

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. КОНСТРУКТОРИ І ДЕКТРУКТОРИ»

Виконав: ст. гр. КН-24

Куріщенко П. В.

Перевірив: асистент

Козірова Н. Л.

Кропивницький 2025

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

Варіант – 14 - 10 = 4

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.

У функції main() створіть об'єкт класу, задайте значення полів через сетери та виведіть інформацію про співробітника на екран за допомогою гетерів.

Програму реалізуйте з використанням роздільної компіляції (.h + .cpp).

У вашому рішенні можуть бути додаткові методи та поля, якщо ви вважаєте їх необхідними.

Лістинг .h:

```
#ifndef EMPLOYEE_H
#define EMPLOYEE_H
#include <iostream>

using namespace std;

class Employee {
private:
    string name = "";
    int id = 0;
    double salary = 0.0;
public:
    Employee() {}
    Employee(const string& n, int i, double s) : name(n), id(i), salary(s) {}

    ~Employee() { cout << "employee \"" << name << "\" was deleted\n"; }

    void setName(const string& n) { name = n; }
    void setId(int i) { id = i; }
    void setSalary(double s) { salary = s; }

    const string& getName() { return name; }
```

```

    int getId() { return id; }
    double getSalary() { return salary; }
};

#endif // EMPLOYEE_H

```

Лістинг .cpp:

```

#include "employee.h"

int main()
{
    Employee emp1;
    cout << "Employee's name - \">< emp1.getName()
        << "\\nEmployee's id - \">< emp1.getId()
        << "\\nEmployee's salary - \">< emp1.getSalary() << "\\n\\n";

    Employee emp2;
    emp2.setName("Pavlo");
    emp2.setId(1);
    emp2.setSalary(20000);
    cout << "Employee's name - \">< emp2.getName()
        << "\\nEmployee's id - \">< emp2.getId()
        << "\\nEmployee's salary - \">< emp2.getSalary() << "\\n\\n";

    Employee emp3("Artem", 2, 6.5);
    cout << "Employee's name - \">< emp3.getName()
        << "\\nEmployee's id - \">< emp3.getId()
        << "\\nEmployee's salary - \">< emp3.getSalary() << "\\n\\n";

    return 0;
}

```

Результати виконання:

```

Employee's name - ""
Employee's id - 0
Employee's salary - 0

Employee's name - "Pavlo"
Employee's id - 1
Employee's salary - 20000

Employee's name - "Artem"
Employee's id - 2
Employee's salary - 6.5

employee "Artem" was deleted
employee "Pavlo" was deleted
employee "" was deleted

```

Завдання 2

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

Лістинг .h:

```
#ifndef EMPLOYEE_H
#define EMPLOYEE_H
#include <iostream>

using namespace std;

struct Employee {
    string name = "";
    int id = 0;
    double salary = 0.0;
};

#endif // EMPLOYEE_H
```

Лістинг .cpp:

```
#include "employee.h"

int main()
{
    Employee emp = {"Vova", 3, 60000};
    cout << "Employee's name - \"\" << emp.name
        << "\\nEmployee's id - \"\" << emp.id
        << "\\nEmployee's salary - \"\" << emp.salary << "\\n\\n";

    return 0;
}
```

Результати виконання:

```
Employee's name - "Vova"
Employee's id - 3
Employee's salary - 60000
```

Висновок:

Структуроване програмування розділяє дані і функції, використовуючи структури для зберігання інформації, тоді як об'єктно-орієнтоване програмування об'єднує дані і методи в класи, дозволяючи працювати з об'єктами більш організовано та підтримуючи інкапсуляцію, спадкування і поліморфізм.