Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет

ЗВІТ
ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 1
з навчальної дисципліни
“Об’єктно-орієнтоване програмування”

ОСНОВНІ ПОНЯТТЯ ООП. КЛАСИ ТА ОБ’ЄКТИ. ФУНКЦІЇ ДОСТУПУ. КОНСТРУКТОРИ І ДЕСТРУКТОРИ

ВИКОНАВ
студент академічної групи КН-24
Булюкін В.Ю

ПЕРЕВІРИВ
асистент кафедри кібербезпеки та
програмного забезпечення
Ткаченко О.С

Кропивницький – 2025

IV Варіант

Мета роботи ознайомитись з основними поняттями ООП. Вивчити поняття клас, об’єкт, сеттер, геттер та навчитись їх програмно реалізовувати мовою С++.

Завдання 1

Створіть клас Employee з приватними полями name, id, salary. Реалізуйте сетери та гетери: setName(), getName(), setId(), getId(), setSalary(), getSalary(), конструктор за замовчуванням, конструктор з параметрами та деструктор, який виводить повідомлення при видаленні об’єкта.

employee.h

```
#ifndef EMPLOYEE_H
#define EMPLOYEE_H

#include <string>
#include <windows.h>
#include <iostream>

using namespace std;

class Employee{
private:
    short id;
    string name;
    int salary;
public:
    Employee();

    Employee(short newId, string newName, int newSalary) : id(newId), 
name(newName), salary(newSalary) {};

    ~Employee();

    short getId();
    void setId(short id);


    string getName();
    void setName(string name);

    int getSalary();
    void setSalary(int salary);
};

#endif // EMPLOYEE_H
```

employee.cpp

```
#include "Employee.h"

Employee::Employee() {
    id = 0;
    name = "User";
    salary = 1000;
}

Employee::~Employee(){ cout << "\nObject Removed\n";};

short Employee::getId() { return id; }
void Employee::setId(short id) { this->id = id; }

string Employee::getName() { return name; }
void Employee::setName(string name) { this->name = name; }

int Employee::getSalary() { return salary; }
void Employee::setSalary(int salary) { this->salary = salary; }
```

main.cpp

```
#include "Employee.h"

using namespace std;

void showEmployee(Employee& emp){
    cout << "Lab01 Buliukin Volodymyr\n\n"
         << "Id: " << emp.getId() << "\n"
         << "Name: " << emp.getName() << "\n"
         << "Salary: " << emp.getSalary() << "\n";
}

void editEmployee(Employee& emp){
    short id;
    string name;
    int salary;

    cout << "\nSet new Id: ";
    cin >> id;
    emp.setId(id);

    cout << "Set new Name: ";
    cin >> name;
    emp.setName(name);

    cout << "Set new Salary: ";
    cin >> salary;
    emp.setSalary(salary);

    cout << "\nUpdated User Info:\n"
```

```
          << "Id: " << emp.getId() << endl
          << "Name: " << emp.getName() << endl
          << "Salary: " << emp.getSalary() << endl;
}

int main() {
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);

    Employee user1, user2(2, "Employee", 9999);

    showEmployee(user1);
    editEmployee(user1);

    showEmployee(user2);

    return 0;
}
```

Console Output:

```
Lab01 Buliukin Volodymyr

Id: 0
Name: User
Salary: 1000

Set new Id: 1
Set new Name: Volodymyr
Set new Salary: 10000

Updated User Info:
Id: 1
Name: Volodymyr
Salary: 10000
Lab01 Buliukin Volodymyr

Id: 2
Name: Employee
Salary: 9999

Object Removed

Object Removed
Для закрытия данного окна нажмите <ВВОД>...
```

Завдання 2

Реалізувати вище наведену задачу за допомогою структурного програмування. У висновку описати різницю цих методів.

main.cpp

```
#include <iostream>
#include <string>
#include <windows.h>

using namespace std;

struct Employee {
    short id;
    string name;
    int salary;

    Employee() {
        id = 0;
        name = "User";
        salary = 1000;
    }

    short getId(){ return id; }
    void setId(short newId){ id = newId; }

    string getName(){ return name; }
    void setName(string newName){ name = newName; }

    int getSalary(){ return salary; }
    void setSalary(int newSalary){ salary = newSalary; }
};

int main() {
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);

    Employee user1;

    cout << "User 1: Default\n"
         << "Id: " << user1.getId() << endl
         << "Name: " << user1.getName() << endl
         << "Salary: " << user1.getSalary() << endl;

    short newId;
    string newName;
    int newSalary;

    cout << "\nSet new Id: ";
    cin >> newId;
    user1.setId(newId);
```

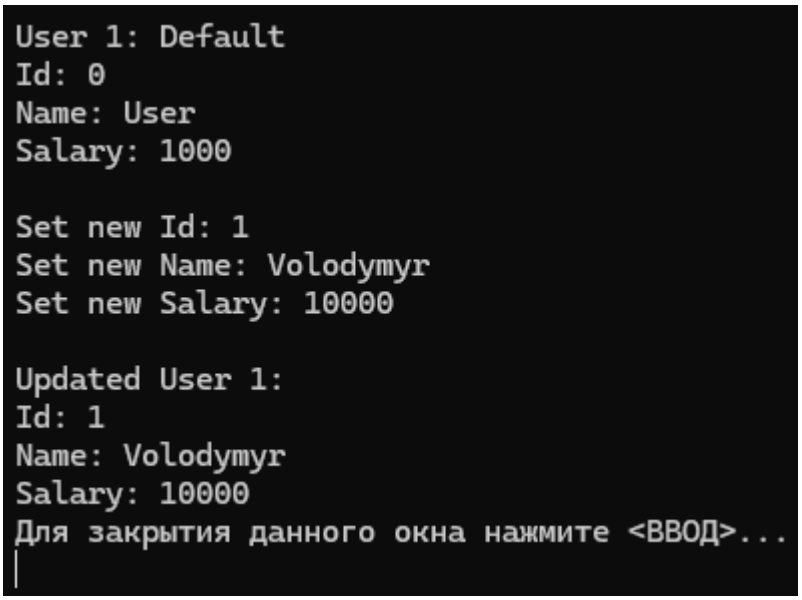
```
    cout << "Set new Name: ";
    cin >> newName;
    user1.setName(newName);

    cout << "Set new Salary: ";
    cin >> newSalary;
    user1.setSalary(newSalary);

    cout << "\nUpdated User 1:\n"
         << "Id: " << user1.getId() << endl
         << "Name: " << user1.getName() << endl
         << "Salary: " << user1.getSalary() << endl;

    return 0;
}
```

Console Output:



Висновок

Основна різниця між класом та структурою полягає у рівні доступу за замовчуванням. У класі поля оголошені як private, що забезпечує повну інкапсуляцію - доступ до даних можливий тільки через контрольовані методи (геттери та сеттери). У структурі поля за замовчуванням є public, що дозволяє прямий доступ до них.

Клас забезпечує справжню інкапсуляцію, захищає дані від некоректного використання та дозволяє валідувати вхідні значення. Структура порушує принцип інкапсуляції, оскільки дані доступні напряму, що може призвести до встановлення некоректних значень.

Контрольні запитання

1. Дайте визначення класу та об'єкта. У чому різниця між ними?

- Клас - це шаблон або опис структури та поведінки об'єктів. Він визначає набір атрибутів (полів) та методів (функцій), що характеризують певну сутність.
- Об'єкт - це конкретний екземпляр класу, створений на основі його опису.

Різниця: Клас є абстракцією (планом), а об'єкт - конкретною реалізацією цього плану з власними значеннями полів.

2. Що таке інкапсуляція і як вона реалізується в C++?

Інкапсуляція- це об'єднання даних і методів, які з ними працюють, в одну сутність (клас) з обмеженням доступу до внутрішніх деталей реалізації.

В C++ реалізується через специфікатори доступу:

- private - доступ тільки всередині класу
- protected - доступ всередині класу та його нащадків
- public - доступ звідусіль

3. Поясніть призначення специфікаторів доступу public, private та protected.

- public - члени класу доступні з будь-якого місця програми
- private - члени доступні тільки всередині самого класу
- protected - члени доступні всередині класу та його похідних класів (при наслідуванні)

4. Що таке сеттери (setter) та геттери (getter)? Навіщо вони потрібні?

- Геттер - метод для отримання (читання) значення приватного поля класу
- Сеттер - метод для встановлення (запису) значення приватного поля класу

Призначення:

- Контроль доступу до даних
- Можливість валідації даних перед записом
- Збереження принципу інкапсуляції
- Можливість додавання логіки при зміні/отриманні значень

5. Поясніть призначення ключового слова this в C++.

this - це вказівник на поточний об'єкт, з яким пов'язаний виклик методу.
Використовується для:

- Розв'язання конфліктів імен між параметрами методу та полями класу
- Передачі посилання на поточний об'єкт як параметра
- Створення ланцюжка викликів методів

6. Що таке конструктор? Які типи конструкторів існують в C++?

Конструктор - спеціальний метод класу, який викликається автоматично при створенні об'єкта для його ініціалізації.

Типи конструкторів:

1. Конструктор за замовчуванням - без параметрів
2. Параметризований конструктор - з параметрами
3. Конструктор копіювання - приймає об'єкт того ж класу як параметр

7. У чому різниця між конструктором за замовчуванням і параметризованим конструктором?

- Конструктор за замовчуванням - не має параметрів, ініціалізує об'єкт значеннями за замовчуванням
- Параметризований конструктор - приймає параметри для ініціалізації полів об'єкта конкретними значеннями

8. Що таке конструктор копіювання і коли він викликається?

Конструктор копіювання - конструктор, який створює новий об'єкт як копію існуючого об'єкта того ж класу.

Викликається при:

- Створенні об'єкта з ініціалізацією іншим об'єктом
- Передачі об'єкта у функцію за значенням
- Поверненні об'єкта з функції за значенням

9. Поясніть призначення деструктора.

Деструктор - спеціальний метод для очищення ресурсів при знищенні об'єкта. Має ім'я класу з префіксом ~.

Викликається автоматично при:

- Виході об'єкта з області видимості
- Явному видаленні об'єкта оператором delete
- Завершенні програми (для глобальних об'єктів)

10. Чи може клас мати декілька конструкторів?

Так, клас може мати декілька конструкторів завдяки механізму перевантаження функцій. Вони повинні відрізнятися:

- Кількістю параметрів
- Типами параметрів
- Порядком параметрів

Це дозволяє створювати об'єкти різними способами залежно від потреб.

11. Чому важливо використовувати принцип інкапсуляції при проектуванні класів?

Інкапсуляція важлива тому, що:

- Захищає дані від неконтрольованої зміни
- Дозволяє змінювати внутрішню реалізацію без впливу на зовнішній код
- Забезпечує цілісність даних через валідацію
- Спрощує використання класу, приховуючи складність реалізації
- Полегшує підтримку та розвиток коду

12. Що відбудеться, якщо не визначити деструктор у класі, який динамічно виділяє пам'ять?

Якщо не визначити деструктор:

- Компілятор створить деструктор за замовчуванням
- Деструктор за замовчуванням не звільняє динамічно виділену пам'ять
- Виникне витік пам'яті (memory leak)
- Ресурси залишаються заблокованими до завершення програми