_nCountVar + 1
Loop

MsgBox " Сума дорівнює " + Str$(int_nCounterSum)
End Sub
```

Цикл Do Loop з післяумовою

```
Private Sub btnStart_Click ()
Dim int_nCounter, int_nCountVar, int_nCounterSum As Integer
int_nCounter = Val(InputBox("Введіть кількість чисел "))
int_nCounterSum = 0
int_nCountVar = 1

Do
    int_nCounterSum = int_nCounterSum + int_nCountVar
    int_nCountVar = int_nCountVar + 1
Loop Until int_nCountVar > int_nCounter

MsgBox " Сума дорівнює " + Str$(int_nCounterSum)
End Sub
```

За допомогою оператора **Exit** можна здійснити достроковий вихід із циклу незалежно від значення, що має в цей момент умова виходу.

Приклад:

```
Dim n As Integer
n = 10
Do While n > 1
    n = n - 1
    Debug.Print n
    If n = 5 Then Exit Do ' Якщо лічільник = 5, то виходимо з циклу
Loop
```

Оператор **DO ... LOOP** у спрощеному вигляді

DO

[*тіло циклу*]

LOOP

створює нескінченний цикл. З нього можна вийти, використовуючи оператор **EXIT DO** або інший управляючий оператор.

Приклад:

```
Do While 2 > 1
   Degub.Print "Нескінчений цикл"
Loop
```

Якщо випадково вийшов такий цикл, то вийти з нього можна при натисканні Ctrl+Break. Але це працює тільки в середовищі розробки.

БІБЛІОГРАФІЧНИЙ СПИСОК

1 Браун, С. Visual Basic 6 [Текст]: учебный курс /С. Браун. — С.Пб.: Питер, 2006. — 574 с.

2 Основи програмування мовами високого рівня [Текст]: навч. посібник / В.М. Бутенко, В.С. Меркулов, О.В. Казанко, О.В. Чаленко. — Харків: УкрДАЗТ, 2009. — 206 с.

3 Вирт, Н. Систематическое программирование [Текст] / Н. Вирт. — М.: Мир, 1977. — 183 с.

4 Волченков, Н.Г. Программирование на Visual Basic 6 [Текст]: в 3 ч. / Н.Г. Волченков. — М.: ИНФРА-М, 2000. — 280 с.

5 Голуб, А.И. Правила программирования С и С++ [Текст] / А.И. Голуб. — М.: БИНОМ, 2001. – 272 с.

6 Грэхем, И. Объектно-ориентированные методы [Текст]: принципы и практика / И. Грэхем. – 3-е изд. — М.: Вильямс, 2004. — 880 с.

7 Дал, У. Структурное программирование [Текст] / У. Дал, Э. Дейкстра, К. Хоор. — М.: Мир, 1973. — 247 с.

8 Дейкстра, Э. Дисциплина программирования [Текст] / Э.Дейкстра. — М.: Мир, 1978. — 275 с.

9 Зелковиц, М. Принципы разработки программного обеспечения [Текст] / М. Зелковиц, А. Шоу, Дж. Гэннон. — М.: Мир, 1982. — 368 с.

10 Иванова, Г.С. Основы программирования [Текст]: учеб. для вузов / Г.С. Иванова. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. — 416 с.

11 Иванова, Г.С. Технология программирования [Текст]: учеб. для вузов / Г.С. Иванова. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. — 320 с.

12 Информатика и ИКТ. Задачник-практикум [Текст]: в 2 т. / под ред. И.Г. Семакина, Е.К. Хеннера. — М. Лаборатория базовых знаний, 2000.

13 Інформатика, комп'ютерна техніка, комп'ютерні технології [Текст]: посібник / за ред. О.І. Пушкаря. — К.: Академія, 2001.

14 Йодан, Э. Структурное проектирование и конструирование программ [Текст] / Э. Йодан. — М.: Мир, 1979. — 415 с.

15 Керниган, Б. Практика программирования [Текст] / Б. Керниган, Р. Пайк. — С.Пб.: Невский диалект, 2001. — 381 с.

16 Максимов, Н.А. Азбука программирования на Visual Basic [Текст]: практикум / Н.А. Максимов. — Чебоксары, 2007. — 64 с.

17 Основи алгоритмізації базових обчислювальних процесів [Текст]: навч. посібник / В.С. Меркулов, В.О. Гончаров, І.Г. Бізюк, В.М. Бутенко, О.В. Головко . — Харків: УкрДАЗТ, 2008. — 163 с.

18 Райтингер, М. Visual Basic 6.0 [Текст] / М. Райтингер, Г. Муч. — К.: BHV, 2000. — 288 с.

19 Семакин, И.Г. Информатика [Текст]: структурированный конспект базового курса / И.Г. Семакин, Г.С. Вараксин. – М.: Лаборатория базовых знаний, 2001.

20 Синтес, А. Освой самостоятельно объектно-ориентированное программирование за 21 день [Текст] / А. Синтес. — М.: Вильямс, 2002. — 672 с.

21 Тассел, Д.В. Стиль, разработка, эффективность, отладка и испытание программ[Текст] / Д.В. Тассел: пер. с англ. — 2-е изд., испр.— М.: Мир, 1985.—332 с, ил.

22 Філіппенко, І.Г. Програмування інженерно-технічних задач в середовищі QBasic [Текст]: конспект лекцій / І.Г. Філіппенко, В.О. Гончаров, В.С. Меркулов. — Харків: УкрДАЗТ, 2007. – ч. 3. — 134 с.

23 [http://www.alexsoft.ru Visual Basic 6.0.]

24 [http://www.ctc.msiu.ru/materials/Book/node82.html Основные концепции ООП]

25 [http://www.weblibrary.biz/c-sharp/princypy