**Лекція № 7. Структура програми. Операції порівняння, логічні операції.**

**Виведення значень виразів. Блок. Область дії оголошення імені.**

*Контрольні запитання*

1. Для чого слугують заголовні файли?
2. Як компілятор шукає розташування заголовних файлів?
3. Які маніпулятори використовуються для завдання довжини поля, дробової частини числа?
4. Який заголовний файл використовується для маніпуляторів завдання довжини поля, дробової частини числа?

**Структура програми**

Основними частинами типової структури програми на С++ є такі:

* директиви препроцесорної обробки;
* опис зовнішніх змінних (вихідних даних і результатів) та функцій;
* функції програми;
* головна функція — програми **main().**

Порядок дій в головній функції повинен відповідати логіці дій. Наприклад для проведених ЛР, структура програми повинна мати вигляд:

1. Оголошення констант та змінних.
2. Введення даних, необхідних для проведення розрахунку.
3. Проведення розрахунку.
4. Виведення результатів розрахунку.

Для програми ЛР, з урахуванням її специфіки, загальний вигляд може бути таким.

**#include <iostream> *// підключення потокового введення-виведення***

**#include <iomanip.h> //для маніпуляторів setw, setprecision**

**#define \_USE\_MATH\_DEFINES // для використання констант С++**

**#include <cmath> // для використання математичних формул**

**#include <Windows.h>// для зміни кольору консолі**

**using namespace std; *// простір імен***

**int main() *// головна функція***

**{const float a =1, b=1; // константи**

**double x, // змінна, що вводиться+**

**y1, y2, y3, y4, y5, y6; // змінні з результатами**

**/\*Отримання дескриптора для зміни кольорів консолі**

**Колір фону білий, колір тексту чорний\*/**

**HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);**

**system("color F0");**

**cout<< "\nx="; cin >> x; // введення даних**

**y1 = a/x + sqrt(x\*x +1);**

**y2 = a/x + sqrt(x\*x +1); // замінити на потрібну формулу**

**y3 = a/x + sqrt(x\*x +1); // замінити на потрібну формулу**

**y4 = a/x + сos(M\_PI\*3/4); // замінити на потрібну формулу**

**y5 = a/x + pow (M\_E,x); // замінити на потрібну формулу**

**y6 = a/x + sqrt(x\*x +1); // замінити на потрібну формулу**

***// виведення повідомлення на екран***

**cout<<"\nResults"<<endl;**

**cout<<"\t 1 \t 2 \t 3 \t 4 \t 5 \t 6 \n";**

**cout<< setw(7) << setprecision(2)<< y1<<"\t";**

**cout<< setw(7) << setprecision(2)<< y2<<"\t";**

**cout<< setw(7) << setprecision(2)<< y3<<"\t";**

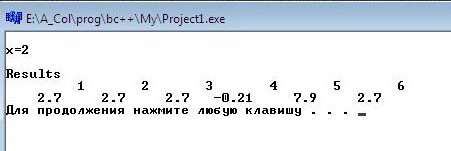
**cout<< setw(7) << setprecision(2)<< y4<<"\t";**

**cout<< setw(7) << setprecision(2)<< y5<<"\t"<<y6<<endl;**

**system("pause"); *// пауза***

**return 0; *// повернення результату (вихід із функції)***

**}**



Результати виконання.

В C++ оператори об’єднуються в блоки — функції. **Функція** — це послідовність операторів. Кожна програма в C++ повинна містити **головну функцію main()**. Саме з першого оператора в main() і починається виконання програми. Функції, як правило, виконують конкретне завдання. Наприклад, функція max() може містити оператори, які визначають максимальне число з двох переданих їй. Функція calculateGrade() може вираховувати оцінку студента..

**Порада**: Пишіть функцію main() в файлі .cpp з тим же іменем, що має проект. Наприклад, якщо ви пишете програму Chess, то помістіть вашу функцію main() в chess.cpp.

У загальному випадку програма складається з декількох функцій, що не перетинаються (тобто «вкладення» однієї функції в іншу неприпустиме). Перед функціями і між ними можуть бути присутні оголошення об’єктів даних і оператори препроцесорної обробки. Функції користувача, які викликаються у головній функції main(), слід обов’язково описати до їх використання. Наведемо приклад запису фрагмента простої програми:

//..............фрагмент простої програми на С++

#include <iostream.h>

#include <соnio.h> //директиви препроцесора

const int n=20;

void main()

{

float mas[n]; //опис одновимірного масиву

//..............введення елементів масиву

for (int i=0; i < n; i++)

cin >> mas[i];

//......виконання перетворень

//......виведення перетвореного масиву

for(int i=0; i < n; i++)

cout << " " << mas[i] << " ";

getch (); // затримка результату на екрані

}

**Коментарі** необхідні для пояснень призначення тих чи інших частин програми і їх текст завжди ігнорується компілятором. Мова С++ використовує два різновиди коментарів:

* *// текст* — **однорядковий коментар**, який починається з двох символів «/» («коса риска») і закінчується символом переходу на новий рядок;
* */\* текст \*/* — **багаторядковий коментар**, що розташовується між символами-дужками «/\*» і «\*/».

Багаторядкові коментарі не можуть бути вкладеними один в одний, а однорядкові коментарі можна вкладати в багаторядкові коментарі. Багаторядкові коментарі доцільно застосовувати для тимчасового виключення блоків при налагодженні програми.

Наведемо кілька порад стосовно раціонального складання коментарів:

* коментарі повинні бути добре складеними реченнями, мати правильну пунктуацію та містити тільки потрібну для супроводу інформацію;
* пропуск — один з найбільш ефективних коментарів, що значно поліпшує розуміння програми;
* штрихові лінії коментаря або порожні рядки застосовуються для поділу функцій та інших логічно завершених фрагментів програм.

Директива препроцесора **#include<iostream.h>** забезпечує підключення до програми засобів зв’язку зі стандартними потоками введення-виведення даних. Ці засоби знаходяться у заголовному файлі **iostream.h**, де **і (*input*)** — введення, **о (*output*)** виведення, **stream** — поток, **h (*head*)** — заголовок. Ураховуючи те, що середовище **Borland C++** не забезпечує затримку результатів на екрані, у текстах програм посібника з цією метою використано стандартний потік введення **сіn** (бібліотека **iostream.h**), функцію введення **getch()** (бібліотека **conio.h**) тощо.

При створенні програми враховують такі основні вимоги:

* усі використані константи, змінні, функції та нестандартні типи повинні бути оголошеними (описаними) до їхнього першого використання, і ці оголошення можна розміщати в будь-якому місці програми;
* кожний оператор мови закінчується символом **«;»;**
* фігурні дужки (**« { »** та **« } »**) виділяють складений оператор і все, що подано між такими дужками, синтаксично сприймається як один оператор;
* вкладені блоки повинні мати відступ у 3-4 символи, при цьому блоки одного рівня вкладеності слід вирівняти за вертикаллю.

**Макровизначення.**

***Макровизначення*** (*макропідстановка* або просто *макрос*)– це вираз, який при компіляції файлу з кодом програми підставляється замість символьного імені, що визначає дану макропідстановку. Макроси дозволяють замінити великий текст коротким ім'ям макросу.

Визначаються макроси директивою препроцесора ***#define***. Формат завдання:

***#define ім'я\_макросу текст\_макросу***

Перед компіляцією замість імені макроса в програму підставляється його значення.

Директива **#define** – слугує для заміни у всіх місцях коду програми замість вказаної послідовності символів обрану користувачем. За негласною домовленість слова підстановки записують великими літерами, для візуального виділення у програмі.

Наприклад,

#define MAXNUMBER 100  
У всіх місцях коду замість MAXNUMBER буде підставлено 100.  
Тобто при компіляції рядок  
if (a>MAXNUMBER) { … }  
Буде замінено на:  
if (a>100) { … }

#define SIMPLETEXT “lorem ipsum dolor sit amet”  
Аналогічно замість:  
printf(SIMPLETEXT);  
printf(“lorem ipsum dolor sit amet”);

#define РІ 3.14159

Дана директива вказує замінити всі появи у тексті програми символьної константи РІ на чисельну константу 3.14159. Якщо є необхідність змінити значення такої константи у всій програмі, то достатньо це зробити в одному місці програми, в директиві #define, і після повторної компіляції програми всі включення константи в програму будуть автоматично замінені.

Дозволяються також макроси з параметрами, останні підставляються в текст заміни, після чого макрос розширюється, тобто в програму підставляється текст заміни замість ідентифікатора і списку параметрів. Наприклад, для макровизначення з одним аргументом для знаходження площі круга:

#define CIRCLE\_AREA (x) ((PI)\*(x)\*(x))

Кожен раз, коли в програмному коді з’явиться CIRCLE\_AREA (x), значення х підставляється замість х в тексті заміни, символьна константа РІ автоматично заміняється її значенням (визначеним раніше), і макрос розширюється в програмі. Наприклад, оператор

area=CIRCLE\_AREA (4)

розширюється в

area=((3.14159)\*(4)\*(4)).

Макроси є спадщиною мови С, і в С++, зазвичай, замінюються більш безпечними можливостями мови, такими, як шаблони, inline-функції, перерахування enum, перевизначення типів typedef.

Символьні константи і макроси можна відмінити, використовуючи директиву препроцесора ***#undef***. Ця директива відміняє визначення символьної константи або імені макроса. Область їх дії продовжується від місця визначення в програмному коді до місця відміни визначення за допомогою директиви #undef, чи до кінця файлу, якщо дана директива відсутня. Після дії директиви #undef дозволяється повторно визначати макрос за допомогою директиви #define.

**Директиви умовної компіляції**.

Умовна компіляція дозволяє управляти виконанням директив препроцесора і компіляцією програмного коду. Директиви умовної компіляції: #if, #elif, #else та #endif. Кожна конструкція #if закінчується #endif. Умовні конструкції препроцесора, що перевіряють декілька варіантів, реалізуються за допомогою #elif (еквіваленту else if структури if) і #else (еквіваленту else структури if). Порядок використання директив умовної компіляції:

**#if константний\_вираз\_1**

**…**

**#elif константний\_вираз\_2**

**…**

**#elif константний\_вираз\_3**

**#else**

**…**

**#endif**

Кожна із умовних директив препроцесора оцінює значення цілочисельного виразу. Якщо цілий константний вираз у директиві #if має ненульове значення (TRUE), то при компіляції включаються всі наступні рядки до #elif або #endif або #else (elif діє як гілка else-if). Наприклад, змінюючи константу VERSION, можна керувати включенням файлів:

#define VERSION 3

#if VERSION == 1

#define INCLUDE\_FILE "file\_1.h"

#elif VERSION == 2

#define INCLUDE\_FILE "file\_2.h"

#else

#define INCLUDE\_FILE "file\_3.h"

#endif

#include INCLUDE\_FILE

**Зовнішні оголошення**

*Зовнішні оголошення* – це глобальні описи даних (змінних або констант). Змінна у програмуванні є моделлю (зображенням) об’єкта в пам’яті комп’ютера. На фізичному рівні поняттю змінної відповідає група комірок оперативної пам'яті. Ці адреси ставляться у відповідність ідентифікаторам (іменам) змінних під час їх оголошення. Таким чином, ім'я змінної *вказує* (або *посилається*) на першу комірку з групи, а величину групи визначає тип змінної (рис. 1).

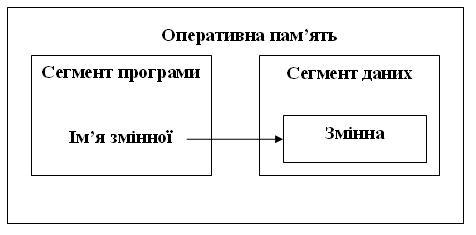


Рис. 1. Ідентифікатор змінної та його асоціація з коміркою пам'яті

**Функція** – це частина програми, що виконує логічно завершений набір дій. В C/С++ вона є єдиним способом представлення підпрограм. Формат оголошення функції:

**тип\_результату ім'я([параметри]) *// заголовок функції***

**{ … } *// тіло функції***

Тіло функції може містити описи констант, змінних і оператори мови, які відокремлюються символом «;». Описи даних можуть зустрічатися в будь-якому місці функції, але до першого звертання до змінної. Якщо параметри у функції відсутні, то в заголовку все рівно записуються круглі дужки. Одна з функцій програми на мові C/С++ повинна мати ім'я main***.*** Саме з неї починається виконання програми. Функція main визначає дії, що виконуються програмою, і викликає інші функції. Якщо програма містить тільки одну функцію, то вона і є головною (має ім'я main).

Програма може містити довільне число директив препроцесора, оголошень і визначень змінних, описів функцій. Порядок появи цих елементів у програмі досить істотний: він впливає на можливість використання змінних, функцій і типів у різних частинах програми.

Логічні операції та операції порівняння в мові Сі оперують поняттями "істина" і "хиба

**Операції порівняння**

У мові С++ можна використовувати операції порівняння:

|  | **операція** | **відповідник** | **приклади** |
| --- | --- | --- | --- |
| Менше  Менше або дорівнює  Більше  Більше або дорівнює | <  <=  >  >= | <  ≤  >  ≥ | a < b  a<=b  a>b  a>=b |
| Дорівнює  Не дорівнює | ==  != | =  ≠ | a==b  a!=b |

Порівнювати можна операнди будь-якого типу, але вони повинні бути того ж самого вбудованого типу даних (порівняння на рівність і нерівність працює для двох величин будь-якого однакового типу даних), або між ними повинна бути визначена відповідна операція порівняння. Результат – логічне значення true або false. Використовуються для порівняння значень і дають відповідь false (0) true (1).

Операнди можуть бути символи, логічні, числові, покажчики.

Умова, яка перевіряється, може містити значення, яке не є логічним. Тоді значення нуль (0) сприймається як false, а значення  ≠0 - як true. Наприклад:

**int a = 10;**

**int b = 4;**

**int c = a == b; // 0**

**int c = 10 <= 4; // 0**

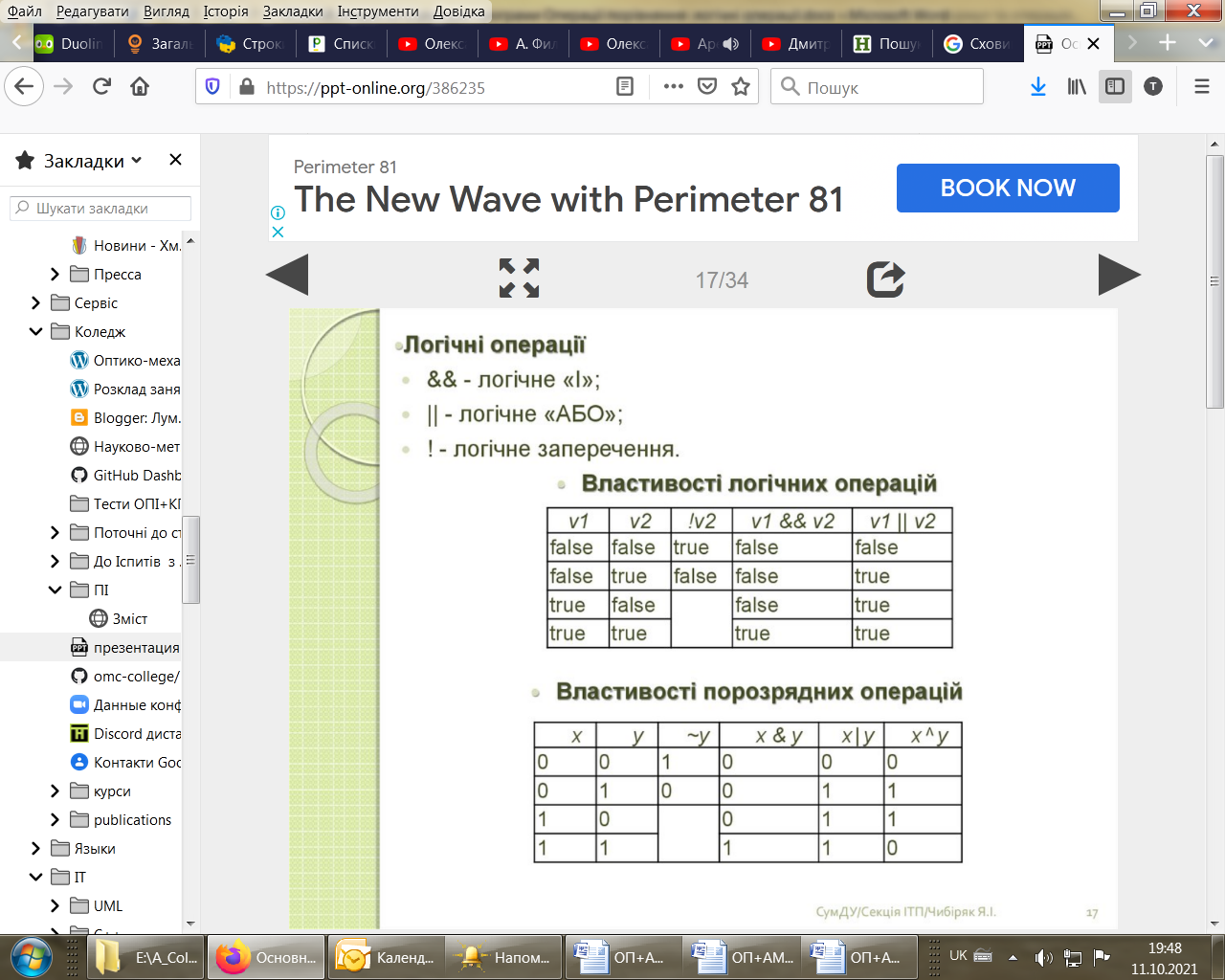
**Логічні операції**

Логічні операції реалізують операції математичної логіки. Серед логічних операцій є унарні й бінарні.. Дають відповідь типу bool:  false (0)  та true (1).

| **Операція** | **Відповідник** | **Приклади** |
| --- | --- | --- |
| ! (унарна) | not | !b, !(c>1 && c<9) |
| &&(бінарна) | and | a && b, a>1 && c<9 |
| || (бінарна) | or | a||b, c<0 || c>10 |

Операнди логічних операцій можуть бути логічні змінні та відношення.

Правила виконання логічних операцій



**Логічні вирази**

Логічні (булівські) вирази складаються з  логічних операцій та операндів - логічних змінних і значень та відношень, з урахуванням пріоритетів. Для зміни порядку обчислень можна використовувати круглі дужки. Результат виконання має тип ***bool***. Правила побудови загальні - лінійна форма написання, явно вказуються знаки операцій та операнди без дублювання.

Приклади запису логічних виразів:

**x<0 || x>11**

**(i!=0 ||i>100) && (j!=i || j>0)**

**x!=0 && x!=10 && x!=100**

Вираз може містити операнд, який не є логічним. Тоді значення нуль (0) сприймається як false, а значення  ≠0 - як true. Приклади:

**// логічні операції**

**bool res;**

**int a, b;**

**// операція && (AND)**

**a = 8;**

**b = 5;**

**res = a && b; // res = True**

**a = 0;**

**res = a && b; // res = False**

**// операція || (OR)**

**a = 0;**

**b = 0;**

**res = a || b; // res = False**

**b = 7;**

**res = a || b; // res = True**

**// операція ! (логічне "НІ")**

**a = 0;**

**res = !a; // res = True**

**a = 15;**

**res = !a; // res = False**

**Тернарна операція (?:)**

Тернарна операція —це операція, що має 3 операнда. Використовується як альтернатива запису умовного оператора. Синтаксис тернарної операції:

**B ? V1 : V2.**

Виконання:

1. Спочатку обчислюється значення виразу B.

2. Якщо B==true(1), то обчислюється значення виразу V1,

інакше - обчислюється значення виразу V2.

Для читабельності  рекомендується брати у круглі дужки умову та вирази. Приклади:

**(i < 1) ? 1 : i;**

**max = (a<=b) ? b : a;**

**х<0 ? -x : x ;**

**Введення-виведення значень виразів**

Умові С++ основним поняттям уведення й виведення даних є потік – послідовність символів або інших даних. У програмі потік є представником фізичного файлу на зовнішньому носії даних (диску, клавіатурі або екрані монітора), а операції обміну даних із файлом зображено як операції добування даних із потоку або дописування їх до нього.

У програмуванні існує поняття стандартних файлів уведення й виведення – зазвичай ними є клавіатура та екран. У С++-програмі їм відповідають стандартний потік уведення з ім'ям cin і стандартний потік виведення з ім'ям cout. Отже, імена cin і cout, означені у файлі iostream, насправді позначають не клавіатуру та екран, а потоки, що відповідають цим пристроям у програмі. На початку виконання програми обидва потоки cin і cout порожні.

Із засобів обробки потоків розглянемо лише операції введення >>, або добування, даних зі вхідного потоку, і виведення <<, або вставки (insertion), даних у вихідний потік. Розгляд інших засобів виходить за межі цієї книги.

Вираз із операцією введення (вставки з потоку) >> має вигляд cin >> ім'я-змінної. Додавши в кінці виразу символ ;, маємо інструкцію введення.

Виконуючи операцію введення, комп'ютер забирає зі вхідного потоку послідовність непорожніх символів, за якою створює відповідне значення та присвоює змінній. Усі порожні символи пропускаються. Якщо у вхідному потоці немає непорожніх символів (потік порожній), то комп'ютер очікує на його поповнення. Потік поповнюється, коли людина набирає на клавіатурі деяку послідовність символів (вони з'являються на екрані) і натискає на клавішу Enter.

Згідно з типами змінних, в які вводяться значення, операція >> розбиває вхідний потік на лексеми (послідовності непорожніх символів) і перетворює їх на значення певних типів (інтерпретує лексеми). Отже, операцію >> можна розглядати як інтерпретацію вхідного потоку символів.

Значенням виразу введення є той потік, з якого взято символи. До нього знову можна застосувати операцію >>, тобто можливі вирази вигляду

cin >> ім'я-змінної\_1 >> ім'я-змінної\_2

та аналогічні їм із більшою кількістю змінних. Наведений вираз еквівалентний виразу

(cin >> ім'я-змінної\_1) >> ім'я-змінної\_2

й задає послідовне введення значень двох змінних.

Приклад. Якщо є змінні char c1, c2; і під час виконання

cin >> c1 >> c2; натиснути клавіші Space, W, Space, Space, 3, Q, Enter, то змінні отримають значення 'W' та '3', а Q залишиться у вхідному потоці.

За стандартних налаштувань операція >> пропускає всі порожні символи вхідного потоку.

За стандартних налаштувань, утворюючи за вхідною послідовністю числове значення, операція >> розглядає вхідну послідовність як десяткове зображення числа. Дійсні числа можна задавати в нормалізованій формі, а також без дробової частини. У дробовій частині використовується крапка, а не кома. Для цілих чисел дробова частина й порядок не допускаються; задане число повинно належати діапазону можливих значень типу, який має змінна, інакше введення може мати непередбачувані наслідки.

Приклад. Якщо є змінна int k; і під час виконання cin >> k; натиснути клавіші 0, 1, 1, Enter, то значенням k буде 11. Якщо є змінна double x; і під час виконання cin >> x; натиснуто клавіші 8, Enter, то x отримує значення 8.0, а якщо 1, e, 2, Enter, – значення 100.0.

Під час уведення з клавіатури помилки найчастіше трапляються з числовими змінними. Якщо під час виконання виразу введення сталася помилка, то вхідний потік переходить у стан помилки, в якому виконання подальших операцій уведення неможливе. За наявності помилки значення змінної, в яку мало відбутися введення, не змінюється. Отже, якщо змінну не ініціалізовано та значення їй не присвоєно, то її значення залишається "випадковим сміттям"!

**Класифікація змінних в залежності від місця оголошення в програмі**

Змінні можуть бути оголошені:

* всередині функцій;
* у визначенні параметрів функцій;
* за межами усіх функцій.

В залежності від місця оголошення в програмі, змінні класифікуються наступним чином:

* локальні змінні (оголошуються всередині функцій);
* формальні параметри (є параметрами функцій);
* глобальні змінні (оголошуються за межами функцій).

***Локальні змінні*** – це змінні, які оголошені (описані) всередині функцій. Доступ до таких змінних мають тільки оператори, що використовуються в тілі функції.

Локальні змінні створюються в момент виклику функції. Як тільки виконання функції завершується, локальні змінні знищуються.

Локальні змінні створюються в момент виклику функції. Як тільки виконання функції завершується, локальні змінні знищуються.

Термін “знищення змінної” означає: знищення змінної в оперативній пам’яті після завершення функції, в якій ця змінна була оголошена.

Глобальна змінна – це змінна, доступ до якої мають усі функції в програмі. Глобальні змінні зберігають свої значення на протязі усього часу виконання програми. Цей час називається часом життя глобальної змінної.

**Приклад.** У даному прикладі оголошено глобальну змінну maxi та дві функції GetMax2() та GetMax3().

#include <iostream>

#include <Windows.h>

using namespace std;

int maxi=0; // maxi - глобальна змінна

// Оголошення функції GetMax2

int GetMax2(int a, int b)

{ int maxi=1;

// a, b - формальні параметри

// використання глобальної змінної max у функції GetMax2()

if (maxi<b) ::maxi = b;

return maxi;

}

// Оголошення функції GetMax3

int GetMax3(int a, int b, int c)

{ int maxi;

// a, b, c - формальні параметри

// використання глобальної змінної max у функції GetMax3()

if (::maxi<b) maxi = b;

if (::maxi<c) maxi = c;

return maxi;

}

int main()

{system("color F0");

int gma;

**// Зверніть увагу на виведення maxi!**

cout << "1 maxi="<< maxi <<endl;

cout <<" Getmax2(1, 2)=" <<GetMax2(1, 2)<<";Getmax3(4, 5, 6)="<<

GetMax3(4, 5, 6)<< " maxi="<< maxi << endl;

maxi=0;

cout << "2 maxi="<< maxi <<endl; **// Зверніть увагу на виведення maxi!**

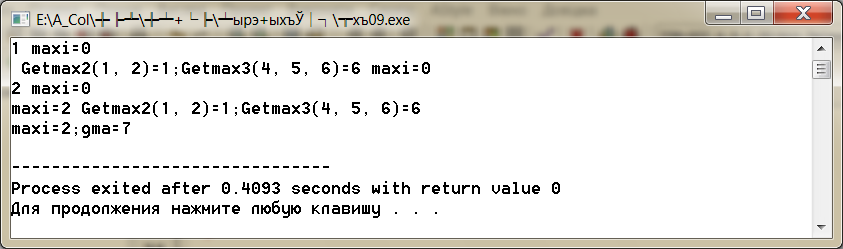
cout << "maxi="<< maxi << " Getmax2(1, 2)=" << GetMax2(1, 2)<<";Getmax3(4, 5, 6)="<<GetMax3(4, 5, 6)<<endl;

gma=GetMax2(1, 2) + GetMax3(4, 5, 6);

cout << "maxi="<< maxi<< ";gma="<<gma<<endl;

return 0;

}



При розробці власних програм, що містять глобальні змінні, може виникнути ситуація коли всередині функції використовується локальна змінна з таким самим іменем. У цьому випадку ім’я локальної змінної перекриває ім’я глобальної змінної. При звертанні до імені всередині функції приймається до уваги локальна змінна (глобальна змінна ігнорується). Якщо виникла така ситуація, то для доступу до глобальної змінної можна використати операцію розширення області видимості, яка позначається **::** .

З допомогою операції розширення області видимості **::** можна отримати доступ не тільки до змінних, але і до констант та функцій.

**Приклад.**

У прикладі продемонстровано:

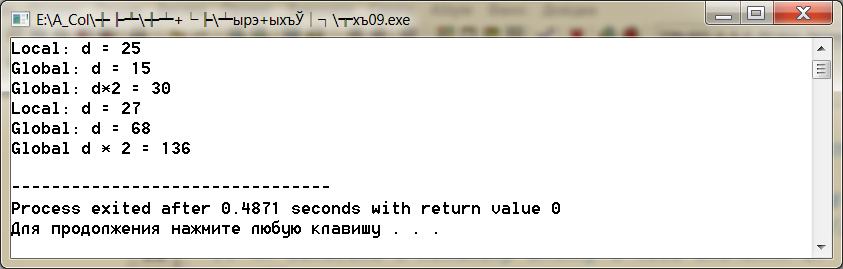
* використання локальних та глобальних змінних;
* використання прототипу функції;
* використання операції розширення області видимості **::** для доступу до глобальної змінної.

#include <iostream>  
using namespace std;  
// Глобальні та локальні змінні. Операція ::  
// Оголосити глобальну змінну d.  
// Ця змінна є доступна в обох функціях: MultGlobal2() та main()  
int d = 15;  
// Прототип функції, яка множить глобальну змінну d на 2  
int MultGlobal2(void);  
int main()  
{  
  // Оголосити локальну змінну d у функції main()  
  int d;  
  // 1. Вивести значення локальної змінної d  
  d = 25;  
  cout << "Local: d = " << d << endl; // d = 25 - локальна змінна перевизначає глобальну змінну  
  // 2. Вивести значення глобальної змінної d,  
  //   використовується операція :: - розширення області видимості  
  cout << "Global: d = " << ::d << endl;  
  // 3. Вивести подвоєне значення глобальної змінної d  
  cout << "Global: d\*2 = " << MultGlobal2() << endl; // d\*2 = 30  
  // 4. Записати в локальну змінну d нове значення 27  
  d = 27;  
  cout << "Local: d = " << d << endl;  
  // 5. Записати в глобальну змінну d нове значення 68  
  ::d = 68;  
  cout << "Global: d = " << ::d << endl; // глобальна змінна d = 68  
  // 6. Вивести подвоєне значення глобальної змінної ::d  
  cout << "Global d \* 2 = " << MultGlobal2() << endl; // буде виведено 136

return 0;

}// Функція, яка використовує значення глобальної змінної d.  
// До локальної змінної d у функції main() ця функція доступу не має  
int MultGlobal2(void)  
{  
  // тут доступна глобальна змінна d  
  return d \* 2;  
}

Результат виконання програми



При оголошенні змінної для неї виділяється пам’ять. Розмір пам’яті, що виділяється, залежить від типу змінної. Розмір виділеної пам’яті впливає на діапазон значень, які може приймати змінна.

**Приклад. Дослідження виконання інкременту.**

**#include <iostream>**

**#include <Windows.h>**

**using namespace std;**

**int main()**

**{system("color F0");**

**int i=0,j,k;**

**cout << "1 i++="<< i++ << endl;**

**i=0;**

**cout << "2 ++i="<< ++i<< endl;**

**i=0;**

**cout << "3 ++i="<< ++i << " i++="<< i++ << endl;**

**k=i=0;**

**j=k+++i++;**

**cout <<"4 j="<<j<<";k="<<k<<";i="<<i<<endl;**

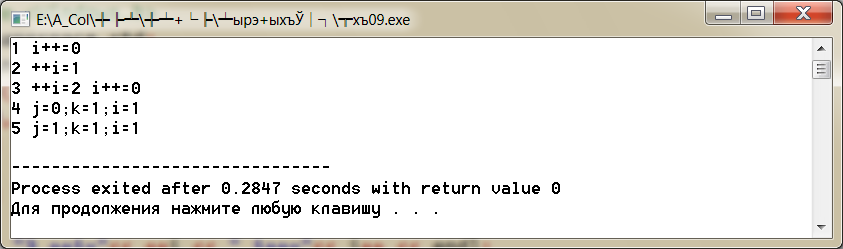
**k=i=0;**

**j=k+++(++i);**

**cout <<"5 j="<<j<<";k="<<k<<";i="<<i<<endl;**

**return 0;**

**}**



***Для самостійного вивчення (2 години)*:** Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

***Література***

1. Вступ до програмування мовою С++. Організація обчислень: навч. посіб. / Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставовський. – К. : Видавничо-поліграфічний центр "Київський університет", 2012. – 175 с. с.: іл. ISBN (укр.) . URL: <http://csc.knu.ua/uk/library/books/belov-24.pdf>
2. Джейс Либерти Освой самостоятельно С++ за 21 день: 3-е изд. пер. с англ.: Уч. пос. – М.: Издательский дом „Вильямс”, 2001. – 816 с.: ил..
3. Бондарев В. М. Программирование на С++: Учеб. пособие. — Харьков: СМИТ, 2004г. — 294 с. URL: <https://www.rulit.me/author/bondarev-v-m/programmirovanie-na-c-get-161082.html>
4. Липпман С. Б., Лажойе Ж. Язык программирования С++: Вводный курс. — М.: ДМК, 2001. URL: <http://www.insycom.ru/html/metodmat/inf/Lipman.pdf>
5. Дейтел Х., Дейтел П. Основы программирования на С++. – М.: Бином, 1999. – 1024 с.

***Контрольні запитання***.

1. Визначте основні частини типової структури програми на С++.
2. В чому особливості функції main()?
3. Які особливості функції користувача?
4. Для чого призначені коментарі, які їх форми, які рекомендації по їх складанню?
5. Що таке макровизначення? Наведіть приклади.
6. Для чого використовується умовна компіляція?
7. Наведіть директиви умовної компіляції.
8. Що таке зовнішні оголошення?
9. Що може міститися в зовнішніх оголошеннях?
10. Надайте визначення функції.
11. Які операції порівняння ви знаєте та з якими операндами вони застосовуються?
12. Які логічні операції ви знаєте?
13. Як визначається тернарна операція?
14. Як реалізується введення-виведення значень виразів?
15. Як класифікуються змінні в залежності від місця оголошення в програмі?
16. Для чого використовується операція розширення області видимості **::?**
17. Які значення може містити умова, що перевіряється?
18. Як визначається результат операція **not?**
19. Як визначається результат операція **and?**
20. Як визначається результат операція **or?**
21. Поясніть, що представляє собою логічний вираз.