**ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ № 1**

**по курсу " Об’єктно-орієнтоване програмування"**

кафедра математичного забезпечення ЕОМ, ДНУ ім. О. Гончара

20\_\_/20\_\_ н.р.

**Тема**: "Інтегроване середовище розробки

(Visual Studio)"

Налагодження тривіальної програми

1. Постановка задачі

Запустити програму "Hello, World!" на C++ у режимі налагодження, використовуючи можливості покрокового виконання та роботи з точками зупинки.

2. Опис розв’язку

Основні кроки для запуску програми у покроковому режимі:

* додавання точок зупинки: клік лівої панелі IDE біля початку рядка з кодом, дивись рисунок 1.1.

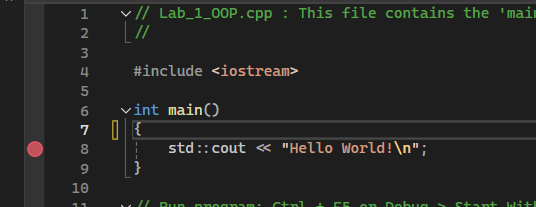


Рисунок 1.1 – Додавання точок зупинки

* умовні точки зупинки: задаємо умови в IDE, дивись рисунок 1.2.

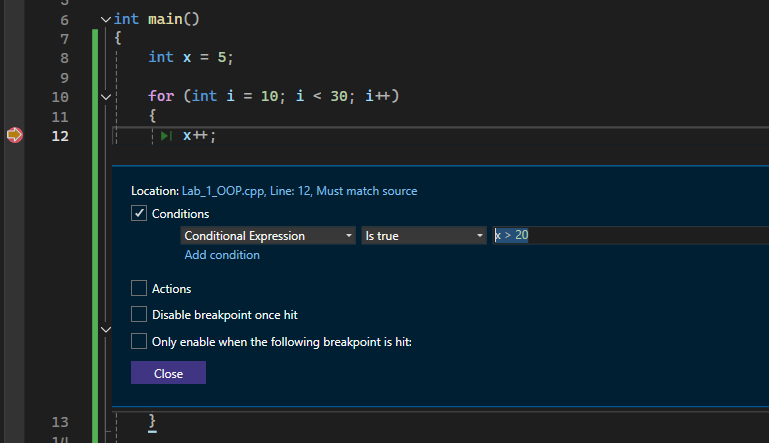


Рисунок 1.2 – Умовні точки зупинки

3. Вихідний текст програми розв’язку задачі

Лістинг 1.1 – Головний текст програми

#include <iostream>

int main()

{

std::cout << "Hello World!" << std::endl;

}

4. Опис інтерфейсу

Інтерфейс програми показує вивід "Hello World!" на екран, дивись рисунок 1.3.

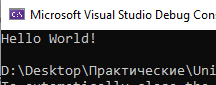


Рисунок 1.3 – Інтерфейс програми

Налаштування інтегрованого середовища

1. Постановка задачі

Налаштувати IDE для роботи з C++ проєктами, змінити параметри середовища та проєкту.

2. Опис розв’язку

Основні кроки:

* створення нового проєкту в IDE.
* збереження проєкту в бажану папку.
* зміна параметрів середовища: кольори, клавіатурні комбінації, дивись рисунок 1.4.

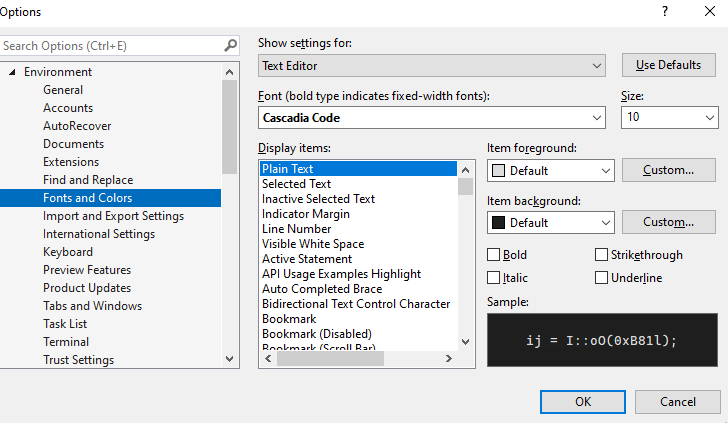


Рисунок 1.4 – Налаштування кольорів та шрифтів

* зміна параметрів проєкту: ключі компілятора або параметри командного рядка.

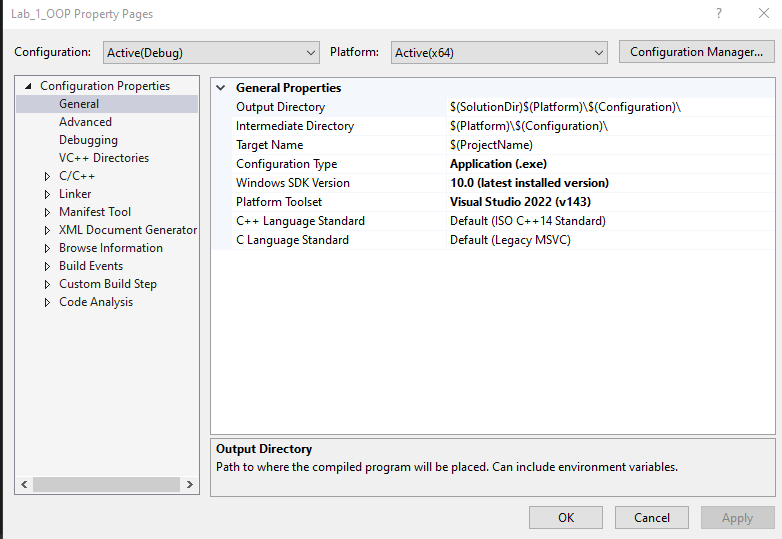


Рисунок 1.5 – Налаштування змінних проєкту

Багатомодульні програми

1. Постановка задачі

Скласти програму з кількома модулями, яка викликає функції з різних файлів.

2. Опис розв’язку

Програма складається з кількох файлів:

* main.cpp – основний файл.
* AdditionalModule.cpp – окремий файл з функціями.

3. Вихідний текст програми розв’язку задачі

Лістинг 1.2 – Головний текст програми

#include <iostream>

#include "AdditionalModule.cpp"

int main()

{

std::cout << GetGreetingText();

}

Лістинг 1.3 –Текст програми AdditionalModule.cpp

#include <iostream>

char\* GetGreetingText()

{

return (char\*)"Hello World!";

}

4. Опис тестових прикладів

Приклад програми наведено на рисунку 1.6



Рисунок 1.6 – Приклад програми

Проєктування ієрархії та структури класів

1. Постановка задачі

Створити ієрархію класів на основі будь-якої предметної галузі.

2. Опис розв’язку

Для проєктування ієрархії класів оберемо предметну галузь транспортних засобыв.

Базовий клас: Vehicle.

Наслідувані класи: Car, Bike.

1. Vehicle:

* void showDetails — чистий віртуальній метод, призначений для виведення інформації про транспорт.
* ~Vehicle — віртуальний деструктор.

1. Car:

* void showDetails() — перевизначення виведення інформації про транспорт.
* Додаткові поля: std::string brand, int numberOfDoors.

1. Bike:

* void showDetails() — перевизначення виведення інформації про транспорт.
* Додаткові поля: std::string type.

3. Вихідний текст програми розв’язку задачі

Лістинг 1.4 –Текст програми Vehicle.h

#pragma once

class Vehicle

{

public:

virtual ~Vehicle() = default;

virtual void showDetails() const = 0;

};

Лістинг 1.5 –Текст програми Car.h

#pragma once

#include <iostream>

#include "Vehicle.h"

class Car : public Vehicle {

private:

std::string brand;

int numberOfDoors;

public:

Car(const std::string& brand, int doors)

: brand(brand), numberOfDoors(doors) {}

void showDetails() const override {

std::cout << "Car: " << brand

<< ", Doors: " << numberOfDoors << std::endl;

}

};

Лістинг 1.6 –Текст програми Bike.h

#pragma once

#include <iostream>

#include "Vehicle.h"

class Bike : public Vehicle {

private:

std::string type;

public:

Bike(const std::string& type) : type(type) {}

void showDetails() const override {

std::cout << "Bike: " << type << std::endl;

}

};

Лістинг 1.7 –Текст програми main.cpp

#include <iostream>

#include <vector>

#include "Car.h"

#include "Bike.h"

int main()

{

std::vector<Vehicle\*> vehicles;

vehicles.emplace\_back(new Car("Toyota", 4));

vehicles.emplace\_back(new Bike("Mountain"));

for (const auto& vehicle : vehicles) {

vehicle->showDetails();

}

return 0;

}

4. Опис тестових прикладів

Приклад програми наведено на рисунку 1.7

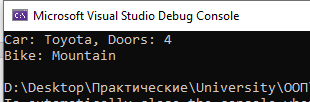


Рисунок 1.7 – Приклад програми