

ДІАГНОСТИЧНА КОНТРОЛЬНА РОБОТА №1

Варіант 7

Студент: Вівчар Вадим Вікторович

Група: АЛК-43

1. Мета роботи

Метою роботи є ознайомлення з механізмами успадкування в мові C++, аналіз особливостей перевантаження бінарних операторів та закріплення практичних навичок роботи з потоковою системою введення-виведення під час роботи з файлами.

2. Умова завдання (варіант 7)

Пояснити, що відбувається з відкритими (public), захищеними (protected) та закритими (private) членами базового класу при public-успадкуванні.

Проаналізувати особливості перевантаження бінарних операторів.

Написати програму, яка створює файл, записує в нього число 100 у десятковій і шістнадцятковій системах числення, закриває файл, відкриває його для читання, зчитує дані та виводить їх на екран.

3. Теоретична частина

Успадкування public / protected / private

При використанні конструкції:

```
class Derived : public Base { };
```

Рівні доступу змінюються так:

- public члени залишаються public у похідному класі;
- protected члени залишаються protected;
- private члени не стають доступними у похідному класі.

Перевантаження бінарних операторів

- хоча б один операнд має бути об'єктом класу;
- реалізація можлива як метод або friend-функція;
- пріоритет та асоціативність не змінюються;
- оператор має виконувати логічну дію.

Приклад:

```
Vec operator+(const Vec& v) const {
    return Vec(x + v.x, y + v.y);
}
```

4. Практична частина

```
#include <iostream>
#include <fstream>
#include <iomanip>

int main() {
    std::ofstream outFile("data.txt");
    if (!outFile.is_open()) {
        std::cerr << "Cannot open file for writing!\n";
        return 1;
    }

    int number = 100;

    outFile << std::dec << number << ' ' << std::hex << number;
    outFile.close();

    std::ifstream inFile("data.txt");
    if (!inFile.is_open()) {
        std::cerr << "Cannot open file for reading!\n";
        return 1;
    }

    int decValue = 0;
    int hexValue = 0;

    inFile >> std::dec >> decValue;
    inFile >> std::hex >> hexValue;
    inFile.close();

    std::cout << "Decimal value: " << std::dec << decValue << '\n';
    std::cout << "Hex value (as decimal): " << std::dec << hexValue << '\n';
    std::cout << "Hex value (printed in hex): " << std::hex << hexValue << '\n';

    return 0;
}
```

5. Результати виконання програми

```
Decimal value: 100
Hex value (as decimal): 100
Hex value (printed in hex): 64

D:\std\ConsoleApplication5\x64\Debug\ConsoleApplication5.exe (процесс 5188) завершил работу с кодом 0 (0x0).
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```

6. Висновок

У ході виконання діагностичної роботи було послідовно опрацьовано теоретичні та практичні аспекти об'єктно-орієнтованого програмування. Спочатку було розглянуто механізм public-успадкування, що дозволило зрозуміти, як змінюється доступність членів класу та як забезпечується інкапсуляція даних у похідних структурах.

Далі було проаналізовано принципи перевантаження бінарних операторів. Це дало змогу усвідомити, як оператори можуть адаптуватися до роботи з власними типами даних та зберігати логічність у програмному коді.

Практична частина дозволила закріпити набуті знання шляхом написання програми з використанням файлових потоків. У процесі виконання завдання було виконано всі необхідні кроки: створення файлу, запис даних, повторне відкриття, зчитування та відображення результатів. Це сприяло розвитку навичок роботи з потоками, маніпуляторами форматів та взаємодії з файлами в C++.