МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Фізико-Технічний Інститут

Програмування

Лабораторна робота №7

Принципи створення програм на мові C++. Класи, спадкування.

Варіант 4

Виконала:

Студентка 2 курсу ФТІ
групи ФФ-41

Щербина Софія

Перевірив:

_____

Мета: отримати навички роботи з спадкуванням класів.

Робоче завдання

1. Проаналізувати умови задач. Варіанти у Табл 7.1.
2. Розробити алгоритми та створити програми розв’язання задачі згідно з номером варіанту.
3. Підготувати відповіді на контрольні питання.
4. Результати роботи оформити протоколом. Приєднати до класу і захистити.

Завдання

У відповідності з варіантом 4 спроектувати і запрограмувати клас з:

4	***Клас кіл*** **Базовий клас (коло – координати центру, радіус):** *Конструктори*: за замовчуванням, з параметрами та копіювання. *Деструктор*. *Функції:* • обчислення площі кола; • обчислення довжини кола; • зміна радіусу, координат центру; • виведення інформації про коло на екран; **Похідний клас**: коло, яке має колір.

Спроєктувати клас-нащадок і запрограмувати всі необхідні поля і методи. У похідному класі самостійно придумати і реалізувати два методі, які можна застосувати тільки до похідного класу. У програмі створити об'єкти обох класів і продемонструвати їх роботу з усіма методами (і конструкторами і деструктором).

Код реалізації завдання

```
#include <iostream>
#include <cmath>
#include <string>
using namespace std;

class Circle {
protected:
    double radius;
    double x, y;

public:
    Circle() : x(0), y(0), radius(1) {
        cout << "Default Circle created at (" << x << ", " << y <<
") with radius " << radius << endl;
    }
    Circle(double xCoord, double yCoord, double radius) : x(xCoord),
y(yCoord), radius(radius) {
        cout << "Circle created at (" << x << ", " << y << ") with
radius " << radius << endl;
    }
    Circle(const Circle& c) : x(c.x), y(c.y), radius(c.radius) {
        cout << "Circle copied from (" << c.x << ", " << c.y << ")
with radius " << c.radius << endl;
    }
    ~Circle() {
        cout << "Circle at (" << x << ", " << y << ") with radius "
<< radius << " destroyed." << endl;
    }
    void setCenter(double xCoord, double yCoord) {
        x = xCoord;
        y = yCoord;
    }
    void setRadius(double r) {
        radius = r;
    }
    double area() const {
        return M_PI * radius * radius;
    }
    double length() {
        return 2 * M_PI * radius;
    }
    virtual void display() {
        cout << "Circle at (" << x << ", " << y << ") with radius "
<< radius << endl;
    }
};

class ColoredCircle : public Circle {
private:
    string color;
public:
    ColoredCircle() : Circle(), color("white") {
        cout << "Default ColoredCircle created with color " << color
<< endl;
    }
    ColoredCircle(double xCoord, double yCoord, double radius,
string c)
        : Circle(xCoord, yCoord, radius), color(c) {
        cout << "ColoredCircle created at (" << x << ", " << y << ")
with radius " << radius << " and color " << color << endl;
    }
    ColoredCircle(const ColoredCircle& cc)
```

```
        : Circle(cc), color(cc.color) {
        cout << "ColoredCircle copied from (" << cc.x << ", " <<
cc.y << ") with radius " << cc.radius << " and color " << cc.color
<< endl;
    }
    ~ColoredCircle() {
        cout << "ColoredCircle with color " << color << "
destroyed." << endl;
    }
    void display() override {
        Circle::display();
        cout << "Color: " << color << endl;
    }
    void changeColor(string newColor) {
        color = newColor;
        cout << "Color changed to " << color << endl;
    }
    void printColoredSymbol() {
    if (color == "red")
        cout << "#FF0000" << endl;
    else if (color == "green")
        cout << "#00FF00" << endl;
    else if (color == "blue")
        cout << "#0000FF" << endl;
    else
        cout << "#FFFFFF" << color << endl;
    }
};

int main() {
 cout << " ~~~ Test base class ~~~ " << endl;
    Circle a;
    a.display();
    Circle b(2, 3, 5);
    b.display();
    Circle c(b);
    c.display();
    cout << "Area of circle b: " << b.area() << endl;
    cout << "Length of circle b: " << b.length() << endl;
    a.setCenter(10, 10);
    a.setRadius(7);
    cout << "After setters: ";
    a.display();

    cout << "\n=== DERIVED CLASS TEST ===\n\n";

    ColoredCircle f;
    f.display();

    ColoredCircle f1(3, 3, 4, "green");
    f1.display();

    ColoredCircle f2(f1);
    f2.display();

    f1.changeColor("red");
    f1.printColoredSymbol();

    f2.changeColor("blue");
    f2.printColoredSymbol();

    // cin.get();

    return 0;
}
```

Приклад реалізації коду

```
PS C:\prog_proj\lab_7> ./main.exe
 ~~~ Test base class ~~~
Default Circle created at (0, 0) with radius 1
Circle at (0, 0) with radius 1
Circle created at (2, 3) with radius 5
Circle at (2, 3) with radius 5
Circle copied from (2, 3) with radius 5
Circle at (2, 3) with radius 5
Area of circle b: 78.5398
Length of circle b: 31.4159
After setters: Circle at (10, 10) with radius 7
```

```
=== DERIVED CLASS TEST ===

Default Circle created at (0, 0) with radius 1
Default ColoredCircle created with color white
Circle at (0, 0) with radius 1
Color: white
Circle created at (3, 3) with radius 4
ColoredCircle created at (3, 3) with radius 4 and color green
Circle at (3, 3) with radius 4
Color: green
Circle copied from (3, 3) with radius 4
ColoredCircle copied from (3, 3) with radius 4 and color green
Circle at (3, 3) with radius 4
Color: green
Color changed to red
#FF0000
Color changed to blue
#0000FF
ColoredCircle with color blue destroyed.
Circle at (3, 3) with radius 4 destroyed.
ColoredCircle with color red destroyed.
Circle at (3, 3) with radius 4 destroyed.
ColoredCircle with color white destroyed.
Circle at (0, 0) with radius 1 destroyed.
Circle at (2, 3) with radius 5 destroyed.
Circle at (2, 3) with radius 5 destroyed.
Circle at (10, 10) with radius 7 destroyed.
PS C:\prog_proj\lab_7>
```

Висновки:

У ході роботи було спроєктовано та реалізовано базовий клас Circle і похідний клас ColoredCircle, що продемонструвало практичне застосування принципів об’єктно-орієнтованого програмування, зокрема інкапсуляції, успадкування, конструкторів різних типів, деструкторів та поліморфізму. У програмі створено об’єкти обох класів, протестовано всі методи, включно з унікальними функціями похідного класу, які не властиві базовому класу. Отримані результати підтверджують коректність роботи ієрархії класів та механізмів ООП під час побудови моделі кольорового кола.