МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

О.В. СОБОЛЕНКО, О.А. ГУЛЯЄВА, Г.А. ПАВЛЕНКО

ОСНОВИ ПРОГРАМУВАННЯ

Друкується за Планом видань навчальної та методичної літератури,
затвердженим Вченою радою НМетАУ
Протокол № 1 від 27.01.2017

Дніпро НМетАУ 2017

УДК 004(075.8)

Соболенко О.В., Гуляєва О.А., Павленко Г.А. Основи програмування: Конспект лекцій. − Дніпро: НМетАУ, 2017. – 51 с.

Розглянуті методи програмування на мовах VBA, C, C++ у середовищі Builder.

Призначений для студентів напряму 6.050202 – автоматизація та комп'ютерно-інтегровані технології та спеціальності 151 – автоматизація та комп'ютерно-інтегровані технології (бакалаврський рівень).

Іл. 12. Бібліогр.: 6 найм.

Друкується за авторською редакцією.

Відповідальний за випуск Г. Г. Швачич, д-р техн. наук, проф.

Рецензенти : І.Г. Тригуб, канд. техн. наук, доц. (НМетАУ)
О.Г. Холод, канд. техн. наук, проф. (Університет ім. Альфреда Нобеля)

Тема 1. Середовище VBA (Visual Basic for Application)

*Об'єктно-орієнтоване програмування (**ООП**) є однією з кращих технологій при створенні великих програмних проектів.*

VBA функціонує тільки у складі Office додатків, таких як Excel, Word, Access та ін. Запуск редактора VBA з середовища Excel можливий наступними способами:

- пункт меню **Сервіс → Макрос → Редактор Visual Basic;**
- за допомогою гарячих клавіш **[ALT+F11]**;
- панель інструментів **Visual Basic → Редактор Visual Basic;**
- панель інструментів *Елементи управління* **→ початковий текст.**

Повернення з редактора **VB** в робочу книгу:
панель інструментів **Standard →View Microsoft Excel** .
Основні компоненти **VBA :**

- вікно проекту (Prodject - VBA Prodject);
- вікно редагування коду (Code);
- вікно властивостей (Properties);
- вікно форми (UserForm);
- панель інструментів (ToolBox).

Виклик вікон може бути здійснений за допомогою пункту меню **View**.

Excel і VBA можуть знаходитися в двох режимах.

Режим конструктора. У цьому режимі розташовуються об'єкти, редагуються їх властивості, пишеться програмний код. Перехід в режим конструктора: панель інструментів *Елементи управління → інструмент режим конструктора*

Режим виконання – це звичайний режим, в якому виконуються обчислення.

Елементи управління. Властивості, події і методи елементів управління

Елементи управління – це видимі об'єкти. Вони розташовані:
у Excel: панель інструментів *Елементи управління;*
у VBA: панель інструментів **ToolBox**.

Деякі з них:

- **Кнопка** (**CommandButton**) – для ініціалізації, закінчення або переривання яких-небудь дій;
- **Поле** (**TextBox**) – для відображення інформації, що вводиться користувачем, а також для відображення результатів обчислень;
- **Напис** (**Label**) – для відображення заголовків або короткого пояснювального тексту;
- **Малюнок** (**Image**) – призначений для виведення растрових зображень;
- **Смуга прокрутки (ScrollBar)** – дозволяє встановити числові значення, грунтуючись на положенні повзунка.

Об'єкти можуть бути розташовані:

у Excel: на робочому Листі;

у VBA: у вікні **Форми** (**UserForm).**

Додавання Форми в проект:

- перейти в редактор VB;
- пункт меню **Insert → UserForm.**

Кожен об'єкт має властивості (характеристики), події, на які він реагує і методи (обробники подій).

Способи виклику вікна властивостей:

- у вікні VBA п/м View → Properties;
- на Листі Excel панель інструментів Елементи управління → **властивості;** контекстне меню об'єкту→властивості.

Деякі властивості об'єктів:

- **Name** – ім'я об'єкту (не змінюємо);
- **Caption** – заголовок об'єкту;
- **BackColor** – колір фону об'єкту;
- **ForeColor** – колір заголовка (колір переднього плану);
- **Font** – шрифт.

У VBA зумовлені події і **процедури обробки подій**.

Наприклад, процедура обробки події Click по об'єкту:

```
Private Sub CommandButton1_Click()
' рядки програми, написані користувачем
End Sub
```

У VBA клітина A2 як об'єкт може бути записана двома способами:
Range ("A2") або **Cells(2,1)**, де 2 – номер рядка, 1 – номер стовпця.

Приклад 1.

```
Cells(2,1) = 14.6              'в клітину A2 заноситься число 14.6
Range("A1") = Sin(0.5)         'в клітину A1 заноситься результат обчислення формули
Range("A3") = Range("A1")*Cells(2,1)^2
```

Для оформлення зовнішнього вигляду клітин використовуються властивості **Interior** (інтер'єр), **Font** (шрифт) і вкладені в них властивості :

Color колір; **Pattern** візерунок;
Size розмір; **Bold** жирний;
Italic курсив та ін.

Приклад 2.

```
Range("A6") = "текст"                    'текст, що вводиться, повинен бути у лапках
Range("A6").Interior.Color = vbGreen     'заливка діапазону зеленим кольором
Range("A6").Font.Color = vbBlue          'колір шрифту блакитний
Range("A6").Font.Bold=True               'шрифт жирний
```

Текст, що йде за символом (') ігнорується компілятором і є коментарем.

Типи даних

У VBA є наступні базові типи даних :

Integer цілий;
Single з плаваючою точкою звичайної точності;
Double з плаваючою точкою подвійної точності;
String рядок;
Variant тип, використовуваний за умовчанням;

Dim – оператор оголошення змінних (змінна має бути оголошена до її застосування).

Variant – основний тип даних середовища VBA.

Локальна змінна доступна тільки в тій процедурі, в якій вона оголошена. *Глобальна* змінна доступна на рівні модуля.

Приклад 1. Оголошення простих змінних:

```
Dim a, c              'за умовчанням тип Variant
a = 100.5: c = 0      'декілька операторів в одному рядку відділяються знаком ":".
```

Приклад 2. Змінну типу Variant можна переоб'явити в одновимірний масив, використовуючи функцію **Array**:

```
Dim a
a = Array(10, 20, 30)       'нумерація елементів йде з нуля
Range ("A1")=a(0)           'в клітині A1 відобразиться 10
Range ("A2")=a(2)           'в клітині  A2 відобразиться 30
```

Приклад 3. Оголошення та ініціалізація одновимірного масиву:

```
Dim A(2) 'верхнє значення індексу дорівнює 2
A(0)= 3.6 : A(1)= 4.82
A(2)= – 12.7
```

Приклад 4. Оголошення та ініціалізація двовимірного масиву:

```
Dim B(1,1)
B(0,0)=2:   B(0,1)=4
B(1,0)=1:   B(1,1)=6
```

Математичні функції VBA

Середовище VBA надає користувачеві великий список вбудованих математичні функцій. Деякі з них:

функція	**опис**
Abs	модуль (абсолютна величина)
Atn	арктангенс
Cos	косинус
Exp	експонента, тобто піднесення числа **e** до вказаного степеня
Log	натуральний логарифм
Int	ціла частина числа
Rnd	повертає випадкове число з інтервалу [0;1)
Sin	синус
Sgn	знак числа
Sqr	квадратний корінь
Tan	тангенс
Mod	залишок від ділення

Тема 2. Основні конструкції мови VBA

Операції відношення

При перевірці умов можуть використовуватися операції відношення**:**

= рівно; > більше;

< > нерівно; >= більше або рівно;

< менше; <= менше або рівно.

Логічні операції:

And (логічне множення), Or (логічне складання), Not (логічне заперечення), що повертають логічні значення.

Таблиця істинності

x	y	AND	OR	NOT x
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

1 – істина; 0 – брехня

Приклад. Нехай x(i) – елемент масиву.

Запис у математиці:	*Запис на мові VBA:*
1) $x_i \in [-1, 1]$	x(i)>= - 1 And x(i)< =1
2) $x_i \in (-\infty;-1] \cup [1;\infty)$	x(i)<=-1 **OR x(i)>=1**

Обчислювальний процес, що розгалужується.

Оператор умовного переходу IF

Існує декілька варіантів цього оператора :

а) **IF умова Then оператори**

Якщо умова набуває значення ІСТИНА, то виконуються оператори, що йдуть за **Then,** інакше оператор, що йде за **IF.**

б) **IF умова Then оператори1 Else оператори2**

Якщо умова набуває значення ІСТИНА, то виконуються *оператори1, що йдуть за* **Then, інакше –** *оператори2, що йдуть за* **Else.**

в) блок **IF**

```
IF умова Then
    оператори1
    [Else
    оператори2]
End IF
```

Група **[Else операторі2] може бути відсутня.** Виконується аналогічно випадкам а) і б), використовується для зручності запису.

Приклад. Вичислити значення функції

$$z = \begin{cases} a+b, & при \quad a > b \\ a-b, & при \quad a < b \\ a*b, & при \quad a = b \end{cases}$$

Значення *a i b* знаходяться в клітинах A2 і B2 відповідно.

Процедура на VBA має вигляд:

```
Sub f1()
Dim a, b, z                'оголошення змінних
a = Range("A2") : b = Range("B2")
If a > b Then z = a + b
 If a < b Then z = a - b
 If a = b Then z = a * b
 Range("C1") = z 'результат поміщається в клітину C2
End Sub
```

Введення даних з клавіатури. Функція InputBox

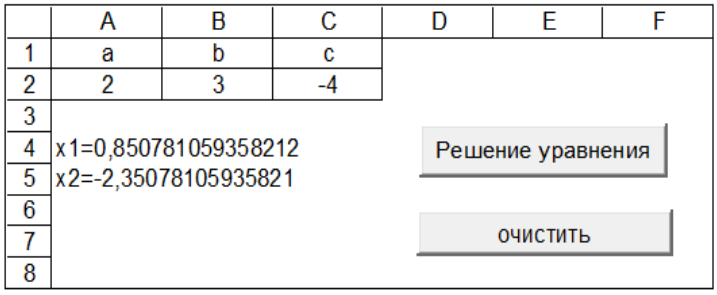
Функція InputBox виводить на екран вікно, що містить повідомлення і поле введення. Повертає значення типу String. Для перетворення текстового типу в числовій використовується функція Val, наприклад:

```
a = val(InputBox("a"))
```

Приклад. Знайти дійсні корені квадратного рівняння $ax^2+bx+c = 0$,. Значення a, b, c вводяться в VBA з клавіатури, результат виводиться в клітини Листа.

Порядок дій :

- у режимі Конструктор розмістити два об'єкти *Кнопка* для запуску проекту і для очищення діапазону, дати їм заголовки *Рішення рівняння* і *Очистити* відповідно.



- обробник події **Click** по кнопці *Рішення рівняння* має вид:

```
Private Sub CommandButton1_Click()
Dim a, b, c, x1, x2, d
a = Val(InputBox("a"))          ' значення змінної a, що вводиться з клавіатури
b = Val(InputBox("b"))
c = Val(InputBox("c"))
Range("A2") = a                 'запис значення змінної в клітину Листа
Range("B2") = b
Range("C2") = c
d = b ^ 2 - 4 * a * c
If d >= 0 Then
                x1=(-b+sqrt(d)/(2 * a)
                x2=(-b-sqrt(d)/ (2 * a)
                Range("A4") = "x 1=" & x1
                Range("A5") = "x 2=" & x2
        Else
                Range("A4") = "дійсних коренів немає"
End If
End Sub
```

- обробник події *Click* по кнопці *Очистити* має вигляд:

```
Private Sub CommandButton2_Click()
Range("A4: A5").ClearContents
End Sub
```

- повернутися в Excel, вийти з режиму конструктора, запустити проект на виконання (RUN або F5). Перевірити роботу проекту клацаннями по кнопках.

Циклічний обчислювальний процес

Оператори циклу FOR … NEXT

Загальний вигляд:

FOR *змінна* = A1 TE A2 [step A3]
    оператори
NEXT [змінна]

де *змінна* – параметр циклу;

A1 – початкове значення параметра циклу;

A2 – кінцеве значення параметра циклу;

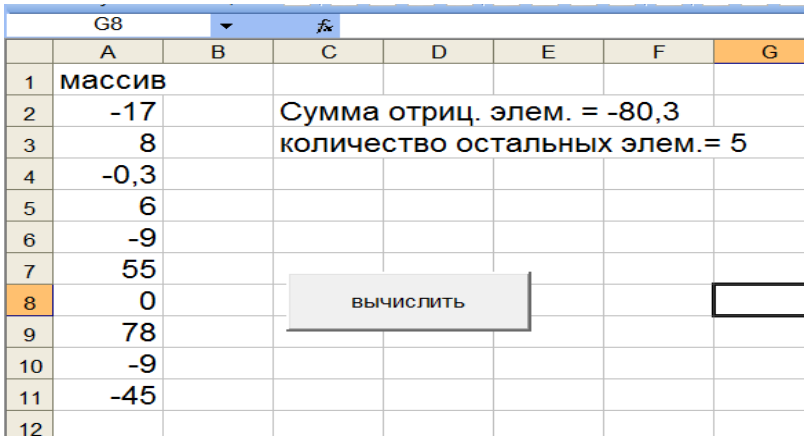
A3 – крок зміни параметра циклу (якщо крок дорівнює 1, то може бути відсутнім).

Одновимірні масиви

Приклад. Даний масив X=(- 17; 8; - 0,3; 6; - 9; 55; 0; 78; -9: - 45). Знайти суму від'ємних елементів та кількість інших елементів масиву.
(Масив ввести у програмі і вивести в клітини Листа).

Порядок дій :

1. у режимі *Конструктор* встановити об'єкт CommandButton1, дати йому заголовок.



2. обробник подій *Click* по кнопці *Вичислити* має вигляд:

```
Private Sub CommandButton1_Click()
Dim x, k, s, i
x = Array(- 17, 8, - 0.3, 6, - 9, 55, 0, 78, - 9, - 45) 'змінна X переоб'являється в масив
s = 0: k = 0
For i = 0 To 9                  'нумерація елементів масиву з 0
Cells(i + 2, 1) = x(i)          'виведення елементів масива в 1-й стовпець
If x(i)< 0 Then
s = s + x(i)                    'накопичення суми
Else
k = k + 1
End If
Next
Range("c2") = "Сума від'ємних елементів = " & s
Range("c3") = "кількість інших елементів = " & k
End Sub
```

Генерація випадкових чисел

Rnd()	**повертає випадкове число з діапазону [0;1);**
100***Rnd()**	**випадкове число з діапазону [0;100);**
а+(в-а)***Rnd()**	**випадкове число з діапазону [а;в);**
Int(100*Rnd())	**повертає ціле випадкове число з діапазону [0;99];**
Randomize	**Ініціалізує генератор випадкових чисел.**

Приклад. Згенерувати масив випадкових чисел з 10 елементів.

```
For 0 = 1 To 9
  x(i) = 100*Rnd() - 50          'випадкові числа з діапазону [- 50;50]
Next
```

Динамічні масиви

Програма буде універсальною, якщо працюватиме для масиву будь-якої розмірності.

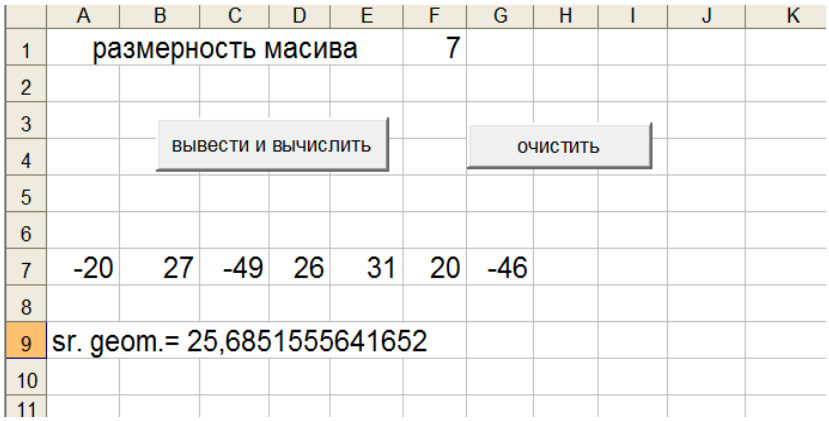
Якщо розмірність масиву в програмі не постійна, то такий масив називається *динамічним*. Динамічні масиви оголошуються за допомогою оператора **ReDim.**

Приклад. Створити процедуру для генерування одновимірного масиву цілих випадкових чисел будь-якої розмірності. Знайти середнє геометричне позитивних елементів цього масиву.

Порядок дій :

- *у режимі Конструктор* встановити елемент управління *CommandButton1*, дати йому заголовок;

Нехай розмірність масиву знаходиться в клітині F7.



- обробник події *Click* по кнопці *Вичислити та вивести* має вигляд:

```
Private Sub CommandButton1_Click()
Dim p, k, n, i, sg
n = Range("F7")
ReDim x(n - 1)
Randomize
For i = 0 To n - 1
  x(i) = int(100 * Rnd() - 50)   'діапазон [- 50;50]
  Cells(7, i + 1) = x(i)
Next
p = 1: k = 0
For i = 0 To n - 1
  If x(i)> 0 Then
                    p = p * x(i) : k = k + 1
  End If
Next
Sg=p^(1/k)
Range("a9") = "sr. geom.= " & sg
End Sub
```

Двовимірні масиви

Приклад 1. Заповнити масив X(3,4) цілими випадковими числами, вивести його в клітини Excel.

```
For i = 0 To 2
For j = 0 To 3
x(i, j) = int(100 * Rnd() - 50)
Cells(i + 1, j + 1) = x(i, j)
Next
Next
```

Приклад 2. Згенерувати масив цілих випадкових чисел розмірністю 3*3. Знайти суму елементів в кожному рядку масиву.

Порядок дій :

- перейти в *режим Конструктора,* встановити елементи управління *CommandButton1* і *CommandButton2,* дати їм заголовки *Вичислити і Очистити* відповідно.
- обробник події *Click по кнопці Вичислити має вигляд:*

```
Private Sub CommandButton1_Click()
Dim i, j, x(2, 2)  ) 'оголошення змінних і масиву розмірністю 3*3
Randomize
For i = 0 To 2
  For j = 0 To 2
       x(i, j) = Int(100 * Rnd() - 50)     'цілі випадкові числа з діапазону [- 50;49]
       Cells(i+2, j+2) = x(i, j)
  Next j
Next i
For i = 0 To 2
  s = 0
  For j = 2 To 2
    s = s + x(i, j)
  Next j
  Cells(i+2, 5) = s
Next i
End Sub
```

- обробник події *Click* по кнопці *Очистити* має вигляд:

```
Private Sub CommandButton2_Click()
Range("b 2: e4").ClearContents
End Sub
```

Тема 3. Середовище Borland C++ Builder

Borland C++ Builder – це середовище програмування, створене компанією Borland, яке дозволяє швидко створювати додатки на мові C++. Це середа, в якій може бути здійснене об'єктно-орієнтоване програмування.

Кожен об'єкт має властивості, події, на які він реагує і методи (обробники подій).

Предком мови C++ явилася мова C, створена в 2-ій половині 1970-х років Денисом Рітчі (США), як мова системного програмування.

У 80-х роках XX ст. Б'ерном Строуструпом створюється мова C++, як мова об'єктно-орієнтованого програмування.

Головні складові частини середовища C++ Builder

Головне вікно (C++ Builder – Project1). Тут знаходяться:

- головне меню;
- панелі інструментів;
- палітра компонентів. Основні візуальні компоненти знаходяться на сторінках Standard, Addition, Win32
- вікно **Дизайнер Форм (Form1)** – це порожня форма, яка автоматично з'являється на екрані при створенні проекту. Тут розташовуються компоненти середовища;
- вікно **Інспектор об'єктів (Object Inspector)** – це вікно, що дозволяє побачити основні властивості і події об'єкту, поміщеного у форму;
- **вікно редактора коду –** тут знаходиться програмний модуль (за умовчанням ім'я Unit1.cpp).

Проект Builder

Усі програми користувача в середовищі Builder оформляються у вигляді *проектів*. Розробка будь-якого проекту починається зі збереження порожнього проекту в новій теці:

- пункт меню **File → Save Project As :**
  треба створити нову папку, в якій зберегти два файли:
  - програмний модуль: за умовчанням ім'я **Unit1.cpp** → зберегти;
  - модуль проекту : за умовчанням ім'я **Project1.bpr** → зберегти.

Проміжні збереження:

пункт меню **File→ Save All** – зберігаються усі початкові файли під поточними іменами.

Папку, що містить файли проекту, можна переміщати, перейменовувати, переносити на інші комп'ютери, але файли, що відносяться безпосередньо до проекту, перейменовувати не можна.

Результатом роботи проекту є здійснимий файл (додаток).

Склад і структура проекту

Середовище Builder зв'язує з кожним своїм застосуванням наступні файли:

Unit1.cpp – модуль форми (програмний модуль). Якщо з'являться нові форми, то для кожної з них буде побудований свій програмний модуль: Unit2.cpp, Unit3.cpp і так далі.

Unit1.h – модуль з оголошеннями компонентів, які розташовані в цій формі. Для кожної нової форми буде побудований свій модуль: Unit2.h, Unit3.h, і так далі.

Unit1.dfm – файл опису форми і усіх розташованих в ній компонентів.

Project1.cpp – файл проекту. Головна програма проекту, що управляє проектом.

Project1.bpr – файл, що відповідає за проведення процесу зборки і компіляції проекту.

Project1.exe – здійснимий файл (додаток), готова до виконання програма, яка може функціонувати під управлінням операційної системи Windows.

Project1.res – файл ресурсів проекту, є двійковий файл. У ньому зберігаються різні значки, графічні зображення, використовувані в проекті.

Тема 4. Елементи мови С++

Символи мови

1. Прописні і рядкові букви латинського алфавіту A, B, C, .X, Y, Z;
   a, b, c,.x, y, z;
   символ підкреслення _
2. Прописні і рядкові букви російського алфавіту
   Однакові прописні і рядкові букви вважаються різними символами.
3. Арабські цифри 0, 1, 2, …,.9.
4. Спеціальні символи . ; ( )< > / * = - % та інші.
5. Символи, що управляють, використовувані у функціях введення-виводу :
   - \n – перехід на новий рядок;
   - \t – горизонтальна табуляція;
   - \o – нульовий символ;
   - \v – вертикальна табуляція та інші.

Константи

Розрізняють чотири види констант :

Цілі константи:

- *десяткові*: наприклад 16; 240;
- *вісімкові:* в якості цифр використовуються символи 0, 1, 2, 3, 4, 5, 6, 7. Вісімкові константи завжди починаються з нуля, наприклад:
  01, 065, 0777;
- *шістнадцяткові:* в якості цифр використовуються 16 символів – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Шістнадцяткові константи завжди починаються з пари символів 0x. Наприклад:
  0x10, 0xFFF, 0x1F1A

Дійсні константи (з плаваючою точкою)

Наприклад: 3.45; 1.5E-2; -1.85E12; -.56.

Символьні константи

Є символом, поміщеним в апострофи. У таблиці кодів ASCII кожному символу ставиться у відповідність ціле позитивне число (код), тому *значенням символьної константи є числовий код символу.*

Приклади символьних констант : 'Q'; 'a'; ''.

Строкові константи – це послідовність символів, поміщена в лапки. Наприклад: "Borland C++".

У кінці строкової константи завжди стоїть ознака кінця рядка '\0', яку формує компілятор: " Borland C++'\0'".

Коментарі

Якщо текст коментарів займає один рядок, то використовується //..*коментар*… Якщо текст коментарів займає більше за один рядок, то використовується

/* */

Типи даних

Особливістю мови С являється відсутність принципу умовчання. Тому типи усіх змінних мають бути оголошені.

Базові типи даних:

int цілий тип;

float тип з плаваючою точкою;

double з плаваючою точкою подвійної точності;

char символьний тип;

void порожній тип.

На основі цих типів будуються інші типи за допомогою модифікаторів:

short короткий;

long довгий;

unsigned без знаку;

bool логічний тип.

Таблиця основних типів даних

ТИП	РОЗМІР(байт)	ЗНАЧЕННЯ
bool	1	true, false
int	2	-32768 – 32768
short int	2	-32768 – 32767
long int	4	-2147483648 – 2147483648
unsigned int	2	0 – 65535
float	4	-1,2e-38 – 34e38
double	8	2,2e-308 – 1.8e308
char	1	256 значень символів

Оголошення та ініціалізація змінних

1) **int** a = 24, i = 5; //змінним цілого типу **a, i** привласнюються значення

2) **char** c='c'; //змінної символьного типу **c** привласнюється значення

Змінна **c** містить один символ **c** (точніше його код по таблиці кодування), тип **char** і **int** взаємозамінні.

3) **String** a, b; // оголошуються змінні строкового типу **a** і **b**

a = "Програмування";

b = "на C++";

Змінна будь-якого типу може бути оголошена як та, що не змінюється. При цьому додається зарезервоване слова **const**.

Наприклад: **const double** A=2.12E-2.

Масиви

Оголошення масивів :

1) **int** a[30]; //масив з 30 елементів цілого типу, нумерація з нуля

Це елементи: a[0], a[1], ...a[29]. Кількість байт в пам'яті 2*30=60 байт.

2) **double** b[2][3]; /*двовимірний масив з 6 елементів з плаваючою точкою подвійної точності */.

Це елементи: b[0][0], b[0][1], b[0][2], b[1][0], b[1][1], b[1]][2]].

Ініціалізація масивів

1) Одновимірний масив

int m[2]={1,8};

2) Двовимірний масив $A = \begin{pmatrix} 2 & 1 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

int a[2][3]={{2, 1, 3}, {4, 5, 6}};

можливі варіанти **int** a[2][3]={2, 1, 3, 4, 5, 6};

int a[ ][3]={2, 1, 3, 4, 5, 6};

3) Масив символів

char str[3]={'a ',' b ',' c'}; //одновимірний масив символів

Арифметичні операції

+	складання	**/**	цілочисельне ділення
-	віднімання	**%**	залишок від ділення
*****	множення	унарні операції	**+** і **-**

Наприклад:	**операція**	**результат**
	11/3	3
	11./3.	3.666….
	11%3	2
	-x*y	(- x)*y, оскільки пріоритет унарних операцій вищий, ніж у бінарних.

Математичні функції

функція	**опис**
fabs	модуль (абсолютна величина)
abs	модуль (для цілих чисел)
sin	синус
cos	косинус
tan	тангенс
exp	експонента, тобто піднесення числа **e** (основа натурального логарифма) до вказаного степеня
log	натуральний логарифм
log10	десятковий логарифм
sqrt	корінь квадратний
pow(x, y)	піднесення до степеня x^y
pow10(x)	піднесення до степеня 10^x
asin	арксинус
acos	арккосинус
atan	арктангенс
fmode(x, y)	залишок від ділення x на y
M_PI	Число π

Усі функції, окрім **abs,** повертають значення типу **double,** типи аргументів повинні бути теж **double**.

Приклади запису арифметичних виразів на мові C++ :

$\frac{e^{2x}+\sin(x-y)^2}{\sqrt{xy}-\ln\frac{x}{2}}$	(exp(2*x)+sin((x-y)*(x-y)))/(sqrt(x*y) - log(x/2.))
$\sin^2 x$	pow(sin(x), 2)
$\sqrt[3]{x}$	pow(x, 1./3.)

Функції в C++

Усі програми на C++ будуються з функцій. Одна частина функцій знаходиться у бібліотеках. Інша − створюється самим користувачем (функції користувача).

Кожна функція перед використанням має бути оголошена. Оголошення бібліотечних функцій знаходяться в заголовних файлах, які підключаються до програми за допомогою директиви **#include** <ім'я файлу.h>.

Оголошення математичних функцій підключаються за допомогою директиви **#include <math.h>.**

Консольний режим. Можливості введення-виводу C++

Консольне застосування – це програма на мові C++ в середовищі Builder, яка запускається без графічного інтерфейсу в консольному вікні.

Введення, що йде з клавіатури:

cin >>

cin – стандартное ім'я потоку введення.

Наприклад:

а) **cin** >> a; // дані вводяться з клавіатури в змінну **a.**

б) **cin** >> i >> j >> s; //Дані вводяться в три змінні i, j, s.

Вивід, що йде на екран:

cout <<

cout – стандартне ім'я потоку виводу.

Наприклад:

а) **cout** <<b; // значення змінної **b** виводиться на екран.

Оголошення функцій *cin, cout* підключаються до проекту за допомогою директиви **#include <iostream.h>.**

Створення консольного додатка. Лінійний обчислювальний процес

Працюючи в консольному режимі, завжди має бути функція з ім'ям **main** (головна), саме з неї починається виконання програми, в якому б місці вона не знаходилася.

Приклад. Розробити проект для обчислення значення функції ***y*** при різних значеннях ***x.***

$y = a \cdot e^x + \ln c$, де

$c = \sqrt{b + s \cdot \sin\left(\frac{\pi}{4}\right)}$,

a = 3,112; b = 5,85; s = 9, 48**;** значення **x** вводяться з клавіатури.

Порядок дій :

1. створити новий проект: пункт меню **File→New;**
2. у вікні, що відкрилося, вибрати **Console Wizard** → OK;
   активізувати перемикач C++, потім **Console Application** → OK;
   На екрані з'явиться вікно **Unit1.cpp** і заготівля для введення функції :

```
int main(int argc, char* argv[])
{
    return 0;
}
```

3. зберегти проект: пункт меню **File → Save Project As:**
   створити нову папку, зберегти 2 модулі:
   - ім'я програмного модуля за умовчанням **Unit1;**
   - ім'я модуля проекту **Project1;**

4. внести зміни в заготівлю, програма на С++ в консольному режимі має вигляд:

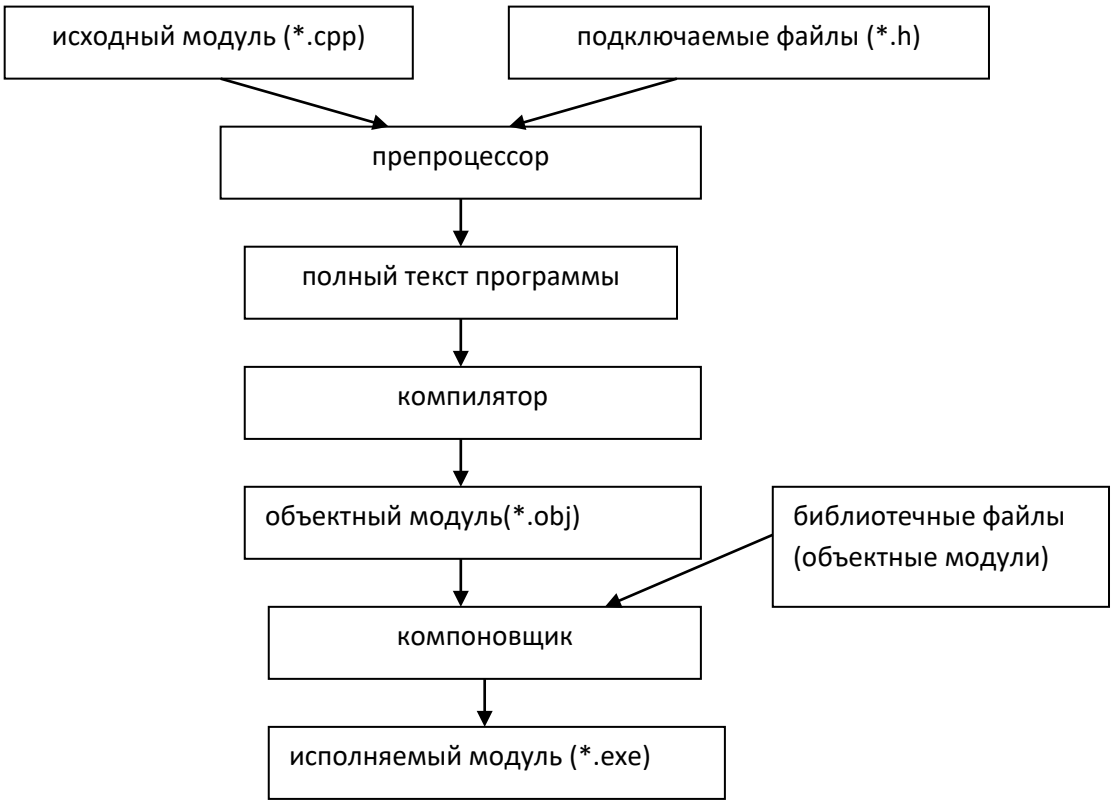
```
#include <math.h>                //для математичних функцій
#include <iostream.h>             //для функцій cin, cout
#include <conio.h>               //для функцій getch(), clrscr ()
//---------------------------------------------------------------------------
main(){
double a=3.112, b=5.85, s=9.48;        //ініціалізація змінних
double  x, y, c;                       //оголошення змінних
clrscr ();                             //функція очищення екрану
cout<<"input x"<<endl;           //вивід на екран і спуск на новий рядок
cin>>x;                                //введення з клавіатури значення x
c= sqrt(b+s*sin(M_PI/4));
y=a*exp(x)+log(c);
cout<<"y="<<y;
getch();
}
```

Функція getch() чекає введення з клавіатури будь-якого символу, роблячи при цьому затримку екрану виводу.

5. запустити проект на виконання RUN [F9]:
   при x=5.5;
   результат y=762,421;

6. зберегти відлагоджену програму **File→ Save All.**

Стадії проходження програми



1. *Початковий модуль* (*.cpp) – програма на алгоритмічній мові. Передається препроцесору, який виконує директиви, що знаходяться в тексті. У текст програми включається вміст вказаних файлів.

#include <ім'я файлу.h> – пошук файлу здійснюється у бібліотеці ОС (бібліотечні файли).

#include "ім'я файлу.h" – пошук файлу здійснюється в поточному каталозі (функції, створені користувачем).

Повний текст програми передається на вхід компілятора.

2. Компілятор виявляє синтаксичні помилки і у разі їх відсутності будує *об'єктний модуль* (*.obj) (це програма в двійкових кодах без синтаксичних помилок).

3. Компонувальник, підключаючи до об'єктного модуля інші об'єктні модулі (бібліотечні файли), формує *здійснимий модуль*.

Здійснимий модуль має розширення *.exe – це готова до виконання програма.

Тема 5. Windows додатки у графічному середовищі. Лінійний обчислювальний процес

Функції приведення типів

StrToInt(рядок) – перетворення рядка в ціле число

StrToFloat(рядок) – перетворення рядка в число з плаваючою точкою

IntToStr(ціле число) – перетворення целого10-го числа в строковий тип

FloatToStr(число) – перетворення значення з плаваючою точкою в строкове представлення

IntToHex(ціле число) – перетворення цілого 10-го числа в 16-не

Операції з рядками

1. Злиття рядків. Вивід здійснюється в об'єкт Label1 :

   Label1 ->Caption="Мова" "C++";

   Виведеться Мова C++

2. Злиття рядка і числового вираження :

   Label2 ->Caption =" Результат=" + IntToStr(4*6);

   Виведеться Результат=24

Введення даних з клавіатури

У графічному режимі введення даних з клавіатури здійснюється за допомогою функції **InputBox,** яка підключається до проекту за допомогою директиви : **#include <Stdlib.h>**

Функція повертає значення типу **String.**

Загальний вигляд:

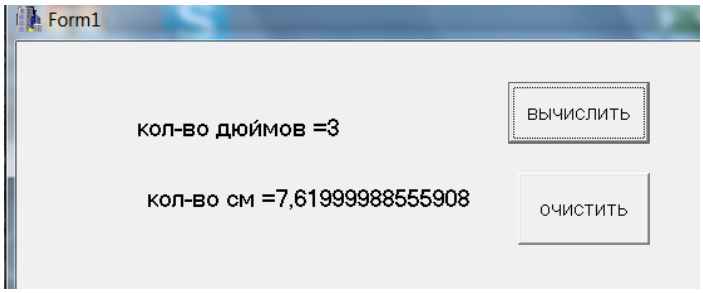
InputBox ("повідомлення", "підказка", "Default")

Default – строкове вираження, використовуване за умовчанням, якщо користувач не введе рядок. У загальному випадку може бути пропуск.

Приклад.1. Створити проект для перекладу кількості дюймів, що вводяться з клавіатури, в сантиметри, 1дюйм = 2,54см.

Порядок дій :

- створити проект, використовуючи пункт меню **File → New Application ;**
- зберегти проект: пункт меню **File → Save Project As;**
- розмістити об'єкти, надати їм необхідні властивості у *вікні властивостей.*



На сторінці **Standard** палітри компонентів :

об'єкт **Label1** – для виведення кількості дюймів;

об'єкт **Label2** – для виведення кількості сантиметрів;

об'єкт **Button1** – для запуску проекту, властивість *Caption* → вичислити ;

об'єкт **Button2** – для очищення полів виводу, властивість *Caption* → очистити;

- обробник події Click по кнопці *вичислити* має вигляд:

```
#include <Stdlib.h>
//=======================================
void __fastcall TForm1::Button1Click(TObject *Sender)
{
float a, inch;
inch=StrToFloat(InputBox("input", "kd"," ")); //введення кількості дюймів
a=inch*2.54;                                  //кількість сантиметрів
Label1 ->Caption="кількість дюймів ="+FloatToStr(inch);
Label2 ->Caption="кількість см ="+FloatToStr(a);
}
```

- обробник події Click по кнопці *очистити* має вигляд:

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  Label1 ->Caption="";
  Label2 ->Caption="";
}
```

- запустити програму на компіляцію і виконання **[F9] (Run)**при inch=3 кількість см=7,61.

Перетворення типів

Якщо у програмі використовуються змінні різних типів, то компілятор автоматично здійснює перетворення.

Правила перетворень

1. У операторові привласнення тип значення правої частини завжди перетвориться в тип значення лівої частини.

Наприклад

int i = 3.14; // 3.14 перетвориться до цілого типу, в результаті i =3.

2. У арифметичному вираженні нижчий тип завжди перетвориться до вищого, наприклад

float → double, int → long int.

3. Будь-який вираз може бути приведений до бажаного типу за допомогою конструкції **(тип) виразу.**

Наприклад: int x=5;
(float) x/2;

Значення виразу буде 2.5

Операції збільшення і зменшення на одиницю

++	збільшити на одиницю;
--	зменшити на одиницю.

Приклад 1.

```
int c;
  ++ c;      // значення c збільшується на одиницю, а потім використовується;
  C++;       // значення C використовується, а потім збільшується на одиницю.
```

Приклад 2. Що буде виведене на екран в результаті роботи фрагменту?

```
c=5;
x=c++;        //x=5, c=6
cout<<x<<c;
```

результат 5 6

Тема 6. Обчислювальний процес, що розгалужується

Операції відношення

= =	рівно	!=	нерівно
>	більше	>=	більше або рівно
<	менше	<=	менше або рівно

Якщо дві змінні порівнюються операцією відношення, то результат завжди буде логічного типу.

Істина (true) – формується значення відмінне від нуля, найчастіше одиниця.

Брехня (false) – формується значення, що дорівнює нулю.

Логічні операції

Логічні операції в С++ відповідають класичним логічним операціям

&&	логічне І
||	логічне АБО
!	логічне НЕ

Таблиця істинності

x	y	x && y	x || y	!x
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

1 – істина

0 – брехня

Пріоритет логічних операцій нижче пріоритету операцій відношення.

Наприклад запис на мові С++ умови $x \in [0;100]$ має вигляд:

```
x >= 0 && x <=100
```

Операція привласнення

Операція привласнення позначається знаком =.

1. Вираз виду i = i + 2 може бути записаний у вигляді i+=2

   i = i*2 еквівалентно i*=2

   i=i/10 еквівалентно i/=10

   i=i%10 еквівалентно i%=10

2. Багатократне привласнення a = b = c = x*y відбувається справа наліво

Спочатку обчислюється **x*y**, потім його значення привласнюється змінній **c**, потім **b**, і лише потім змінній **a**.

Умовна операція

Загальний вигляд **e1? e2:e3** , де e1, e2, e3 – вирази

Якщо **e1** набуває значення істина (тобто ≠ 0), то значенням цієї конструкції буде **e2**, інакше **e3**.

Наприклад, знаходження більшого з 2-х чисел **a** і **b** може бути записано у вигляді: max = (a > b)? a: b;

Складений оператор (блок) – це декілька операторів, які поміщені у фігурні дужки **{ }**, блок еквівалентний одному операторові. Символ "**;**" після блоку не ставиться.

Порожній оператор. Цей оператор використовується там, де по синтаксису мови потрібен оператор, а по сенсу ніякі дії не виконуються.

Умовний оператор IF

Розглянемо два варіанти оператора IF :

а) **if (умова) оператор1;**

Якщо умова набуває значення істина (тобто ≠ 0), то виконується *оператор1*, інакше – наступний оператор програми.

б) **if (умова) оператор1;**
else оператор2;

Якщо умова набуває значення істина (тобто ≠ 0), то виконується *оператор1*, інакше виконується *оператор2*.

Приклад 1. Обчислити значення функції **y** при різних значеннях **x**

$$y = \begin{cases} \cos x, & x \le 0 \\ \arcsin x, & 0 < x \le \dfrac{\pi}{2} \\ \log_4 x, & \dfrac{\pi}{2} < x \le 64 \\ \dfrac{1}{x^2}, & x > 64 \end{cases}$$

Програма в консольному режимі має вигляд:

```
#include<math.h>
#include<conio.h>
#include<iostream.h>
//---------------------------------------------------------------------------
void main() {
double x, y;
  clrscr();
  cin>>x;
  if (x <= 0) y = cos(x);
  if (x > 0 && x <= M_PI/2)   y = asin(x);
  if (x > M_PI/2 && x <= 64)  y=log(x)/log(4);
  if (x > 64)   y=1/pow(x, 2);
  cout << "y=" << y;
 getch();
}
```

Запустити програму на компіляцію і виконання

при x= - 3 отримаємо результат y= -.989972

при x = 1 отримаємо результат y= 1.5702

Приклад 2. Створити в графічному середовищі Builder проект для обчислення значення функції **y** при різних значеннях **x**

$$y = \begin{cases} a + b - x, & x \ge 1{,}5 \\ \sin^2 x + \cos(a + b), & x < 1{,}5 \end{cases}$$

$a = 1{,}3 - x^2$; b = 0,85

Порядок дій :

- створити новий проект, зберегти його;
- розмістити об'єкти, задати їм необхідні властивості;

  на сторінці Standard палітри компонентів :

  об'єкт **Label1**, властивість *Caption → значення аргументу;*

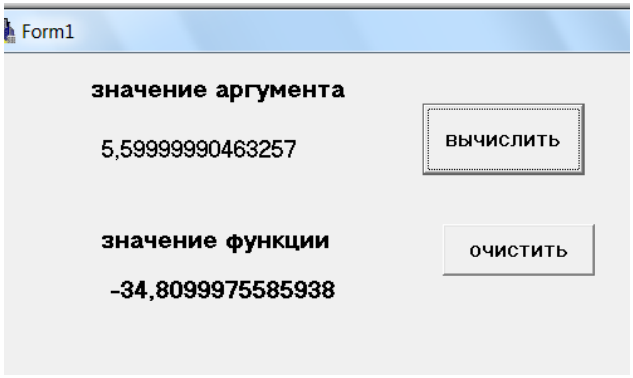
  об'єкт **Label2** для виведення значень **x;**

  об'єкт **Label3**, властивість *Caption → значення функції;*

  об'єкт **Label4**, для виведення значень **y;**

  об'єкт **Button1** для запуску проекту, властивість *Caption → вичислити;*

  об'єкт **Button2** для очищення полів, властивість *Caption → очистити.*



- обробник події Click на об'єкті *вичислити*

```
#include <math.h>
#include <stdlib.h>
//---------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{float b=0.85;
    float x, y, a;
    x=StrToFloat(InputBox("vvedite", "x",""));
    a=1.3 - x*x;
    if(x>=1.5) y=a+b - x;
    else
    y=sin(x)*sin(x)+cos(a+b);
    Label2 ->Caption=x;
    Label4 ->Caption=y;
}
```

- обробник події Click на об'єкті *очистити*

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 Label2 ->Caption="";
 Label4 ->Caption="";
}
```

Оператор вибору switch
(оператор множинних розгалужень)

Синтаксис:

switch (вираз){
case const1: послідовність операторів; **break;**
case const2: послідовність операторів; **break;**
……..
case constN: послідовність операторів; **break;**
[**default**: послідовність операторів; **break;**]
}

Вирази const1, const2, … – мають бути цілого типу.

Правила виконання оператора :

- обчислюється вираження в заголовку і порівнюється послідовно з const1, const2, …, constN. Як тільки буде знайдено відповідність, виконаються оператори, що йдуть за двокрапкою;
- якщо відповідність ніде не встановлена, то виконаються оператори, що йдуть за ключовим словом **default**.

break – вихід з оператора *switch* і перехід до наступного оператора програми.

Приклад 3. Програма, що забезпечує виведення назв днів тижня по їх номеру.
Консольний режим:

```
#include <conio.h>
#include <iostream.h>
//---------------------------------------------------------------------------
void main() {
int dn;              //dn – номер дня тижня
  cout<<"input dn";
  cin>> dn;      //у змінну dn  вводиться номер дня тижня
```

```
    switch (dn){
      case   1 : cout<<"Monday"; break;
      case   2 : cout<<" Tuesday"; break;
      case   3 : cout<<" Wednesday"; break;
      case   4  : cout<<" Thursday"; break;
      case    5 : cout<<" Friday"; break;
      case    6 : cout<<" Satyrday"; break;
      case    7 : cout<<" Sunday "; break;
      default: cout<<" input integer number   1-7 " ; break;
    }
     getch();
}
```

Тема 7. Циклічний обчислювальний процес

Існує три види взаємозамінних операторів циклу : **for, while, do … while**

Оператор циклу FOR

Загальний вигляд оператора :

for (вираз1; вираз2; вираз3) тіло циклу;

вираз1 – задає початкове значення параметру циклу;

вираз2 – задає умову продовження циклу ;

вираз3 – задає зміну параметра циклу;

тіло циклу – може бути або простій, або складений оператор.

Алгоритм роботи циклу:

1. привласнюється значення параметру циклу;
2. перевіряється умова продовження циклу. Якщо умова набуває значення *істина* (≠0), то виконується тіло циклу і перехід на п.1, інакше виконується оператор, що йде за оператором **for.**

Таким чином, перед кожним повторенням циклу відбувається зміна параметра циклу і перевірка умови його завершення.

Приклад 1. `for ( i= 0; i < 10; i ++)  cout << i << "\n";`

В результаті роботи циклу у стовпець виведуться цифри від 0 до 9.

Приклад 2. for (i = 9; i >= 0; i --) cout << i << "\n";

В даному випадку в стовпець виведуться цифри від 9 до 0.

Приклади нескінченних циклів (можуть бути отримані випадково):

for (i = 1; 1; i ++) оператор;

for (i=10; i>6; i++) оператор;

Оператор ***for*** зручний при організації циклів, коли кількість повторень відома.

Приклад 3. Вичислити значення функції

$$z = \begin{cases} ax^2 + \lambda, & x > \lambda \\ ay + x, & x \le \lambda \end{cases}$$

$$x = 3\lambda + \cos\lambda$$

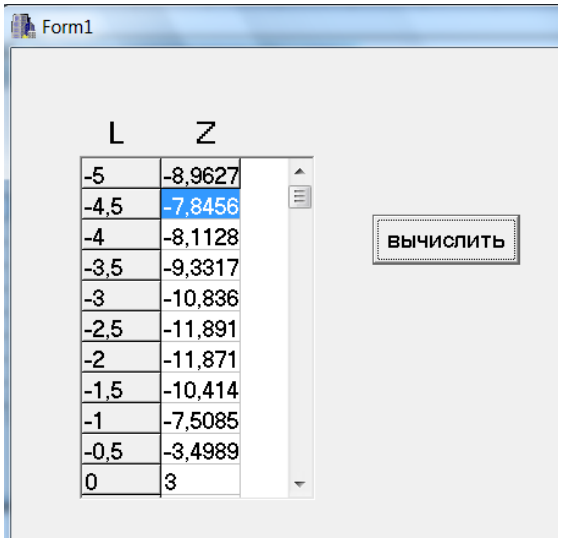
$$y = 2\sin\lambda, \quad \alpha = 3{,}0; \quad \lambda \in [-5;5]; \quad \Delta\lambda = 0{,}5$$

Результат вивести у вигляді таблиці

L	**Z**
.....	.....

Порядок дій (графічний режим):

- створити і зберегти проект;
- для виведення результатів у вигляді таблиці необхідно встановити на формі компонент **StringGrid** (сторінка **Additional**). Визначити кількість рядків в таблиці за формулою $\left[\frac{\alpha_к - \alpha_н}{\Delta\alpha}\right] + 1 = 21$;
- задати властивості компонента *StringGrid :*
  ColCount → 2 (кількість стовпців);
  RowCount → 21 (кількість рядків);
- встановити у форму два компоненти Label і компонент **Button**, дати їм заголовки відповідно (див. рисунок.).



Обробник події Click по кнопці *вичислити* має вигляд:

```
  #include<math.h>
void __fast call TForm1::Button1Click(TObject *Sender)
{
  float x, y, ln=-5, lk=5, dl=0.5, z, a=3.0;              //оголошення змінних
  int i = 0;                          //завдання початкового номера рядка
  for (l = ln; l <= lk; l += dl)  {
    x=3*l+cos(l);
    y=2*sin(l);
    if (x > l) z = a*x*x+l;
    else
    z = a*y+x;
    StringGrid1 ->Cells[0][i]=l;          //Cells[Col][Row]
    StringGrid1 ->Cells[1][i]=z;                //виведення результатів в таблицю
    i++;                                        // зміна номера рядка в таблиці
  }
}
```

Усі результати обчислень можуть бути проглянуті за допомогою вертикальної лінійки компонента StringGrid.

Зона дії змінних

Змінна має бути оголошена до її використання. Існують змінні локальні (внутрішні) і глобальні.

Локальна змінна існує тільки в тому блоці, в якому оголошена. При виході з блоку ця змінна і її значення втрачаються.

Зона дії локальної змінної – блок.

Наприклад:

```
for ( int i=0; i<10; i++)
{тіло циклу}
  i=0;
```

Перша змінна *i* відома тільки в циклі *for*, а у вираженні ***i=0;*** це буде вже інша змінна, тобто їй компілятор присвоїть зовсім іншу адресу.

Глобальна змінна – це змінна, оголошена поза якою-небудь функцією і може бути використана у будь-якому місці програми.

Зона дії глобальної змінної – уся програма.

Наприклад:
```
int a;
int main() {
}
```

Обчислення суми ряду

Приклад. Вичислити суму ряду

$$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots+\frac{1}{49}-\frac{1}{50}$$

Програма в консольному режимі має вигляд:

```
#include<iostream.h>
#include<conio.h>
//---------------------------------------------------------------------------
void main() {
double s=0;
   for (int i=1; i<=50; i++)
   s+=pow(- 1, i+1) / i;
   cout<<"s="<<s;
   getch();
}
```

Результат S=0,683247.

Оператор циклу WHILE

Загальний вигляд:

while (умова продовження циклу) тіло циклу;

Тіло циклу – простий оператор або складений.

Цикл виконується до тих пір, поки умова набуває значення *істина* (≠0). Якщо умова проймає значення *брехня*, то управління передається наступному операторові програми.

Так само як і в циклі *FOR* спочатку перевіряється умова, а потім виконується тіло циклу. Це цикли з передумовою.

Приклад. Для заданого рядка (прізвище користувача, задане латинськими буквами) відобразити ASC коди кожного символу.

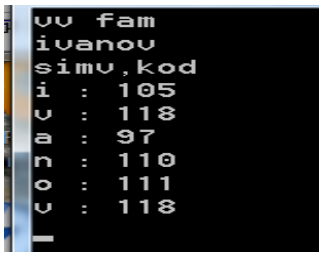
У С++ рядок − це масив символів. Ознака кінця рядка '\0'. Нульовий байт автоматично додається компілятором.

Алгоритм рішення задачі. Консольний режим. name – прізвище.

```
#include<iostream.h>
#include<conio.h>
void main() {
char name [10];                  //довжина масиву з урахуванням '\0'
        cout << "input fam"<<endl;
        cin >> name;
        cout << " simv, kod"<<"";
         int i=0;                //початкове значення параметра циклу
         while (name[i]!='') {
              cout <<name[i]<<":"<< int(name[i])<<"\n";
              i++;          //зміна параметра циклу
          }
         getch();
}
```

Цикл працюватиме до тих пір, поки не зустрінеться ознака кінця рядка "\0".

Результат роботи:



Оператор циклу DO … WHILE

Загальний вигляд:

do

тіло циклу

while (умова продовження циклу **);**

Це цикл з постумовою, тобто спочатку виконується тіло циклу, а потім оцінюється умова продовження циклу. Якщо в результаті оцінки виходить значення *істина*, то цикл повторюється. Якщо отримано значення *брехня*, то цикл завершується.

Таким чином, цикл виконається хоч би один раз.

Приклад . Вичислити суму членів безкінечного ряду

$$e^x = 1 + \frac{x}{1} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots + = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

з точністю до **ε,** де ε- задана скільки завгодно мала величина.

Обчислення проводити до тих пір, поки не виконається умова $\frac{x^n}{n!} \le \varepsilon$.

Алгоритм рішення в консольному режимі:

```
#include<conio.h>
#include<iostream.h>
#include<math.h>
int main() {
  float x, y, s=1, eps, p=1;
  int n=1;
  cout <<"\n vv x i eps\n" ;
  cin>>x>>eps;
   do   {
      p=p*n;
      y=pow(x, n)/p;
      s+=y;
      n+=1;
 }
  while(y>=eps);
  cout<<"summa="<<s<<endl;
  cout<<"element rayda="<<y;
   getch(); }
```

Результати обчислень :

```
 vv x i eps
2.5
0.001
summa=12.1823
element rayda=0.000597289
```

Тема 8. Одновимірні масиви

Генерація випадкових чисел

Функція **rand()** генерує ціле число в діапазоні між **0** і символьною константою **RAND_MAX,** визначеною в заголовному файлі **<stdlib.h>**. Для того, щоб вибрати цілі числа у конкретному діапазоні, використовується операція обчислення залишку **%.**

Наприклад:

rand() %n	цілі випадкові числа з [0; n - 1];
a+rand()%(b - a+1)	цілі випадкові числа з [a; b];
(double) rand()/RAND_MAX	дійсні випадкові числа з [0; 1];

Спільно з функцією **rand** використовується бібліотечна функція **srand**, яка ініціалізувала генератор випадкових чисел від таймера.

srand (time (NULL));

Функція **time** повертає поточний час в секундах.

В результаті, при кожному запуску програми генеруються різні числа.

Одновимірні масиви

Приклад 1. Введення масиву A(10) з клавіатури і вивід його на екран:

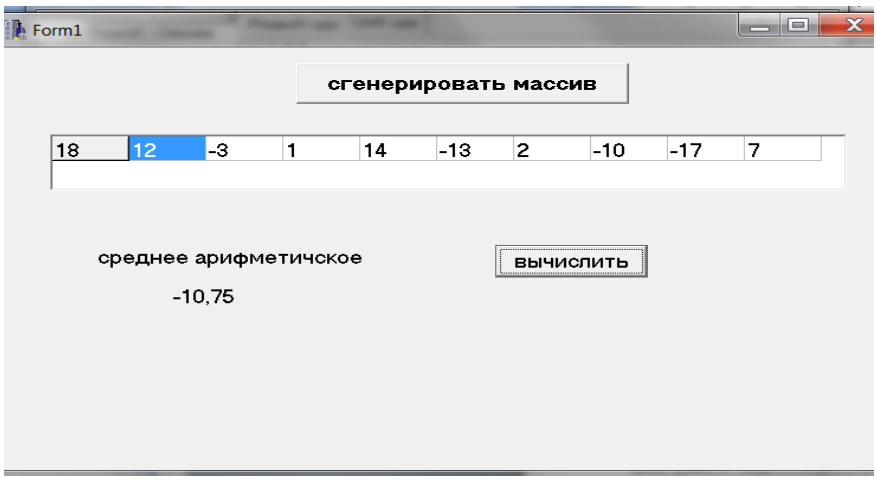
```
for (i=0; i<10; i++) cin>>a[i];
for (i=0;I <10; i++) cout << a[i];
```

Приклад 2. Згенерувати одновимірний масив цілих випадкових чисел розмірністю 7 в діапазоні [- 10;10]. Графічний режим:

```
srand(time(NULL));
for (int i=0; i<7; i++){
    x[i]=-10 +rand()%21;     //заповнення масиву випадковими числами
    StringGrid1 ->Cells[i][0] =x[i];   //виведення масиву в рядок
 }
```

Приклад 3. Заповнити одновимірний масив розмірністю 10 цілими випадковими числами з діапазону [- 20;20]. Знайти середнє арифметичне від'ємних елементів масиву.

Реалізація алгоритму в графічному середовищі. Порядок дій :



- розмістити об'єкти, задати їм необхідні властивості:

✓ об'єкт **Label1**, властивість *Caption* → одновимірний *масив;*

✓ об'єкт **StringGrid1** (сторінка **Addition**);

властивості **ColCount** → 10 (кількість стовпців);

RowCount → 1 (кількість рядків);

✓ об'єкт **Button1** для генерації масиву, властивість *Caption* → *згенерувати масив;*

✓ об'єкт **Button2** для запуску проекту, властивість *Caption* → *вичислити;*

✓ об'єкт **Label2,** властивість *Caption* → середнє арифметичне;

✓ об'єкт **Label3** для виведення результату.

- обробник події Click по кнопці *згенерувати масив* має вигляд:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  srand(time(NULL));
  for (int i=0;i<=9;i++){
  x[i]=-20 +rand()%41;                  //заповнення масиву випадковими числами
  StringGrid1 ->Cells[i][0] =x[i]; //виведення масиву
} }
```

- обробник події Click по кнопці *вичислити* має вигляд:

```
TForm1 *Form1;
int x[10];        //глобальний параметр
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
   double sa;
   int i, sum=0, k=0;
   for (int i=0;i<=9;i++)
   if(x[i]<0){
     sum+=x[i];
     k++;
   }
   sa=(double) sum/k;          //знаходження середнього арифметичного
   Label3 ->Caption=sa;
}
```

Динамічний розподіл пам'яті

Якщо розмірність масиву задається під час виконання програми, наприклад введена з клавіатури, то такі масиви називаються динамічними. Пам'ять під них виділяється за допомогою оператора new.

Наприклад, нехай

arr – ім'я масиву;

*arr – адреса масиву;

n – розмірність масиву (кількість елементів)

тоді у консольному режимі:

```
cin>>n; //ввод розмірності масиву
int *arr=new int[n];
```

В даному випадку, за адресою масиву з ім'ям ***arr*** розмірністю **n**, оператором **new** буде виділена безперервна область пам'яті. Пам'яті виділиться стільки, скільки необхідно для зберігання *n* величин типу *int.*

Приклад. Створити проект для генерації масиву цілих чисел будь-якої розмірності. Знайти суму елементів з номерами кратними 3 і суму інших елементів.

Порядок дій :

- встановити:

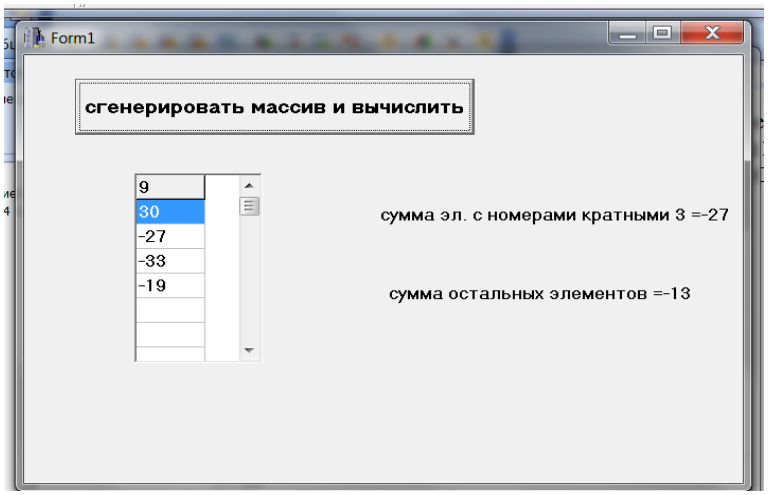
  об'єкти Label1, Label2 для виведення результатів;

  об'єкт Button1 для запуску проекту, властивість *Caption→вичислити;*

об'єкт StringGrid1 для виведення елементів масиву:

ColCount → 1 (кількість стовпців);

RowCount → (кількість рядків);



- обробник події Click по кнопці *згенерувати масив і вичислити* має вигляд:

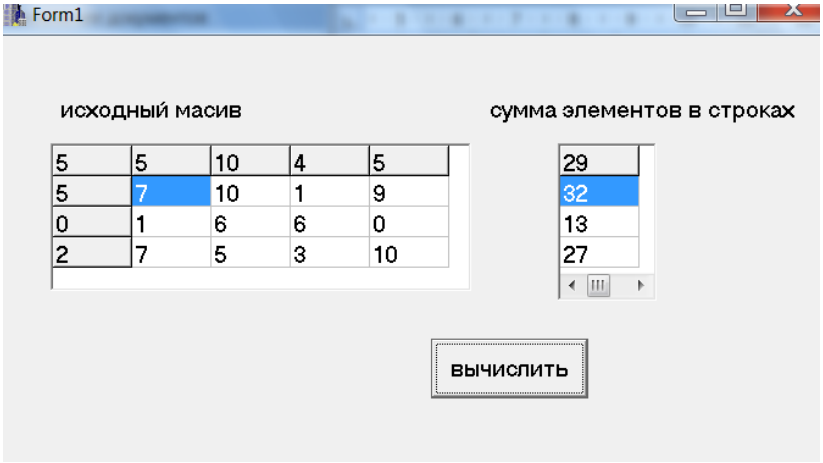
```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  int s1=0, s2=0;
  int size;                  //розмірність масиву
  size=StrToFloat(InputBox("vv", "size",""));       //введення розмірності масиву
  int*arr=new int[size];                  //динамічне виділення пам'яті
  srand (time(NULL));
  for ( int i=0;i<size;i++){
    arr[i]=rand()%100-50;                 //заповнення масиву випадковими числами
    StringGrid1 ->Cells[0][i]=arr[i]; //виведення масиву
  }
    for (int i=0; i<size ;i++)
    if((i+1) %3==0 )
    s1=s1+arr[i];
    else
    s2=s2+arr[i];
    Label1 ->Caption="сума елем. з ном. кратними   3 ="+FloatToStr(s1);
    Label2 ->Caption="сума інших елементів="+FloatToStr(s2);
}
```

- запустити проект на виконання. Ввести з клавіатури розмірність масиву, наприклад 5. Результат дивись на Формі.

Тема 9. Двовимірні масиви

Приклад. Заповнити двовимірний масив A(4,5) випадковими цілими числами з діапазону [- 10;10]. Знайти суму елементів в кожному рядку.
Створити проект в графічному середовищі.

Порядок дій :
Дизайн:



- розмістити об'єкти, задати їм необхідні властивості;
  - ✓ об'єкт **Label1,** властивість *Caption* → початковий *масив;*
  - ✓ об'єкт **Label2,** властивість *Caption* → сума *елементів в рядках;*
  - ✓ об'єкт **StringGrid1** для виведення елементів масиву виді таблиці: : властивості **ColCount** → 5 (кількість стовпців);
    RowCount → 4 (кількість рядків);
  - ✓ об'єкт **StringGrid2** для виведення результатів :
    властивості **ColCount** → 1 ;
    RowCount → 4 ;
  - ✓ об'єкт **Button1** для запуску проекту, властивість *Caption*→ *вичислити.*

- обробник події Click по кнопці *вичислити* має вигляд:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{  const int nrow=4, ncol=5;          //розмірність масиву
   int a[nrow][ncol];
   int s;
   srand(time(NULL));                 //ініціалізація генератора випадкових чисел
   for (int i=0; i<nrow; i++)
```

```
    for (int j=0; j<ncol; j++)  {
    a[i][j]=rand()%21-10;                //заповнення масиву випадковими числами
    StringGrid1 ->Cells[j][i]=a[i][j]; //заповнення таблиці випадковими числами
   }
   for (int i=0; i<nrow; i++) {
     s=0;
     for (int j=0;j<ncol; j++)
     s+=a[i][j];                          //накопичення суми в рядку
     StringGrid2 ->Cells[0][i]=s;        //виведення результатів в таблицю
    }
}
```

- запустити на виконання RUN

У даному коді розмірність масиву задається іменованими константами, що дозволяє їх легко змінювати.

Тема 10. Функції користувача в С++

З використанням функції пов'язано три поняття:

- опис функції – опис дій, які виконує функція;
- оголошення функції;
- звернення до функції.

Опис функції

[тип] ім'я функції ([список формальних параметрів]){
тіло функції
}

Перший рядок **– заголовок функції**.

Тип – це тип повертаного значення. Якщо тип не заданий, то за умовчанням мається на увазі *int*. Якщо функція не повертає значення, то тип *void*.

Формальні параметри – це змінні, використовувані усередині тіла функції. Вони набувають значень при виклику функції шляхом копіювання в них значень відповідних фактичних параметрів.

Для кожного формального параметра має бути вказаний тип.

Оператор **return [вираження];**

Цей оператор забезпечує:

- вихід з функції і повернення результату обчислень у точку виклику.

У тілі функції може бути декілька операторів *return* або жодного.

Приклад. Функція користувача для знаходження більшого з 2-х цілих чисел. Можливі варіанти:

а)
```
max (int a, int b)  {
  if (a>b) return a;
  else return b;
 }
```

б)
```
max (int a, int b){
  return (a>b)? a: b;
 }
```

Оголошення функції

До першого виклику функції *необхідно повідомити тип повертаного результату, а так само кількість і типи аргументів* (це необхідно для виділення місця в пам'яті).

Оголошення функції співпадає із заголовком функції :

[тип] ім'я функції ([список формальних параметрів]);

Оголошення функції може мати вигляд

max (int a, int b); //співпадає із заголовком в описі функції

чи

max (int, int); /* імена формальних параметрів не грають ніякій ролі і ігноруються компилятором*/

Звернення до функції:

ім'я функції ([список фактичних параметрів])

Приклад. Створити функцію, що повертає суму елементів масиву. Скористатися нею при обчислення сум елементів масивів X і Y, що містять 10 і 3 елементи відповідно.

Створити проект в графічному середовищі.

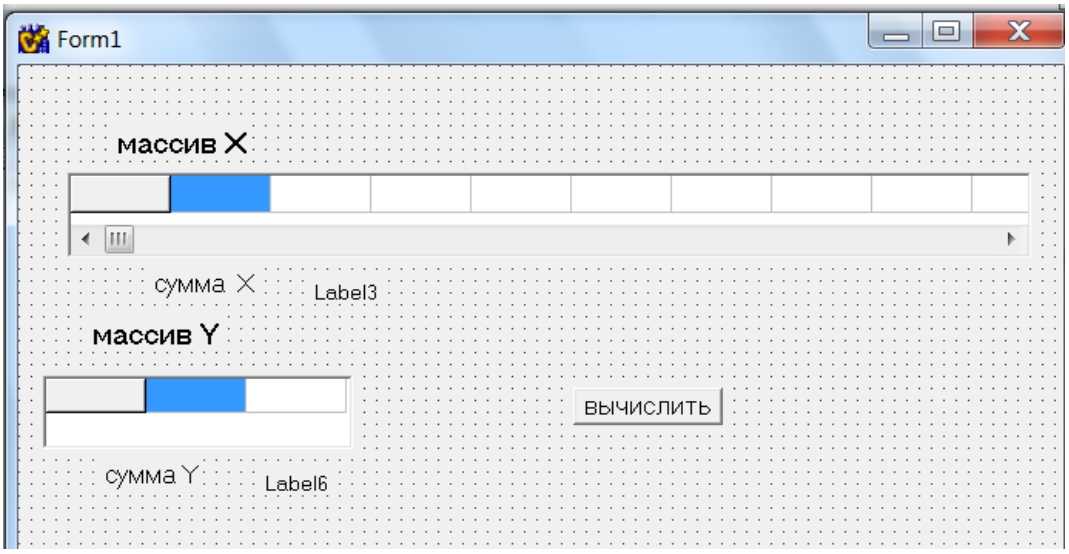
Опис функції :

```
float sum(float a[], int n){
float s=0;
  for (int i=0; i<n;i++)
```

```
    s+=a[i];
    return s;
  }
```

Порядок дій :

- розмістити об'єкти, задати їм необхідні властивості;



- обробник події Click по кнопці *вичислити* має вигляд:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   float sum(float a[], int n);                    //оголошення функції
   float x[10]={7.8, 6, 9, 0, 5.5, 6.6, 7, 8, 9, 10};
   float y[3]={1.1, 2.2, 3.3};
   for (int i=0; i<=9; i++)
   StringGrid1 ->Cells[i][0]=x[i];                 //виведення масиву x
   Label3 ->Caption=sum(x, 10);      //звернення до функції, виведення результату
   for (int i=0; i<=2;i++)
   StringGrid2 ->Cells[i][0]=y[i];                 //виведення масиву y
   Label6 ->Caption=sum(y, 3);
}
float sum(float a[], int n){                       //опис функції
float s=0;
  for (int i=0; i<n;i++)
  s+=a[i];
  return s;
}
```

Рекурсивні функції

У мові С++ функції можуть викликати самі себе. Функція називається рекурсивною, якщо оператор в тілі функції містить виклик цієї ж функції.

Приклад1. Класичний приклад рекурсивної функції – обчислення факторіалу числа. Вичислити n! при n∈[0; 10].

n!=1*2*3*.*n.

Нехай ця функція називається **factorial**.

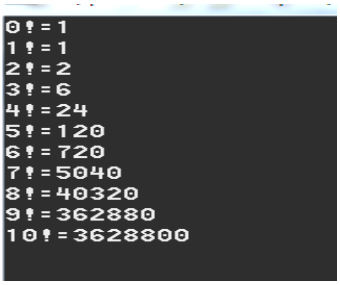
Опис функції :

```
long int factorial(int n){
  if (n==0 | n==1) return 1;
   return factorial (n - 1)*n;
}
```

Програма на С++ в консольному режимі має вигляд:

```
#include<iostream.h>
#include<conio.h>
//……..
long int factorial (int n);                    //це оголошення функції
void main() {                                  //це головна функція
  for (int i=0; i<=10; i++)
  cout<<i<<"!="<<factorial(i)<<""; //звернення до функції
  getch();
}
long int factorial(int n){                  //це опис функції
  if (n==0 | n==1) return 1;
   return factorial(n - 1)*n;
}
```

Результат роботи програми:



Тема 11. Операції над двійковим кодом

Порозрядне (побітове) складання

У обчислювальній техніці уся інформація (числа, текст, графічні дані, звук) кодується двійковим кодом у вигляді {0; 1}. Коди чисел від 1 до 16 в 10-ій, 2-ій, 16-ій системах числення представлені в таблиці.

Decimal	Binary	Hex
1	0000 0001	01
2	0000 0010	02
3	0000 0011	03
4	0000 0100	04
5	0000 0101	05
6	0000 0110	06
7	0000 0111	07
8	0000 1000	08

Decimal	Binary	Hex
9	0000 1001	09
10	0000 1010	0A
11	0000 1011	0B
12	0000 1100	0C
13	0000 1101	0D
14	0000 1110	0E
15	0000 1111	0F
16	0001 1111	10

Порозрядні логічні операції

Порозрядні логічні операції виконуються над відповідними бітами цілих чисел. Кожен біт має значення 0 або 1.

Порозрядними логічними операціями є:

& логічне множення (И);

| логічне складання (ЧИ);

^ що виключає АБО;

~ заперечення;

<< зрушення вліво;

>> зрушення управо.

Порозрядні логічні операції дозволяють забезпечити доступ до кожного біта інформації.

Таблиця істинності

значення бітів		результат операції			
e1	**e2**	**e1 & e2**	**e1 | e2**	**e1 ^e2**	**~e1**
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

1 – істина;

0 – брехня.

Приклад 1. Логічне множення (&).

Результат набуває значення 1, якщо обидва біти мають значення 1.

У 2-ій системі числення 1100 0001

0100 0010

0100 0000

З прикладів видно, що якщо необхідно встановити значення розряду рівне нулю, то користуються операцією &.

Приклад 2. Логічне складання (|).

Результат набуває значення 1, якщо хоч би один з бітів має значення 1.

У 2-ій системі числення 1100 0001

0100 0010

1100 0011

Якщо необхідно встановити значення розряду,що дорівнює 1, то користуються операцією | .

Приклад 3. Що виключає АБО (^)

Результат набуває значення 1, якщо значення тільки одного з бітів дорівнює 1 (один і тільки один).

У 2-ій системі числення 1100 0001

0100 0010

1000 0011

Приклад 4. Логічне заперечення (~).

У 2-ій системі числення : ~0100 0001 → 1011 1110

Таблиця пріоритетів операцій мови С++

операція	призначення
++ --	збільшення (зменшення) на 1
~	побітове НЕ
!	логічне НЕ
+ -	унарний +, унарний -
( )	приведення типу
* / %	множення, ділення, обчислення залишку
+ -	складання, віднімання
>> <<	зрушення
< <= > >= == !=	операції відношення
&	побітове І
^	що побітове виключає АБО
|	побітове АБО
&&	логічне І
||	логічне АБО
?:	умовна операція
= += *=	привласнення

Література

1. Самоучитель VBA. Гарнаев А. Ю. Издание 2, перераб. и доп. – СПб.: БХВ–Петербург, 2004. – 560 с.
2. Швачич Г. Г., Овсянніков О. В. та ін. Інформатика та комп'ютерна техніка. Елементи об'єктно-орієнтованого програмування. Розділ «Реалізація концепції об'єктно-орієнтованого програмування в мові Visual Basic for Application»: Навчальний посібник. – Дніпропетровськ: НметАУ, 2006. – 52 с.
3. Березин Б. И., Березина С. Б. Начальный курс С и С++. – М.: Диалог – МИФИ, 2001. – 288 с.
4. Павловская Т. А. С/С++. Программирование на языке высокого уровня. Підручник для внз. – СПб.: Питер, 2003. – 400 с.
5. Пахомов Б. И. С/С++ и Borland C++ Builder для студентов. – СПб.: БХВ – Петербург, 2006.– 448 с.

Зміст

Навчальне видання

Олександр Викторович Соболенко
Олена Анатоліївна Гуляєва
Ганна Анатоліївна Павленко

ОСНОВИ ПРОГРАМУВАННЯ

Конспект лекцій

Тем. план 2017, поз. 237

Підписано до друку 05.04.2017. Формат 60х84 1/16. Папір друк. Друк плоский. Облік.-вид. арк. 3,0. Умов. друк. арк. 2,95. Тираж 100 пр. Замовлення № 58.

Національна металургійна академія України
49600, Дніпро - 5, пр. Гагаріна, 4

Редакційно-видавничий відділ НМетАУ