Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

Кафедра кібербезпеки та програмного забезпечення

Дисципліна: Об’єктно-орієнтоване програмування

Лабораторна робота №1

Тема: «ОСНОВНІ ПОНЯТТЯ ООП. КЛАСИ ТА ОБ’ЄКТИ. ФУНКЦІЇ ДОСТУПУ. ВКАЗІВНИК THIS»

Виконав: ст. гр. КI-23

Чепіль В.О.

Перевірив викладач:

Козірова Н. Л.

Кропивницький 2024

Тема: Основні поняття ООП. Класи та об’єкти. Функції доступу. Вказівник this.

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

**Варіант№4**

**Завдання 1**

1. Створіть клас "Employee" (співробітник), який має наступні властивості та функціональність:

Приватні поля класу:

• name (ім'я співробітника)

• id (ідентифікатор співробітника)

• salary (заробітна плата співробітника)

Публічні методи класу:

• Метод setName(), який дозволяє задати ім'я співробітника.

• Метод getName(), який повертає ім'я співробітника.

• Метод setId(), який дозволяє задати ідентифікатор співробітника.

• Метод getId(), який повертає ідентифікатор співробітника.

• Метод setSalary(), який дозволяє задати заробітну плату співробітника.

• Метод getSalary(), який повертає заробітну плату співробітника.

2. Створіть об'єкт класу "Employee".

3. Задайте значення полів об'єкта за допомогою відповідних методів.

4. Виведіть інформацію про співробітника на екран, використовуючи методи для отримання значень полів.

5. Реалізувати програму за допомогою роздільної компіляції. У вашому рішенні можуть бути додаткові методи та поля, якщо ви вважаєте їх необхідними.

**Завдання 2**

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

**Лістинг до завдання 1:**

Fileemployee.h:

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

#include <string>

using namespace std;

class Employee

{

public:

Employee();

void setName(string newName);

string getName();

void setId(int newId);

int getId();

void setSalary(int newSalary);

int getSalary();

private:

string name;

int id;

int salary;

};

#endif // EMPLOYEE\_H

Fileemployee.cpp:

#include "employee.h"

Employee::Employee() {}

void Employee::setName(string newName)

{

name = newName;

}

string Employee::getName()

{

return name;

}

void Employee::setId(int newId)

{

id = newId;

}

int Employee::getId()

{

return id;

}

void Employee::setSalary(int newSalary)

{

salary = newSalary;

}

int Employee::getSalary()

{

return salary;

}

File main.cpp:

#include <iostream>

#include <Employee.h>

using namespace std;

int main()

{

Employee Human;

Human.setName("Taras");

Human.setId(87891561);

Human.setSalary(15000);

cout << "Name human: " << Human.getName() << endl;

cout << "Id human: " << Human.getId() << endl;

cout << "Salary human: " << Human.getSalary() << "$" << endl;

return 0;

}

**Лістинг до завдання 2:**

Fileemployee.h:

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

#include <string>

using namespace std;

struct Employee {

string name = "\0";

int id = -1;

int salary = -1;

};

void setName(Employee &employee, const string &newName);

string getName(const Employee &employee);

void setId(Employee &employee, int newId);

int getId(const Employee &employee);

void setSalary(Employee &employee, int newSalary);

int getSalary(const Employee &employee);

#endif // EMPLOYEE\_H

Fileemployee.cpp:

#include "employee.h"

void setName(Employee &employee, const string &newName)

{

employee.name = newName;

}

string getName(const Employee &employee)

{

return employee.name;

}

void setId(Employee &employee, int newId)

{

employee.id = newId;

}

int getId(const Employee &employee)

{

return employee.id;

}

void setSalary(Employee &employee, int newSalary)

{

employee.salary = newSalary;

}

int getSalary(const Employee &employee)

{

return employee.salary;

}

File main.cpp:

#include <iostream>

#include <employee.h>

using namespace std;

int main()

{

Employee Human;

setName(Human, "Taras");

setId(Human, 87891561);

setSalary(Human, 15000);

cout << "Name human: " << getName(Human) << endl;

cout << "Id human: " << getId(Human) << endl;

cout << "Salary human: " << getSalary(Human) << "$" << endl;

return 0;

}

**Висновок:**

**Різниця між підходами**

**1. Інкапсуляція:**

* **ООП:** У класі Employee всі дані приватні, і доступ до них здійснюється через публічні методи. Це дозволяє захистити дані від несанкціонованої зміни.
* **Структурне програмування:** В структурі Employee дані зазвичай є публічними, і функції можуть безпосередньо змінювати їх, що може призвести до небажаних змін.

**2. Спадковість та Поліморфізм:**

* **ООП:** Клас може бути розширений через спадковість, а методи можуть бути перевизначені у підкласах (поліморфізм).
* **Структурне програмування:** Відсутні механізми спадковості та поліморфізму.

**3. Логічна структура:**

* **ООП:** Логіка згрупована навколо об'єктів і їхніх методів, що сприяє кращій організації коду для великих проектів.
* **Структурне програмування:** Логіка розподіляється по окремим функціям, що може бути зручніше для простих програм, але важче підтримувати в великих системах.

**4. Гнучкість та масштабованість:**

* **ООП:** Програми, написані на основі класів та об'єктів, легше масштабувати і підтримувати.
* **Структурне програмування:** Підходить для простих завдань, але для складних систем може бути менш гнучким.

Таким чином, об'єктно-орієнтоване програмування більш підходить для великих та складних проектів, тоді як структурне програмування краще для простих завдань.