**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

Налагодження можна виконати за допомогою будь-якого середовища розробки, наприклад, Visual Studio або Code. Основні кроки:

* Додавання звичайних точок зупинки: кліком на лівій панелі IDE навпроти рядків коду, дивись рисунок 1.1.

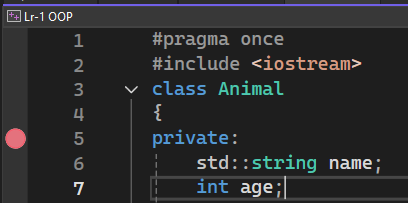


Рисунок 1.1 – Додавання звичайних точок зупинки

* Умовні точки зупинки: задаємо умови в IDE (наприклад, при значенні змінної більше певного числа), дивись рисунок 1.2.

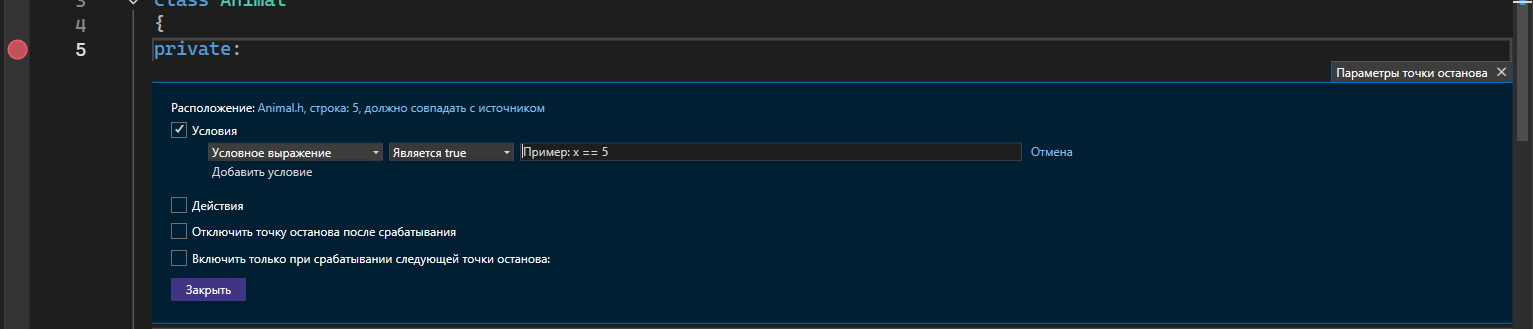


Рисунок 1.2 – Умовні точки зупинки

3. Вихідний текст програми розв’язку задачі

Лістинг 1.1 – Головний текст програми

#include <iostream>

int main()

{

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

return 0;

}

4. Опис інтерфейсу (керівництво користувача)

Інтерфейс програми показує вивід "Hello from module1!" на екран, дивись рисунок 1.3.

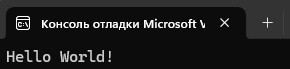


Рисунок 1.3 – Інтерфейс програми

Настроювання 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: кольорові схеми, налаштування клавіатурних комбінацій, дивись рисунок 1.4.

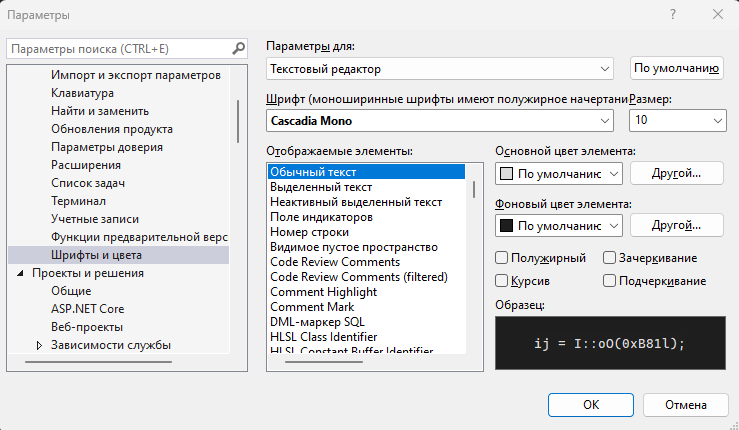


Рисунок 1.4 – Налаштування колірної схеми

* Зміна параметрів проєкту: додавання ключів компілятора або зміна параметрів запуску програми.

3. Вихідний текст програми розв’язку задачі

4. Опис інтерфейсу (керівництво користувача)

Користувач може працювати в налаштованому середовищі з новими кольоровими схемами та збереженими налаштуваннями.

5. Опис тестових прикладів

6. Аналіз помилок (опис усунення зауважень)

Багатомодульні програми

1. Постановка задачі

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

2. Опис розв’язку

Програма складається з кількох файлів:

* main.cpp – основний файл.
* module1.cpp та module1.h – окремі файли з функціями.

3. Вихідний текст програми розв’язку задачі

Лістинг 1.2 – Головний текст програми

#include <iostream>

#include "module1.h"

int main() {

sayHello();

return 0; }

Лістинг 1.3 –Текст програми module1.cpp

#include <iostream>

#include "module1.h"

void sayHello() {

std::cout << "Hello from module1!" << std::endl;

}

Лістинг 1.4 –Текст програми module1.h

#ifndef MODULE1\_H

#define MODULE1\_H

void sayHello();

#endif

4. Опис інтерфейсу (керівництво користувача)

Для запуску готового продукту потрібно виконати наступні дії:

1. Знайти виконуваний файл:

Після збірки проєкту у Visual Studio, файл .exe буде створений у папці Debug або Release, залежно від режиму збірки.

1. Запустити виконуваний файл:

Подвійним клацанням на файл <Назва\_проєкту>.exe запустити застосунок.

Щоб вийти з програми, користувач може зробити наступне:

1. Натиснути кнопку Вихід, розташовану в інтерфейсі програми.
2. Закрити вікно застосунку, натиснувши на хрестик у правому верхньому куті вікна.

5. Опис тестових прикладів

Приклад програми наведено на рисунку 1.5

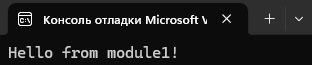


Рисунок 1.5 – Приклад програми

6. Аналіз помилок (опис усунення зауважень)

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 виберемо предметну галузь, пов’язану з тваринами в зоопарку. У цій галузі можна продемонструвати всі три основні складові ООП.

Базовий клас: Animal

1. Інкапсуляція: поля класу Animal є приватними, що забезпечує інкапсуляцію даних.
2. Наслідувані класи: Mammal, Bird та Reptile

Наслідування: класи Mammal, Bird та Reptile наслідують базовий клас Animal.

1. Mammal:
   * Методи: void sound() — перевизначення методу базового класу для видачі звуку, специфічного для ссавців.
2. Bird:

* Методи: void sound() — перевизначення для звуків птахів.
  + Додаткові поля: bool canFly.

1. Reptile:
   * Методи: void sound() — перевизначення для звуків плазунів.
   * Додаткові поля: bool isVenomous.
2. Поліморфізм: у класах-нащадках перевизначається метод sound(), щоб кожен тип тварини видавав різні звуки.
3. Вихідний текст програми розв’язку задачі

Лістинг 1.5 –Текст програми Animal.h

#pragma once

#include <iostream>

class Animal

{

private:

std::string name;

int age;

public:

Animal(const std::string& name, int age) : name(name), age(age) {}

virtual ~Animal() {} // Виртуальный деструктор

void sleep() const {

std::cout << name << " is sleeping." << std::endl;

}

virtual void sound() const = 0; // Чисто виртуальный метод

int getAge() const {

return age;

}

std::string getName() const {

return name;

}

};

Лістинг 1.6 –Текст програми Mammal.h

#pragma once

#include "Animal.h"

class Mammal :

public Animal

{

private:

bool hasFur;

public:

Mammal(std::string name, int age, bool hasFur) : Animal(name, age), hasFur(hasFur) {}

void sound() const override {

std::cout << getName() << " says: Woof!" << std::endl;

}

};

Лістинг 1.7 –Текст програми Bird.h

#pragma once

#include "Animal.h"

class Bird :

public Animal

{

private:

bool canFly;

public:

Bird(std::string name, int age, bool canFly) : Animal(name, age), canFly(canFly) {}

void sound() const override {

std::cout << getName() << " says: Tweet!" << std::endl;

}

};

Лістинг 1.8 –Текст програми Reptile.h

#pragma once

#include "Animal.h" #pragma once

#include "Animal.h"

class Reptile :

public Animal

{

private:

bool isVenomous;

public:

Reptile(std::string name, int age, bool isVenomous) : Animal(name, age), isVenomous(isVenomous) {}

void sound() const override {

std::cout << getName() << " hisses." << std::endl;

}

};

4. Опис інтерфейсу (керівництво користувача)

Користувач може запустити програму, відкривши виконуваний файл з назвою Lr-1 OOP.exe. Після запуску програми користувач побачить головний екран, на якому буде відображено наступне:

1. Програма виведе привітання "Hello from module1!", дивись рисунок 1.6



Рисунок 1.6 – Виведення привітання

1. Далі програма виведе перевантажені методи за відповідним класом, дивись рисунок 1.7.



Рисунок 1.7 – Виведення перевантажених методів

1. Після цього програма створить масив з трьох тварин різних видів та виведе звуки, які вони видають за допомогою відповідного методу sound(), дивись рисунок 1.8.

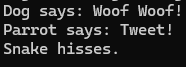


Рисунок 1.8 – Виведення різних звуків за допомогою перевизначеного методу sound()

1. Після виконання програма автоматично завершує роботу, закриваючи всі ресурси і виводячи відповідне повідомлення про успішне завершення.

Користувач не має додаткових інтерактивних елементів у цій версії програми, вона виконується автоматично після запуску. Щоб вийти з програми, користувач може або дочекатися її завершення, або натиснути "Закрити" на вікні консолі.

5. Опис тестових прикладів

Для перевірки коректності роботи програми були проведені наступні тести:

1. Було перевірено, що створюються три різні тварини: ссавець, птах і рептилія. Для кожної з них правильно викликається конструктор і ініціалізуються всі поля, дивись рисунок 1.9.

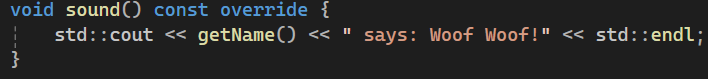


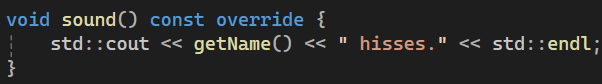




Рисунок 1.9 – Правильно викликається конструктор і ініціалізуються всі поля

1. Перевірено, що метод sound() правильно викликається для кожного з об'єктів, і на екран виводиться коректне повідомлення відповідно до типу тварини, дивись рисунок 1.10 – 1.11





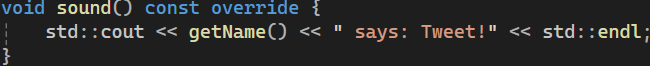


Рисунок 1.10 – Методи sound() які перевизначаються та викликаються для кожного з об'єктів



Рисунок 1.11 – Коректне повідомлення відповідно до типу тварини

1. Переконався, що після видалення тварин пам'ять звільняється коректно, програма не викликає витоків пам'яті або аварійних завершень.