Чернівецький національний університет імені Юрія Федьковича Навчально-науковий інститут фізико-технічних та комп’ютерних наук Кафедра математичних проблем управління і кібернетики

Звіт

про виконання лабораторної роботи №6

«Основи Об’єктно – орієнтоване програмування мовою С++. Класи. Протокол класу. Конструктори та деструктори.з дисципліни «Об’єктно-орієнтоване програмування»

Виконав студент групи 141 А (1)

Книш Володимир Володимирович

Перевірив: канд.фіз.-мат. наук,

доцент Лазорик В.В.

Оцінка:

Дата захисту:

Чернівці 2024

**Варіант №7**

**Адреса репозиторію де виконувалось завдання**:

https://github.com/Volodymyr16e/lab6.git

Тема:

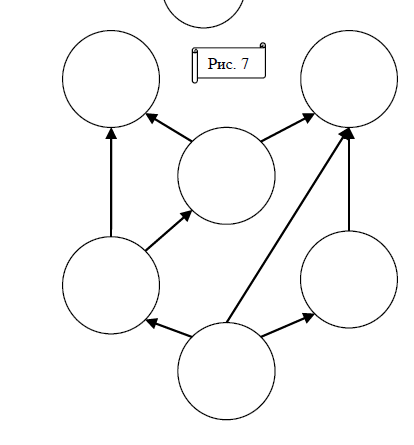
***Спадкування****. Віртуальні класи. Віртуальні функції. Абстрактні класи*

Мета: Набуття навичок в розробки класів та роботі з об’єктами класів.

**Постановка:**

**Завдання 1. Варіанти задач. Віртуальне успадкування класів класи.**

Задача. Створити дві ієрархії класів з віртуальним та без віртуального успадкуванням з елементами даних класів у кожному класі. Схема успадкування на рисунку за варіантами. Створити об’єкти похідних класів з віртуальним та без віртуального успадкуванням. Вивести розміри об’єктів даних класів.



**Код програми(1) віртуальне успадкування:**

#include <iostream>

class Base {

protected:

int dat;

public:

Base() : dat(1) {}

Base(int d) : dat(d) {}

};

class A : virtual public Base {

protected:

int a;

public:

A() : a(1) {}

A(int d) : a(d) {}

A(int d, int dt) : Base(dt), a(d) {}

};

class B : virtual public Base {

protected:

int b;

public:

B() : b(1) {}

B(int d) : b(d) {}

B(int d, int dt) : Base(dt), b(d) {}

};

class C : virtual public Base {

protected:

int c;

public:

C() : c(1) {}

C(int d) : c(d) {}

C(int d, int dt) : Base(dt), c(d) {}

};

class D : public A, public B, public C {

protected:

int d;

public:

D() : d(1) {}

D(int d) : d(d) {}

D(int a, int b, int c, int d, int dtA, int dtB, int dtC)

: A(a, dtA), B(b, dtB), C(c, dtC), d(d) {}

};

int main() {

D obj;

std::cout << "Size of Base: " << sizeof(Base) << std::endl;

std::cout << "Size of A: " << sizeof(A) << std::endl;

std::cout << "Size of B: " << sizeof(B) << std::endl;

std::cout << "Size of C: " << sizeof(C) << std::endl;

std::cout << "Size of D: " << sizeof(D) << std::endl;

return 0;

}

**Код програми(1.2) не віртуальне успадкування:**

#include <iostream>

// Базовий клас

class Base {

protected:

int dat; // елемент даних

public:

Base() : dat(1) {} // конструктор за замовчуванням

Base(int d) : dat(d) {} // конструктор з параметром

};

// Клас A, успадкований від Base з віртуальним успадкуванням

class A : virtual public Base {

protected:

int a; // елемент даних

public:

A() : a(1) {} // конструктор за замовчуванням

A(int d) : a(d) {} // конструктор з параметром

A(int d, int dt) : Base(dt), a(d) {} // конструктор з параметрами для базового і похідного класу

};

// Клас B, успадкований від Base з віртуальним успадкуванням

class B : virtual public Base {

protected:

int b; // елемент даних

public:

B() : b(1) {} // конструктор за замовчуванням

B(int d) : b(d) {} // конструктор з параметром

B(int d, int dt) : Base(dt), b(d) {} // конструктор з параметрами для базового і похідного класу

};

// Клас C, успадкований від Base з віртуальним успадкуванням

class C : virtual public Base {

protected:

int c; // елемент даних

public:

C() : c(1) {} // конструктор за замовчуванням

C(int d) : c(d) {} // конструктор з параметром

C(int d, int dt) : Base(dt), c(d) {} // конструктор з параметрами для базового і похідного класу

};

// Клас D, успадкований від A, B і C

class D : public A, public B, public C {

protected:

int d; // елемент даних

public:

D() : d(1) {} // конструктор за замовчуванням

D(int d) : d(d) {} // конструктор з параметром

D(int a, int b, int c, int d, int dtA, int dtB, int dtC)

: A(a, dtA), B(b, dtB), C(c, dtC), d(d) {} // конструктор з параметрами для всіх базових і похідного класу

};

int main() {

D obj; // створення об'єкту класу D

// Виведення розмірів об'єктів кожного класу

std::cout << "Size of Base: " << sizeof(Base) << std::endl;

std::cout << "Size of A: " << sizeof(A) << std::endl;

std::cout << "Size of B: " << sizeof(B) << std::endl;

std::cout << "Size of C: " << sizeof(C) << std::endl;

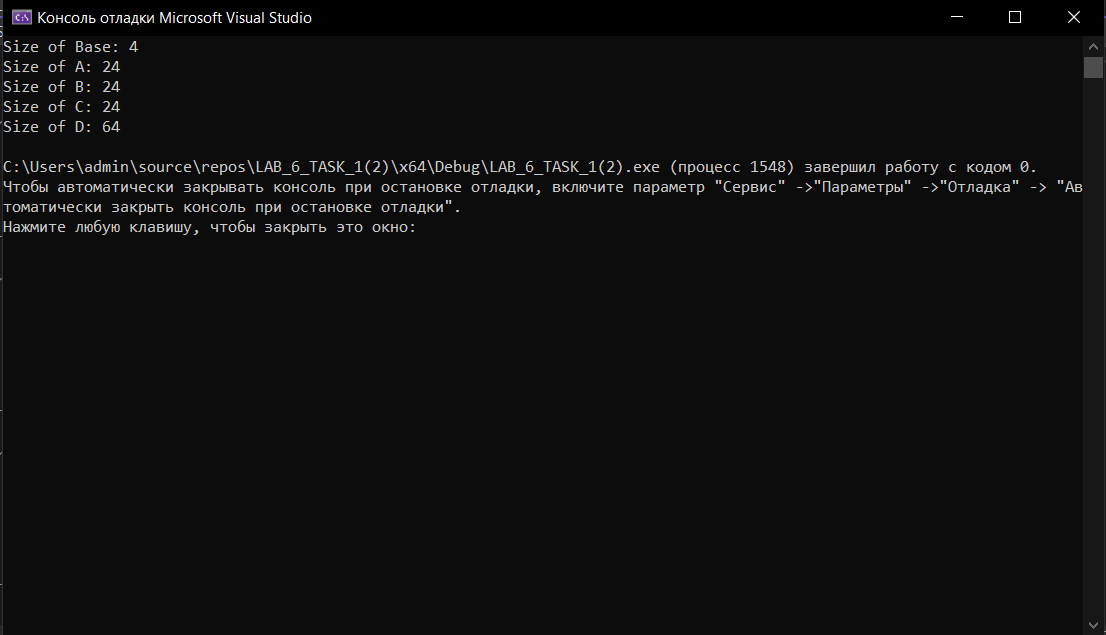
std::cout << "Size of D: " << sizeof(D) << std::endl;

return 0;

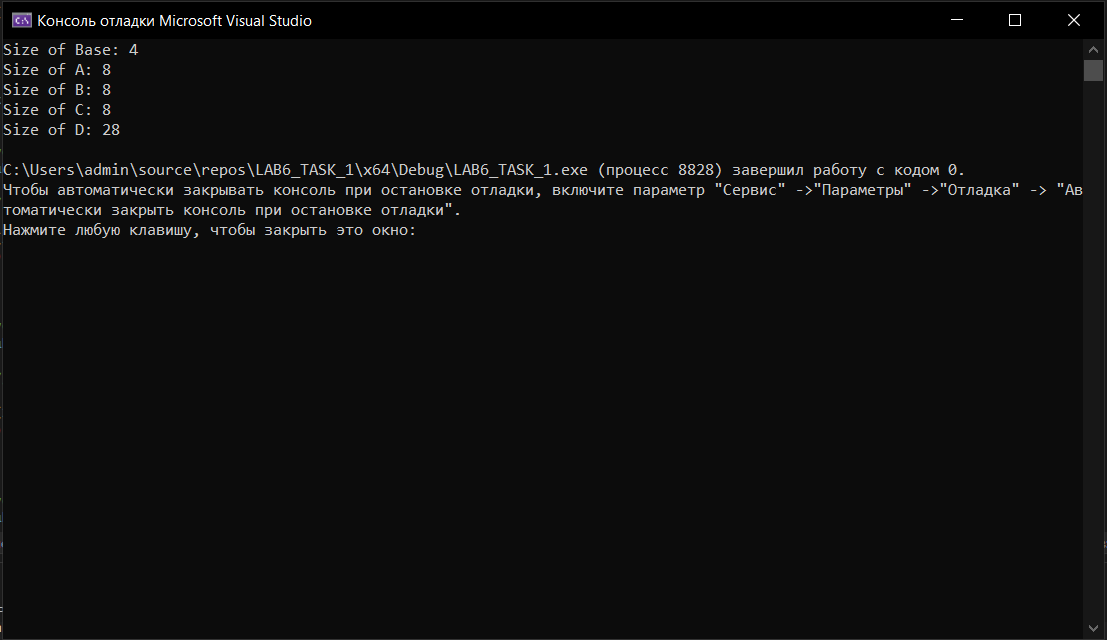
}

**Скріншоти:**

**Віртуальне успадкування:**

****

**Без віртуального успадкування:**

****

**Завдання 2. Варіанти задач. Віртуальні функції та абстрактні класи.**

**Задача 2.7.** Створити абстрактний базовий клас працівник із віртуальною функцією нарахування зарплати. Створити похідні класи: службовець із погодинною оплатою, службовець у штаті та службовець із процентною ставкою.

**Код програми(2):**

#include <iostream>

#include <string>

using namespace std;

// Абстрактний базовий клас Employee

class Employee {

protected:

string name;

public:

Employee(string name) : name(name) {}

// Чисто віртуальна функція для нарахування зарплати

virtual double calculateSalary() const = 0;

// Функція для отримання імені працівника

string getName() const {

return name;

}

// Віртуальний деструктор для правильного видалення через вказівник на базовий клас

virtual ~Employee() {}

};

// Похідний клас HourlyEmployee

class HourlyEmployee : public Employee {

private:

double hourlyRate;

double hoursWorked;

public:

HourlyEmployee(string name, double hourlyRate, double hoursWorked)

: Employee(name), hourlyRate(hourlyRate), hoursWorked(hoursWorked) {}

// Реалізація методу для нарахування зарплати для погодинно оплачуваного працівника

virtual double calculateSalary() const override {

return hourlyRate \* hoursWorked;

}

};

// Похідний клас SalariedEmployee

class SalariedEmployee : public Employee {

private:

double annualSalary;

public:

SalariedEmployee(string name, double annualSalary)

: Employee(name), annualSalary(annualSalary) {}

// Реалізація методу для нарахування зарплати для працівника з фіксованою зарплатою

virtual double calculateSalary() const override {

return annualSalary;

}

};

// Похідний клас CommissionEmployee

class CommissionEmployee : public Employee {

private:

double salesAmount;

double commissionRate;

public:

CommissionEmployee(string name, double salesAmount, double commissionRate)

: Employee(name), salesAmount(salesAmount), commissionRate(commissionRate) {}

// Реалізація методу для нарахування зарплати для працівника з відсотковою ставкою

virtual double calculateSalary() const override {

return salesAmount \* (commissionRate / 100.0);

}

};

int main() {

setlocale(LC\_CTYPE, "Ukr");

// Приклад використання класів

HourlyEmployee hourlyEmp("Василя", 15.0, 40.0); // Погодинно оплачуваний працівник

SalariedEmployee salariedEmp("Андрія", 50000.0); // Працівник з фіксованою зарплатою

CommissionEmployee commissionEmp("Івана", 10000.0, 10.0); // Працівник з відсотковою ставкою

// Виведення зарплат для кожного працівника

cout << "погодинна оплата для " << hourlyEmp.getName() << " становить " << hourlyEmp.calculateSalary() << " долларів" << endl;

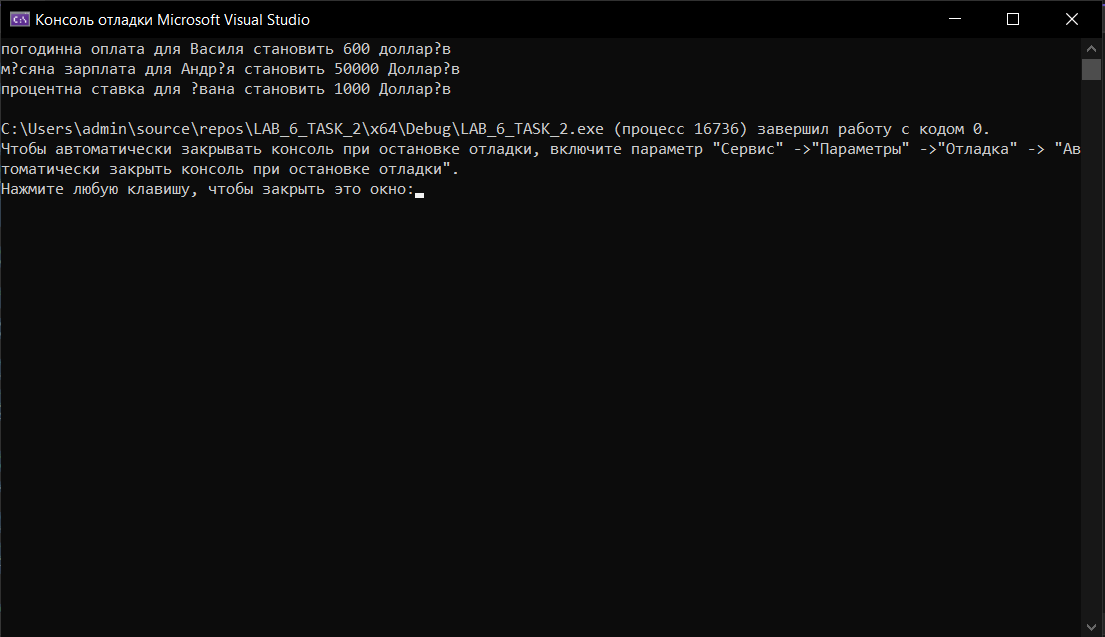
cout << "місяна зарплата для " << salariedEmp.getName() << " становить " << salariedEmp.calculateSalary() << " Долларів" << endl;

cout << "процентна ставка для " << commissionEmp.getName() << " становить " << commissionEmp.calculateSalary() << " Долларів" << endl;

return 0;

}

**Скріншоти:**

****

**Завдання 3.** Варіанти задач. Множинне спадкування. Створити ієрархію типів, що описують класи сутностей. Класи мають поля характерні сутності та успадковувати поля батьківських класів. Визначити конструктори та віртуальні деструктори, перевантажити операції введення та виведення. В похідному класі визначити виводить інформацію про об’єкт класу, що успадковане від батьківських класів. Продумати ієрархію типів таким чином щоб похідний клас успадковував двох батьків(множинне спадкування).

Задача 3.7. Ієрархія типів складається з сутностей: геометрична фігура, коло, квадрат та вписане коло в квадраті.

**Код програми(3):**

#include <iostream>

#include <cmath>

#include <locale>

using namespace std;

// Якщо M\_PI не визначений, визначимо його

#ifndef M\_PI

#define M\_PI 3.14159265358979323846

#endif

// Базовий клас Геометрична фігура

class GeometricFigure {

public:

virtual double getArea() const = 0; // Чисто віртуальна функція для обчислення площі

virtual double getPerimeter() const = 0; // Чисто віртуальна функція для обчислення периметру

virtual ~GeometricFigure() {} // Віртуальний деструктор

};

// Клас Коло, успадковується від Геометрична фігура

class Circle : public virtual GeometricFigure {

protected:

double radius;

public:

Circle(double r) : radius(r) {}

double getArea() const override {

return M\_PI \* radius \* radius;

}

double getPerimeter() const override {

return 2 \* M\_PI \* radius;

}

double getRadius() const {

return radius;

}

};

// Клас Квадрат, успадковується від Геометрична фігура

class Square : public virtual GeometricFigure {

protected:

double side;

public:

Square(double s) : side(s) {}

double getArea() const override {

return side \* side;

}

double getPerimeter() const override {

return 4 \* side;

}

double getSide() const {

return side;

}

};

// Клас Вписане коло в квадраті, успадковується від Коло та Квадрат

class InscribedCircleInSquare : public Circle, public Square {

public:

InscribedCircleInSquare(double side)

: GeometricFigure(), Circle(side / 2), Square(side) {}

// Площа вписаного кола

double getArea() const override {

return Circle::getArea();

}

// Периметр вписаного кола

double getPerimeter() const override {

return Circle::getPerimeter();

}

// Площа квадрата

double getSquareArea() const {

return Square::getArea();

}

// Периметр квадрата

double getSquarePerimeter() const {

return Square::getPerimeter();

}

};

int main() {

setlocale(LC\_CTYPE, "Ukr");

double side = 4.0;

InscribedCircleInSquare inscribedCircle(side);

cout << "Сторона квадрата: " << inscribedCircle.getSide() << endl;

cout << "Площа квадрата: " << inscribedCircle.getSquareArea() << " квадратних одиниць" << endl;

cout << "Периметр квадрата: " << inscribedCircle.getSquarePerimeter() << " одиниць" << endl;

cout << "Радіус вписаного кола: " << inscribedCircle.getRadius() << endl;

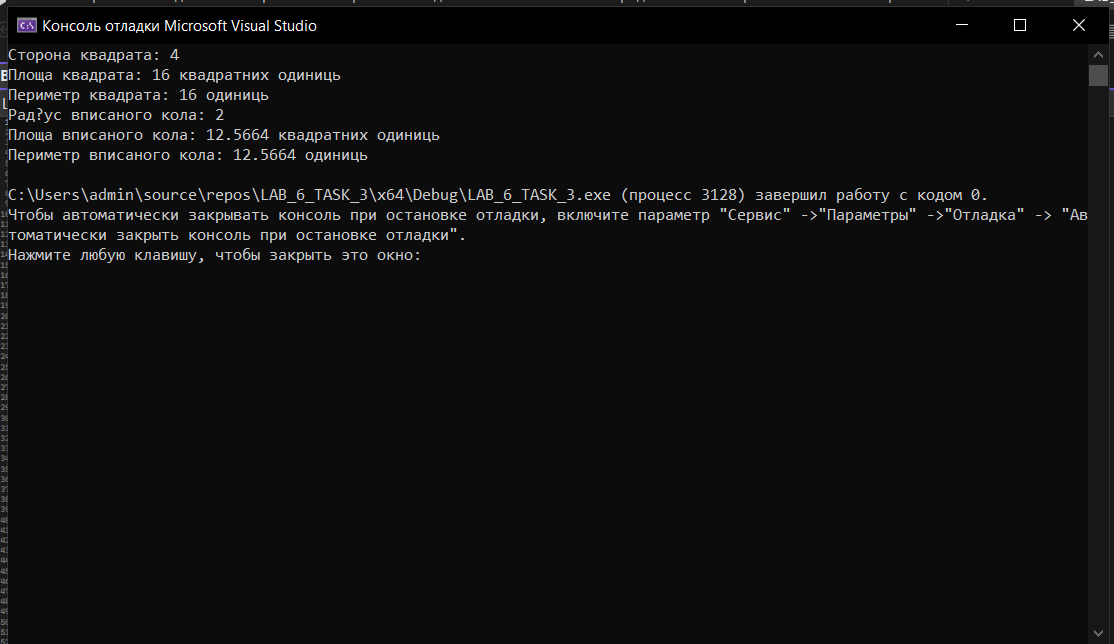
cout << "Площа вписаного кола: " << inscribedCircle.getArea() << " квадратних одиниць" << endl;

cout << "Периметр вписаного кола: " << inscribedCircle.getPerimeter() << " одиниць" << endl;

return 0;

}

**Скріншоти:**

****

**Висновок:**

Під час виконання лабораторної роботи №6 ми розглянули основні принципи об’єктно-орієнтованого програмування мовою C++. Зокрема, ми вивчили спадкування, віртуальні класи, віртуальні функції та абстрактні класи. У завданні 1 ми створили ієрархії класів з віртуальним та без віртуального успадкування, що дозволило порівняти розміри об'єктів цих класів і зрозуміти особливості віртуального успадкування. У завданні 2 ми реалізували абстрактний базовий клас та похідні класи з різними методами нарахування зарплати, що продемонструвало використання віртуальних функцій. У завданні 3 ми створили ієрархію типів з множинним успадкуванням, що дозволило розглянути складніші випадки успадкування та взаємодії між класами. Виконання цієї роботи допомогло закріпити теоретичні знання та набути практичних навичок у програмуванні на C++.