**Практична робота №2. Елементи мови С++. Знайомство з середовищем програмування**

1. Ознайомитися з теоретичними відомостями.
2. Запустити в Інтернет-браузері он-лайн компілятор C++Shell за посиланням <http://cpp.sh/> або використати інший доступний вам компілятор С++, наприклад GBD – URL: https://www.onlinegdb.com/
3. Скласти програму, яка містить окрім головної ще 2 функції, глобальну змінну цілого типу.

Перша функція приймає ціле значення та повертає остачу від ділення глобальної змінної на отримане параметром значення.

Друга функція приймає ціле значення, введене в головній програмі та збільшує його, використовуючи операцію інкремента.

В головній функції оголошені 2 змінні: ціла та дійсна; значення цілої вводиться, значення дійсної задається ініціалізацією. Дійсна змінна розраховується як підсумок двох функцій. Виводяться значення змінних до виклику функцій і після нього.

Перший рядок програми повинен містити коментар з номером ЛР, номером групи та прізвищем студента.

1. Програма компілюється, результати перевіряються на правильність.
2. Результати 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><Прізвищеанглійською>**

Наприклад, 31-01Ivanov.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 ПР.

**Строк відсилки ЛР для МІВТ/ ЕТ-41 24.02.2024**

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

В темі листа вказати, номер групи, прізвище студента та номер ПР як "ПР№8" 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)

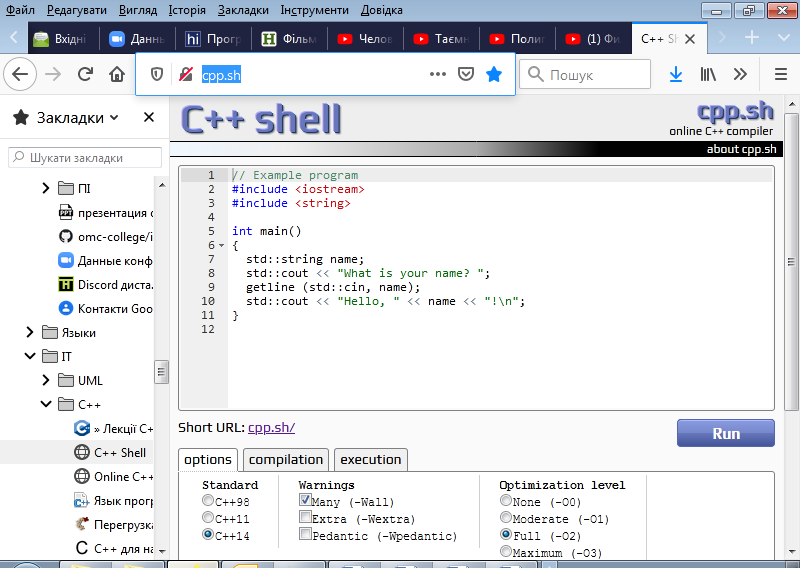


Рисунок 1 – вікно он-лайн компілятору C++SHELL

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

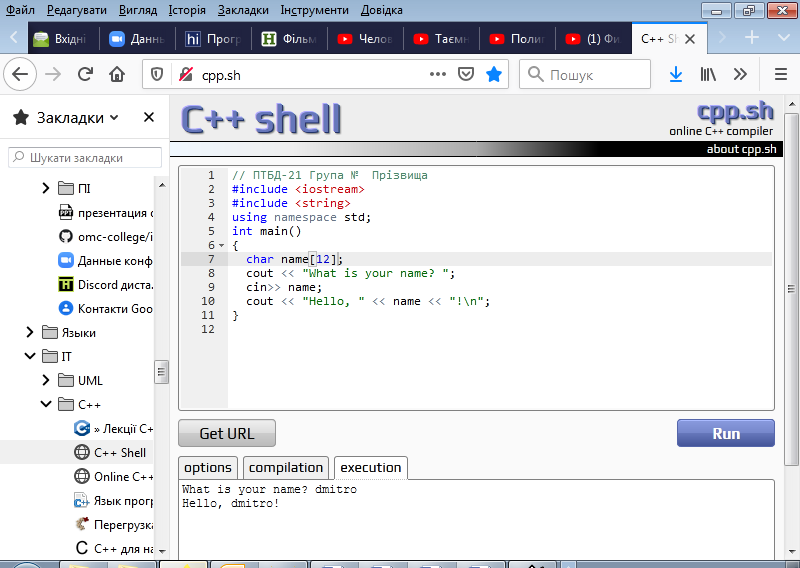


Рисунок 2 – текст зміненої програми

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

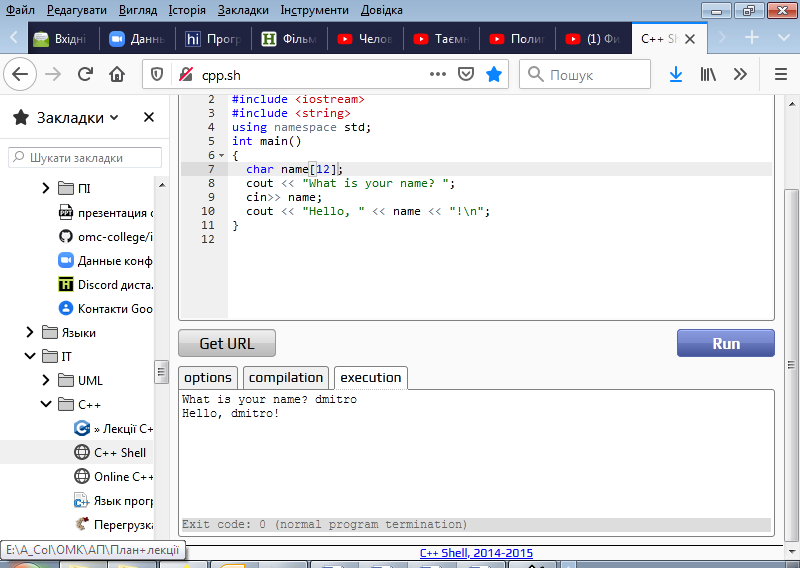


Рисунок 3 – виконання програми

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:

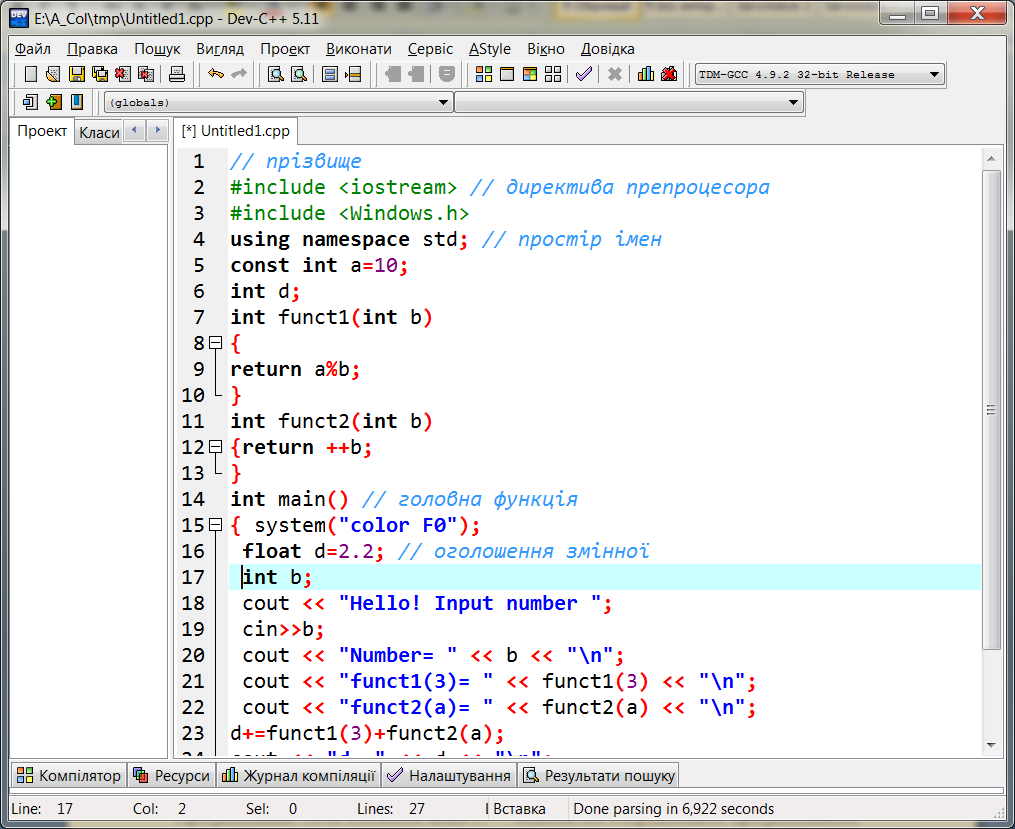
**https://sourceforge.net/projects/orwelldevcpp/**

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++

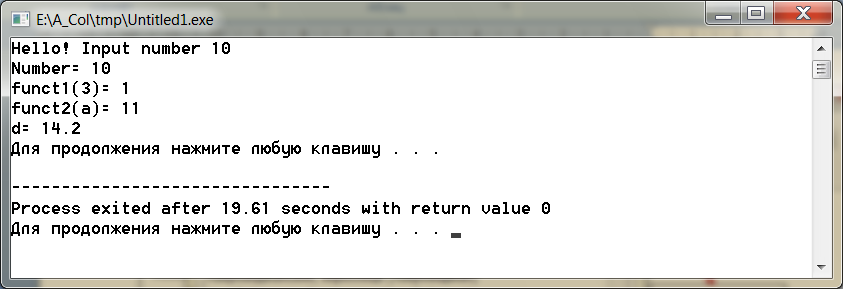
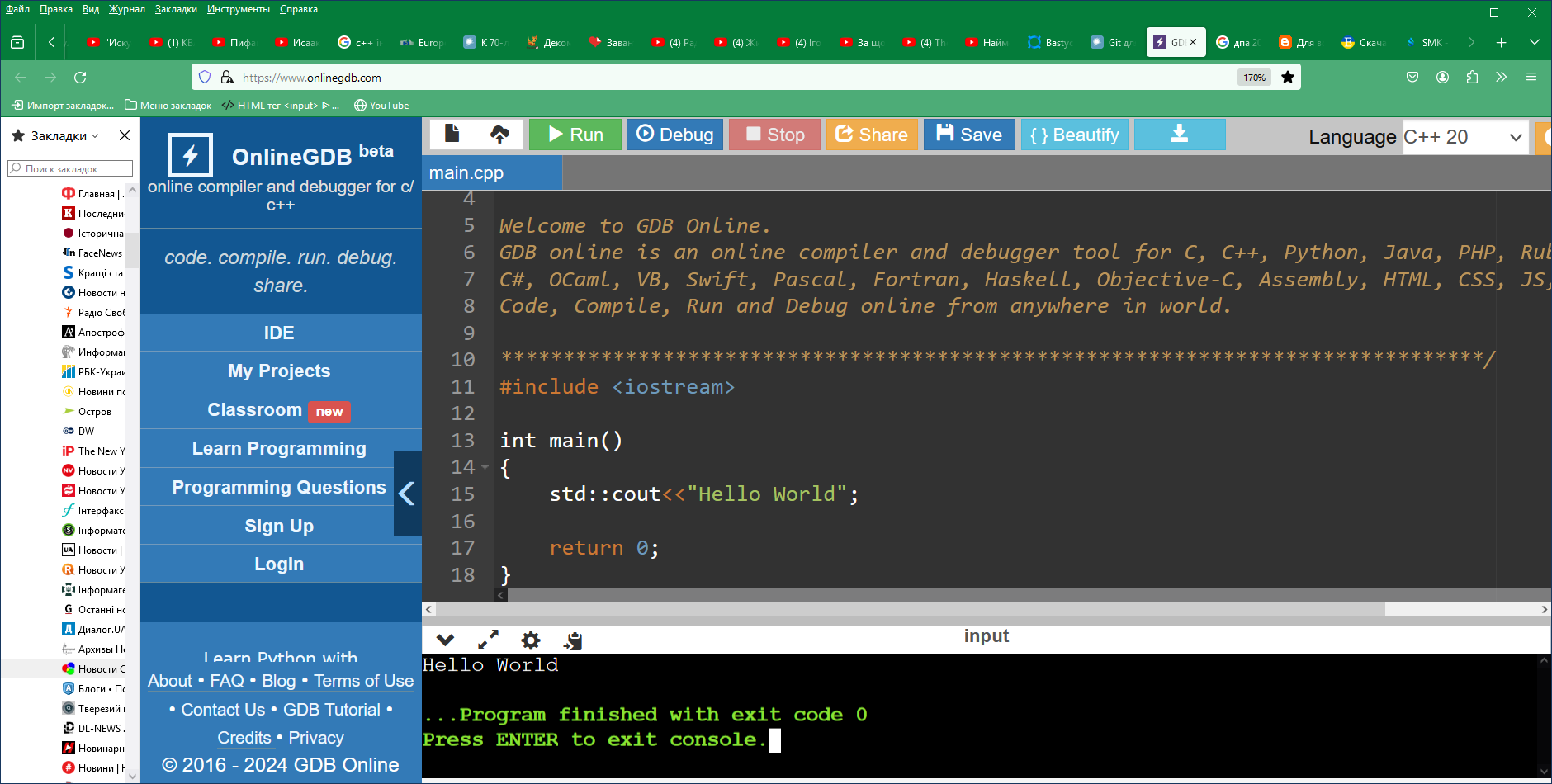


Рисунок 5 – екран виконання

Можна також використати безкоштовний он-лайн компілятор **GBD** за посиланням https://www.onlinegdb.com/



Виклик

виконання

**Теоретичні відомості**

**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!.

**Функції**

**Функція** — це іменована логічно завершена сукупність оголошень і операторів, призначених для виконання певної задачі

**Структура функції**

Усі функції мають однакову структуру визначення у вигляді:

**[тип результату] ім'я функції ([список формальних аргументів])**

**{ // *тело функции***

**опис даних;**

**оператори;**

**[return] [вираз]; };**

**Тип результату** — будь-який базовий або раніше описаний тип значення (за винятком масиву і функції), що повертається функцією (необов’язковий параметр). За відсутності цього параметра тип результату за замовчуванням буде цілий (**int**). Він також може бути описаний ключовим словом (**void**), тоді функція не повертає ніякого значення.

Більш детально функції будуть розглядатися в подальших лекціях.

**Області видимості змінних**

При оголошенні змінних у програмі велике значення має те місце, де вони оголошені. Від того, де оголошена змінна, залежать можливість її використання.

У С++ можливі **три місця оголошення змінних.**

По-перше, поза будь-яких функцій, у тому числі і головної функції main(). Така змінна називається **глобальною** і є **видимою** від місця оголошення **до кінця файлу**.

По-друге, змінна може бути оголошена всередині блоку (пригадаємо, що блок – частина тексту програми поміщена між {}), у тому числі й всередині тіла функції. Оголошена в такий спосіб змінна називається **локальною** і є **видимою до кінця блоку**. Така змінна поза блоком, у якому вона оголошена, невідома.

По-третє, змінна може бути оголошена як параметр функції. Крім спеціального призначення, а саме для передачі даних у функцію, параметр по відношенню до функції слід розглядати як її локальну змінну.

Таким чином, **в С++ існують 2 області видимості (2 контексти видимості) змінних: блок і файл.**

Якщо глобальна й локальна змінні мають одне і теж ім'я, тоді вважається, що оголошені дві різні змінні зі своїми областями використання. При цьому локальна змінна буде видима в тій функції, де вона оголошена, а глобальна у всій програмі за виключенням функції, у якій оголошена локальна змінна.

**Час життя об'єкта даних**

Об'єкт створюється, коли зустрічається його визначення і знищується, коли його ім'я виходить із області видимості.

Оголошення використовуються для визначення інтерпретації, що надається кожному ідентифікатору (імені). Це означає, що потрібно задати тип ідентифікатора щоб повідомити компілятор, до якого виду об'єктів відноситься ім'я.

Глобальні змінні створюються та ініціалізуються (тільки) один раз й "живуть" до завершення програми (**час життя глобальної змінної – до кінця роботи програми**). Об'єкти, визначені описом із ключовим словом static, поводяться так само.

Не ініціалізована явно статична (static) змінна неявно ініціалізується нулем.

**Час життя локальної змінної – до виходу з блоку**.

Часом життя і областю видимості змінних можна керувати **двома шляхами:**

1) місцем оголошення змінної у програмі;

2) використанням модифікаторів auto, register, static, extern***.***

Автоматична(auto)змінна або константа має локальнуобласть дії і відома тільки всередині блоку, у якому вона визначена. Для автоматичної змінної виділяється тимчасова пам'ять (стек). Пам'ять виділяється при вході в блок. При виході з цього блоку пам'ять, виділена для змінної, стає знову вільною. Якщо специфікатор класу пам'яті не визначений, то зміна вважається автоматичною.

Регістрова(register) змінна відрізняється від автоматичної лише пам'яттю, що виділяється для її збереження. Регістрова змінна зберігається в регістрі процесора, і, відповідно, доступ до цієї змінної набагато швидший, ніж до змінної, яка зберігається в оперативній пам'яті (auto). У випадку відсутності вільних регістрів регістрова змінна стає автоматичною.

Зовнішня(extern) змінна є глобальною змінною. Специфікатор externінформує компілятор, що змінна буде оголошена (без extern)в іншому файлі, де їй і буде виділена пам'ять.

Статичній (static) змінній (константі) пам'ять виділяється після її оголошення і зберігається до кінця виконання програми. Статичні змінні при оголошенні ініціалізуються нульовими (логічні, цілі і дійсні) порожніми значеннями.

Проілюструємо це на прикладі, що містить блоки, змінні та функції.

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

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

**const int a=20;**

**int d;**

**int funct1(float a)**

**{int b=10;**

**b+=a;**

**cout << "funct1 b="<<b<<'\n';**

**return b;**

**}**

**void funct2(int b)**

**{static int g=0;**

**b+=a;**

**d=b++;**

**g+=d;**

**cout << "funct2 b="<<b<<" g="<<g<<'\n';**

**}**

**int main() // головна функція**

**{ float d=2.2, g;**

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

**{ int a; // оголошення змінної**

**cout << "Hello! Input number ";**

**cin>>a;**

**cout << "Number= " << a << "\n";**

**};**

**cout << "a= " << a << "\n";**

**funct2(20);**

**funct2(20);**

**g= funct1(d);**

**cout << "g= " << g << "\n";**

**system("pause"); // пауза**

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

**}**

**Результат роботи програми:**

**Hello,world**

**Hello! Input number 25**

**Number= 25**

**a= 20**

**funct2 b=41 g=40**

**funct2 b=41 g=80**

**funct1 b=12**

**g= 12**