**Практична робота №5.** Розробка простої програми з операціями порівняння, використанням математичних формул та операціями введення та виведення.

**Мета:** навчитись складати прості програми з операціями порівняння, використанням математичних формул та операціями введення та виведення.

**Хід роботи**

1. Ознайомитися з теоретичними відомостями лекції №8.
2. Запустити в Інтернет-браузері он-лайн компілятор C++Shell за посиланням <http://cpp.sh/> або використати інший доступний вам компілятор С++
3. Скласти програму, яка:
   1. містить глобальну змінну цілого типу.
   2. В головній функції оголошені 2 змінні: ціла (х) та дійсна;
   3. значення цілої вводиться с консолі, значення дійсної задається ініціалізацією.
   4. Дійсна змінна збільшується на значення, яке розраховується як квадратний корінь від різниці глобальної змінної та заданої в головній функції цілої змінної, якщо цей підсумок більше нуля, або за формулою

, де α приймає значення глобальної змінної, а х – змінна, значення якої приймається з консолі. (використовуємо **триарну операцію ?).**

* 1. . Виводяться значення змінних до розрахунку і після нього у вигляді <ідентифікатор змінної>=<значення>. Значення відділяються комами.

1. В цій же програмі далі виконується розрахунок значень z1, z2, z3 за відповідними формулами, розподіл яких по групах наданий наприкінці завдання, а також в ОП+АМ.ПР 05.Завдання..docx

Результати розрахунку по кожній формулі зберігаються в окремій змінній (z1, z2, z3).

Вивести результати таким чином:

* Перший рядок заголовок"Results".
* Другий рядок через табуляцію виводяться номери формул (1, 2 3).
* Третій рядок - через табуляцію результати розрахунку з використанням маніпуляторів **setw(w)** та **setprecision(d**), обравши довжину поля 7, кількість позицій у дробовій частині дійсних чисел – 2.

1. Перший рядок програми повинен містити коментар з номером ЛР, номером групи та прізвищем студента.
2. Значення змінних вводяться з консолі.
3. Програма компілюється, результати перевіряються на правильність. Програму збережіть для подальшої роботи з удосконалення.
4. Результати (код програми, скріншот виконання) оформити, надавши титульний лист, завдання, та результати, а також окремо код програми у вигляді cpp-файлу з іменем у форматі

**<Номер групи><Номер лабораторної><Прізвище>**

Наприклад, 23-04Ivanov.cpp

надсилати на електронну адресу викладача [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді cpp-файлу з іменем у форматі

**<Номер групи><Номер лабораторної><Прізвище>**

Наприклад, 23-04Ivanov.cpp.

Іншим рішенням є надсилання поштою посилання на текст програми за URL адреси, яку надає C++Shell, вказавши в темі листа, номер групи прізвище студента та номер ЛР.

**Строк відсилки ЛР для ІПЗ-31 19.09.2024**

**ІПЗ-32 21.09.2024**

**ІПЗ-33 20.09.2024**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача.

В темі листа вказати, номер групи, прізвище студента та номер ПР як "ПР№5" В темі листа вказати, номер групи, прізвище студента, номер ПР та фразу "Запитання".

**Порядок роботи з C++Shell**

1. В браузері набрати // <http://cpp.sh/> для виклику он-лайн компілятору C++SHELL, буде надане вікно (рис.1)

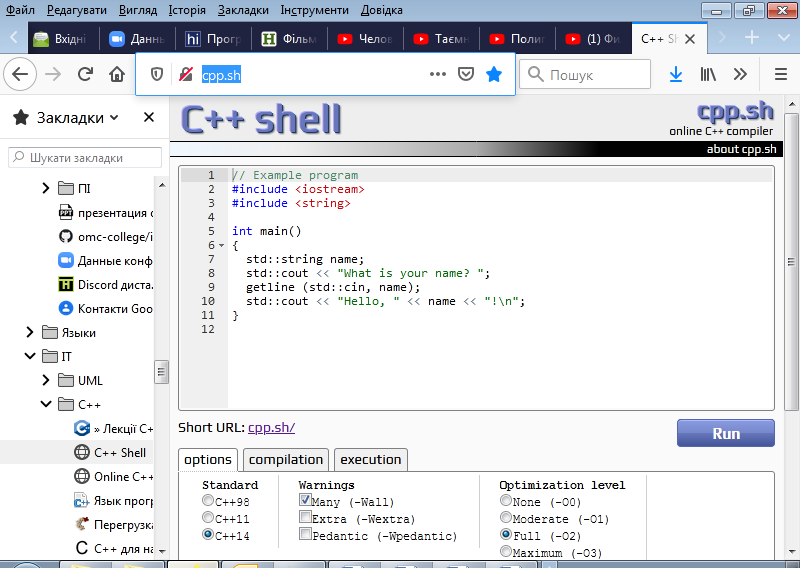


Рисунок 1 – вікно он-лайн компілятору C++SHELL

1. В цьому вікні можна відкоригувати текст відповідно до наданого на рис.2, вказавши в першому рядку своє прізвище.

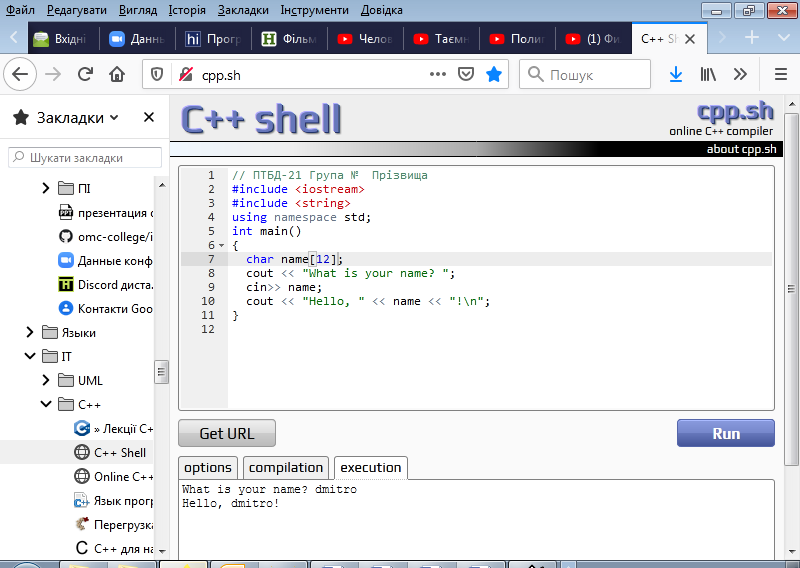


Рисунок 2 – текст зміненої програми

1. Натиснути клавішу **Run** для запуску на виконання і у закладці **execution** ввести ім’я за прикладом на рис.3.

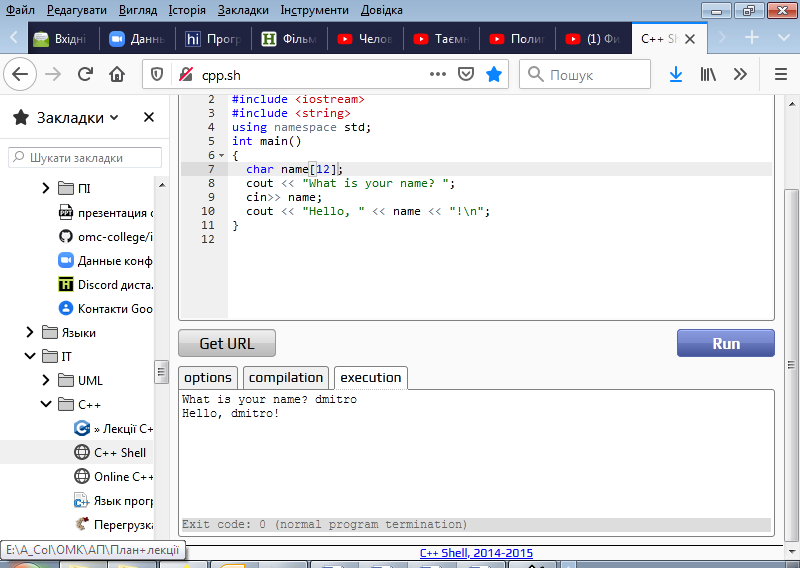


Рисунок 3 – виконання програми

1. Натисніть клавішу **Get URL,** після тексту Short URL буде виведене посилання на створену програму (див. приклад на рис.1, це cpp.sh/). Скопіюйте URL – адресу, наприклад, до блокноту, і надішліть в разі потреби як результат виконання ПР..

**Порядок роботи з Dev-C++**

Можна також використати безкоштовний компілятор Dev-C++, який потрібно встановити на ваш комп’ютер. Скачати його можна за посиланням:

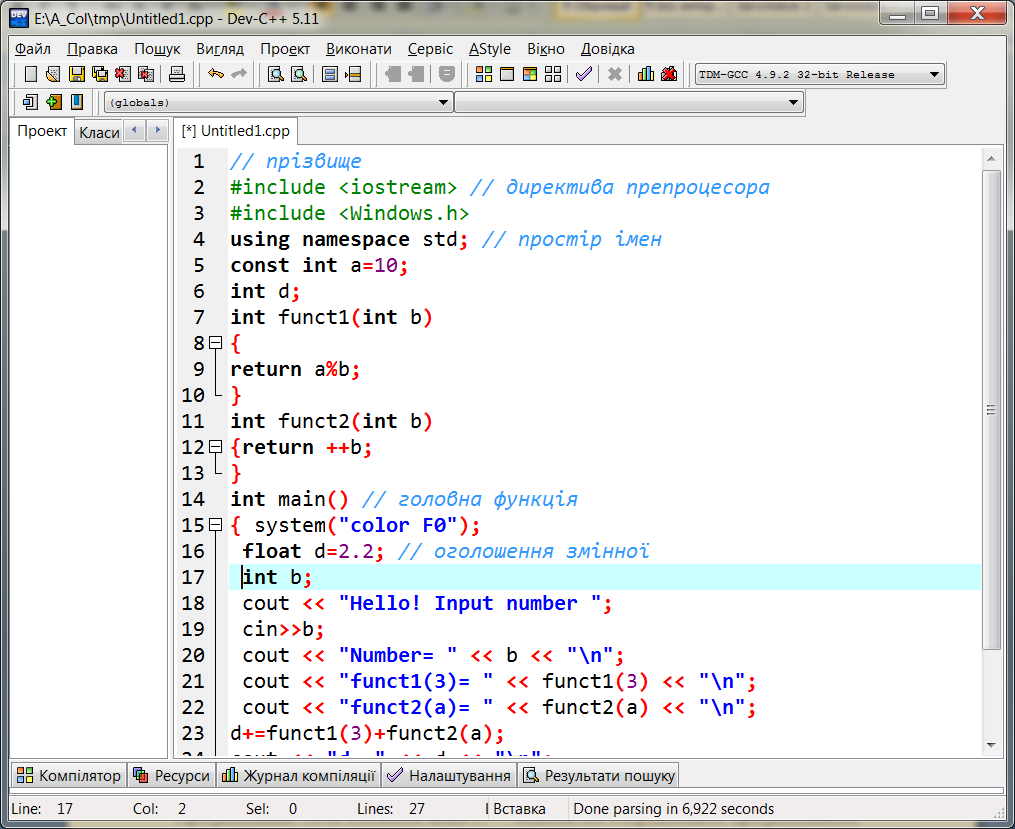
**https://sourceforge.net/projects/orwelldevcpp/**

Інструкція по роботі з компілятором надається за посиланням (але там забагато реклами та іншого сміття):

**https://studopedia.ru/18\_64672\_pokrokove-vikonannya-programi.html**

Інтерфейс програми представлено на рис.4.

Перед виконанням компіляції нової програми буде запит на її збереження на жорсткому диску. Можна створити файл для нової програми, натиснувши на відповідну піктограму (див. рис.4). Зазвичай результати роботи програми виводяться на чорний екран, щоб змінити фон на білий (див. рис.6) на рис.5 вказані оператори, які потрібно додати в свою програму.



Для робот з "білим" екраном виконання

Натиснути для компіляції і виконання

Натиснути для компіляції

Натиснути для запису нової на диск

Рисунок 4 – інтерфейс Dev-C++

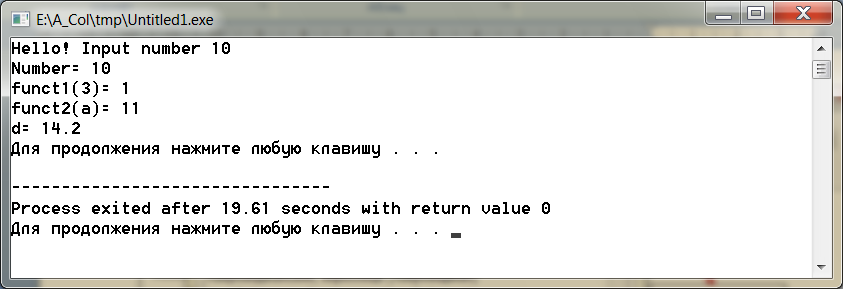
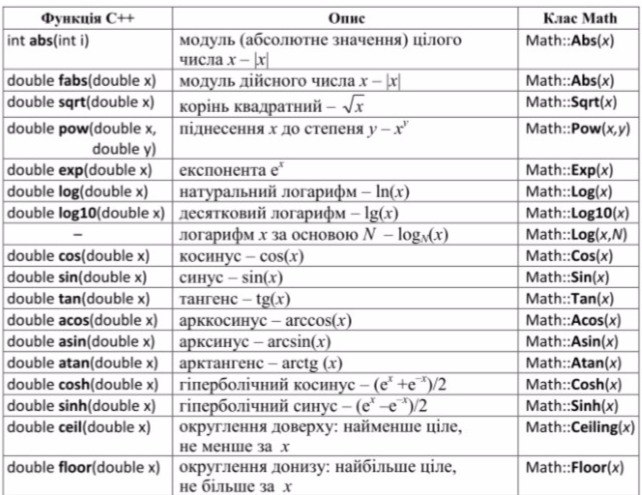


Рисунок 5 – екран виконання

**Теоретичні відомості**

**Основні математичні функції C++ та класу Math**

****

**Правила запису математичних виразів**

* Кількість відкритих і закритих дужок у виразах повинна бути однаковою.
* Усі елементи виразів (дроби, показник ступеню, індекси) записують у горизонтальному рядку.
* Вирази можна записувати у декількох рядках. "Розривати” вирази можна, наприклад, після символу арифметичної операції. Власне символ дублювати не потрібно.
* Операції ++ (інкремент) та -- (декремент) є унарними, тобто мають лише один операнд.
* Операція ++ додає одиницю до операнду, операція -- віднімає одиницю від операнду.

Обчислення в арифметичних виразах виконуються зліва направо згідно з  таким пріоритетом операцій:

1. стандартні функції, ++, --;

2. множення (\*), ділення (/), остача від ділення (%);

3. додавання (+) та віднімання (–).

Для здобуття правильного результату слід дотримуватися таких правил записування арифметичних виразів в операторах С++:

* кожна команда (інструкція) має завершуватись крапкою з комою (;)
* мова С++ є чутлива до регістру, тобто х та Х – це дві різні змінні;
* аргумент функції завжди записують у круглих дужках;
* знаки множення не можна пропускати (3ab це 3\*a\*b);
* якщо знаменник або чисельник має операції (+, −, \*, /), то його слід записувати у круглих дужках;
* для записування раціональних дробів, у чисельнику або знаменнику яких є числові константи, хоча б одну з цих констант слід записати як дійсне число із зазначенням десяткової крапки, наприклад, 2⁄ 𝑘 записують як 2.0/k;
* радикали (тобто корінь кубічний і вище) замінюють на дробові степені, наприклад, кубічний корінь (𝑥 + 1) записують як  pow(x+1, 1/3.0);
* слід враховувати правила зведення типів, оскільки в арифметичних виразах можуть брати участь різнотипні дані та відбувається зведення типів.

**Приклади запису математичних виразів**



Для додаткового керування даними, що виводяться, використовують маніпулятори **setw(w)** та **setprecision(d)**. Маніпулятор **setw(w)**призначений для зазначення довжини поля, що виділяється для виведення даних (**w** — кількість позицій). Маніпулятор **setprecision(d)** визначає кількість позицій у дробовій частині дійсних чисел.

Маніпулятори змінюють вигляд деяких змінних в об’єкті **cout**, що у потоці розташовані за ними. Ці маніпулятори називають *прапорцями стану*. Коли об’єкт посилає дані на екран, він перевіряє прапорці, щоб довідатися, як виконати завдання, наприклад, запис:

**cout << 456 << 789 << 123;**

призводить до виведення значення у вигляді: **456789123**, що ускладнює визначення групи значень.

***Приклад.*** Написати програму, використовуючи маніпулятор **setw()**.

**#include <iostream.h>**

**#include <iomanip.h>**

**#include <conio.h>**

**main ( )**

**{**

**cout << 456 << 789 << 123 << endl;**

**cout << setw(5) << 456 << setw(5) << 789 << setw(5) << 123 << endl;**

**cout << setw(7) << 456 << setw(7) << 789 << setw(7) << 123 << endl;**

**getch ();**

**}**

Результати виконання програми:  
**456789123**   
**456     789     123**   
**456       789       123**

У цьому прикладі з’явився новий заголовний файл **iomanip.h**, що дозволяє застосовувати функції маніпуляторів. При використанні функції **setw()** число вирівнюється вправо в межах заданої ширини поля виведення. Якщо ширина недостатня, то вказане значення ігнорується.

Функція **setprecision(2)** повідомляє про те, що число з плаваючою крапкою виводиться з двома знаками після крапки з округленням дробової частини, наприклад, при виконанні операції

**cout** **<< setw(7) << setprecision(2) << 123.456789;**

буде отримано такий результат: **123.46.**

Функції **cout.width(w)** та **cout.precision(d)**, які потребують підключення тільки заголовного файла **iostream.h**, виконують дії, подібні тим, що і функції **setw(w)** та **setprecision(d)**.

Операція введення використовує ті ж самі маніпулятори, що й операція виведення.

##### ***Приведення типів в операціях +, –, \****

У виразах, в яких фігурують операції **+**, –, **\*** діють такі правила приведення типу результату:

* якщо обидва операнди мають цілий тип, то результат також буде цілого типу;
* якщо хоча б один з операндів має дійсний (з плаваючою комою) тип а інший цілий тип, то результат також буде дійсного типу;
* якщо один з операндів має тип float, а інший тип double, то результат буде типу double. Це зв‘язано з тим, що тип double потребує більше пам‘яті ніж тип float. У цьому випадку відбувається розширення типу float до типу double.

##### ***Пріоритет та асоціативність арифметичних операцій***

Арифметичні операції мають пріоритет та асоціативність такі, як зображено в таблиці.

|  |  |  |
| --- | --- | --- |
| **Знаки операцій** | **Найменування** | **Асоціативність** |
| **\* / %** | Бінарні, мультиплікативні | Зліва направо |
| **+ –** | Бінарні, адитивні | Зліва направо |

##### відмінність між бінарними та унарними операціями додавання (+) та віднімання (–)

Операції додавання (**+**) та віднімання (–) можуть бути як бінарними, так і унарними.

Бінарні операції **+** та **–** використовуються у виразах при проведенні обчислень.

Унарні операції **+** та **–** використовуються для позначення знаку числа (додатне число або від‘ємне число).

**Приклад.**

int a, b;

a = -8; // унарна операція '-', позначає знак числа

b = +9; // унарна операція '+', b = 9

a = b - 5; // бінарна операція '-', використовується у виразі для обчислення

##### особливості використання операції % (остача від ділення

Операція **%** використовується над цілими операндами. Операція **%** дозволяє отримати остачу від ділення цілих операндів.

**Приклад.**

// Операція % - взяття остачі від ділення

#include <iostream>

#include <Windows.h>

#define \_USE\_MATH\_DEFINES

#include <cmath>

using namespace std;

int main() {

system("color F0");

int a = 8;

int b = 4, c;

c = a % b; // c = 0

cout << c << endl;

c = 12 % 35; // c = 12

cout << c << endl;

c = 35 % 12; // c = 11

cout << c << endl;

c = -5 % -3; // c = -2

cout << c << endl;

return 0;

}

##### ***Особливості використання операції / (ділення)***

Операція ділення має свої особливості, які полягають в наступному:

* якщо два операнди мають цілочисельний тип, то результат повертається цілого типу. У цьому випадку відбувається ділення націло. Остача від ділення відкидається;
* якщо один з операндів має тип з плаваючою комою, тоді результат має також тип з плаваючою комою.

**Приклад.**

// Операція ділення

int a, b;

int c;

float x;

a = 8;

b = 3;

c = a / b; // c = 2

x = a / b; // x = 2.0

x = a / (float)b; // x = 2.666667

x = 17.0 / 3; // x = 5.666667

x = 17 / 3; // x = 5.0

##### ***Особливості використання операторів інкременту та декременту в програмах на C++***

У мові C++ визначено два оператора, що здійснюють збільшення або зменшення цілочисельної величини на 1:

* оператор **++** – інкремент;
* оператор **––**  – декремент.

Ці оператори є унарними. Вони вимагають одного операнда. Ці оператори можуть розміщуватись до та після операнда.

**Оператор інкременту ++** збільшує значення операнду на 1. Наприклад, рядок

x = x + 1;

є аналогічний рядку

x++;

або

++x;

Так само, **оператор декременту —** зменшує значення операнду на 1. Наприклад, рядок

x = x – 1;

можна записати

x--;

або

--x;

##### ***Приклади застосування операторів інкременту (++) та декременту (––)***

Фрагмент коду, що пояснює роботу операторів **++** та **––**.

// оператори інкременту (++) та декременту (--)

int a, b;

a = 10;

b = a++; // b = 10; a = 11

a = 10;

b = ++a; // b = 11; a = 11

a = 10;

b = a--; // b = 10; a = 9

a = 10;

b = --a; // b = 9; a = 9

##### ***Відмінність між виразом ++x (***[***––***](https://www.bestprog.net/uk/2017/10/13/operators-of-increment-and-decrement-the-complex-assignment-operators_ua/#q02)***x) та виразом x++ (x***[***––***](https://www.bestprog.net/uk/2017/10/13/operators-of-increment-and-decrement-the-complex-assignment-operators_ua/#q02)***)***

Відмінність між префіксною та постфіксною формами операторів інкременту (**++**) та декременту ([**––**](https://www.bestprog.net/uk/2017/10/13/operators-of-increment-and-decrement-the-complex-assignment-operators_ua/#q02)) проявляється, коли ці оператори беруть участь в операції присвоювання.

Якщо вираз **++x** використовується в операторі присвоєння

y = ++x;

то він працює у такому порядку:

* спочатку значення x збільшується на 1, а потім результуюче значення присвоюється змінній y.

Якщо виконати вираз

y = x++;

то він працює у такому порядку:

* спочатку змінній y присвоюється значення x, а потім значення x збільшується на 1.

##### ***Складені оператори присвоювання, що використовуються в C++***

У мові C++ можна використовувати складені оператори присвоювання. Ці оператори є зручними, коли в програмі використовуються довгі імена змінних. У цьому випадку відпадає необхідність зайвий раз вводити довге ім’я змінної.

**Загальний вигляд** складеного оператора присвоювання наступний:

ім‘я\_змінної ***operation***= вираз;

де

* *ім‘я\_змінної* – безпосередньо ім‘я змінної, якій присвоюється значення;
* ***operation*** – одна з операцій **+**, **–**, **\***, **/**, **%**

Мова C++ підтримує такі складені оператори присвоювання:

**+=**, **-=**, **\*=**, **/=**, **%=**

##### Приклади використання складених операторів присвоювання

// складені оператори присвоювання

float x, y;

int a, b;

// +=, -=

a = 8;

b = 5;

a += b; // a = a + b = 13

b -= 4; // b = b - 4 = 1

// \*=, /=

x = 4;

y = 5;

x \*= y; // x = x \* y = 4 \* 5 = 20

y /= 2.5; // y = y / 2.5 = 2.0

// %=

b = 15;

b %= 6; // b = b % 6 = 3

##### ***Застосовання операцій інкременту та декременту до типів з плаваючою комою (float, double, long double)***

Зі змінними типів з плаваючою комою операції інкременту та декременту працюють так само, як і зі змінними цілого типу.

**Приклад.**

float x;

x = 24.5;

x--; // x = 23.5

##### застосовувати операції інкременту та декременту до символьного типу (char)

Оскільки, символьний тип char неявно відноситься до цілочисельних типів, то до змінних символьного типу можна застосовувати операції інкременту та декременту.

**Приклад.**

char c;

c = 'x';

c++; // c = 'y'

c = '6';

--c; // c = '5'

##### ***Призначення у програмі на C++ операції sizeof***

Операція sizeof призначена для визначення розміру типу даних, змінної базового типу, змінної структурного типу, числового значення, рядкового значення тощо.

Загальний вигляд операції sizeof:

sizeof(*тип\_або\_змінна*)

де

* *тип\_або\_змінна* – назва типу даних або змінної (об‘єкту), що використовується на даний момент в програмі.

##### ***Приклади використання операції sizeof для базових типів та числових значень***

У наведеному нижче фрагменті коду наведено приклад визначення розміру змінної базового типу, числового значення, рядкового значення або результату виразу.

// операція sizeof

int a;

short int b;

float x;

double y;

long double z;

int size;

size = sizeof(b); // size = 2

size = sizeof(int); // size = 4

size = sizeof(a); // size = 4

size = sizeof(x); // size = 4

size = sizeof(double); // size = 8

size = sizeof(z); // size = 8

size = sizeof(long double);

size = sizeof(bool); // size = 1

size = sizeof(true); // size = 1

size = sizeof(28); // size = 4 - як тип int

size = sizeof(9.8 + 5); // size = 8 - як тип double

size = sizeof("Hello world!"); // size = 13

size = sizeof('\n'); // size = 1

##### ***Тернарна операція ? :***

Тернарна операція **? :** може замінювати оператор [**умовного переходу if … else**](https://www.bestprog.net/uk/2017/08/02/conditional-jump-operator-if-2_ua/). Загальний вигляд операції ? : наступний:

***вираз1* ? *вираз2* : *вираз3***

де

* *вираз1* – будь-який логічний вираз, результатом якого є значення true або false;
* *вираз2* – вираз, що буде обчислений, якщо значення *вираз1* = true;
* *вираз3* – вираз, що буде обчислений, якщо значення *вираз1* = false.

##### Приклади використання операції ? :

У нижченаведеному фрагменті коду обчислюється мінімальне значення між двома змінними a та b:

// операція ? :

// мінімальне значення між двома числами

int a, b;

int min;

a = 15;

b = 8;

min = a > b ? b : a; // min = 8

**Директиви підключення ресурсів**.

Служать для підключення до тексту програми текстів *заголовних файлів* (мають стандартне розширення .h). ***Заголовний файл*** – це файл, що містить описи деяких функцій та оголошення типів даних і констант, які використовуються цими функціями. Список заголовних файлів визначається стандартом мови.

Існують два способи підключення директиви #include:

1. # include <ім'я> - підключення стандартних заголовних файлів;
2. # include “ім'я” - підключення заголовних файлів користувача.

Різниця між ними полягає в тому, де препроцесор буде шукати файли-ресурси, які необхідно підключити для повної підготовки коду програми до компіляції: якщо ім’я файлу взяте в кутові дужки <…>, то пошук буде вестись в наперед визначених каталогах; якщо ім’я взяте в лапки, то препроцесор шукає його в поточному каталозі, де міститься і основний файл компілювання.

Слід зазначити, що багато стандартних заголовних файлів має порожнє розширення, для решти - традиційно використовують розширення h. Наприклад,

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

#include < stdio.h> *// підключення засобів форматованого введення-виведення*

Основні заголовні файли, які входять до складу стандартної бібліотеки, подані в табл.1.

Таблиця 1. Основні заголовні файли стандартної бібліотеки С/С++

| *Заголовний файл* | *Вміст* |
| --- | --- |
| <iosream.h> | прототипи для функцій стандартних вхідних і вихідних потоків |
| <iomanip.h> | прототипи функцій для маніпуляторів, що дозволяють форматувати потоки даних |
| <math.h> | прототипи функцій математичної бібліотеки |
| <stddef.h> | загальні визначення типів, що використовуються в С/С++ для виконання деяких обчислень |
| <stdio.h> | прототипи для функцій вводу/виводу стандартної бібліотеки і інформація, що ними використовується |
| <stdlib.h> | прототипи функцій для перетворення чисел в текст і тексту в числа, прототипи функцій розміщення в пам’яті, генерації випадкових чисел і інших сервісних функцій |
| <time.h> | прототипи функцій і типи для функцій управління часом і датою |

Розглянемо найпростіший приклад програми на C/*С++* :

**#include <iostream> *// директива препроцесора***

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

**int main() *// головна функція***

**{ cout << "Hello,world\n"; *// виведення повідомлення на екран***

**system("pause"); *// пауза***

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

**}**

Рядок #include (включити) означає, що препроцесор перед компіляцією програми має включити в неї вміст заголовного файлу iostream.h. У цьому файлі оголошено засоби потокового введення та виведення. Без включення цього файлу ім’я cout буде невизначеним і компілятор повідомить про цю помилку.

У другому рядку розташовано інструкцію компілятору «використати простір імен std». Не пояснюючи значення слів «простір імен», скажемо лише, що простір імен std є стандартним. У сучасних системах програмування мовою С++ у ньому описано всі бібліотечні засоби «останнього покоління». Проте не всі компілятори розуміють цю інструкцію, тому її можна писати не завжди.

Частина програми

**int main()**

**{ ...**

**}**

визначає головну функцію main.

Операція << ("помістити в") пише свій перший аргумент у другий (рядок "Hello,world\n" у стандартний потік виведення cout). У даному рядку символ “\n“ є управляючим символом переходу на новий рядок.

За інструкцією system("pause"); виконання програми призупиняється і на екрані з’являється повідомлення, що треба натиснути будь-яку клавішу. Після її натискання програма завершується. Завдяки використанню управляючого символу “\n“, повідомлення виводиться в новому рядку. Якби його не було, повідомлення з’являлося б відразу після слів Hello, world!.

**Розподіл завдань між студентами**

|  | Формули | Прізвища студентів по групах | | |
| --- | --- | --- | --- | --- |
| ІПЗ-31 | ІПЗ-32 | ІПЗ-33 |
|  |  |  |  | Бондарець Олександр Володимирович |
|  |  | Балан Ілля Євгенійович |  | Бурлаченко Тимофій Андрійович |
|  |  | Барбара Владислав Андрійович | Воробйов Микита Тимурович | Гуль Даніл Олександрович |
|  |  | Богатирьов Олексій Русланович | Глушко Анна Федорівна | Дубовик Андрій Миколайович |
|  |  | Буряк Вікторія Дмитрівна | Жувак Матвій Дмитрович | Завадський Олександр Костянтинович |
|  |  | Возний Артем Вікторович | Ігонов Артем Олександрович | Капічніков Єгор Сергійович |
|  |  | Войнов Денис Олегович | Каламурза Георгій Олександрович | Касаткіна Любов Олексіївна |
|  |  | Герасімов Артем Богданович | Козаченко Артем Олександрович | Кириленко Тимур Сергійович |
|  |  | Голубенко Захар Євгенійович | Курбацький Владислав Дмитрович | Ковальчук Ілля Дмитрович |
|  |  | Гуцаленко Андрій Олексійович | Матяш Роман Ігорович | Костенков Глєб Анатолійович |
|  |  | Дзюбенко Олександр Дмитрович | Панич Назарій Володимирович | Неїзжалий Назар Вікторович |
|  |  | Коротков Даніел Сергійович | Пількевич Владислав Вікторович | Онипченко Олег Олегович |
|  |  | Логвиненко Роман Андрійович | Ракоїд Андрій Ігорович | Романенко Артемій Євгенович |
|  |  | Подвойський Микита Сергійович | Цюкало Володимир Володимирович | Свідерок Олександр Олександрович |
|  |  | Самойлюк Володимир Романович | Шевченко Ростислав Андрійович | Семенова Валерія Сергіївна |
|  |  | Терещенко Ярина Тарасівна | Шульга Матвій Максимович | Стариченко Роман Андрійович |
|  |  | Чемер Антон Вячеславович | Ярошенко Вікторія Олександрівна | Стешенко Анастасія Сергіївна |
|  |  |  |  | Столбун Богдан Леонідович |
|  |  |  |  | Щур Діана Володимирівна |
|  |  |  |  | Ющенко Михайло Олександрович |
|  |  |  |  | Яткевич Максим Павлович |