**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: «Наследование»

|  |  |  |
| --- | --- | --- |
| Студент гр. 7381 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Адамов Я.В. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Жангиров Т.М. |

Санкт-Петербург

2018

**Цель работы.**

Ознакомиться с понятиями наследование, полиморфизм, абстрактный класс, изучить виртуальные функции, принцип их работы, способ организации в памяти, раннее и позднее связывания в языке С++.

**Задание.**

Вариант №1.

Необходимо спроектировать систему классов для моделирования геометрических фигур: квадрат, эллипс, правильный пятиугольник. Задание предполагает использование виртуальных функций в иерархии наследования, проектирование и использование абстрактного базового класса. Разработанные классы должны быть наследниками абстрактного класса Shape, содержащего методы для перемещения в указанные координаты, поворота на заданный угол, масштабирования на заданный коэффициент, установки и получения цвета, а также оператор вывода в поток. Необходимо также обеспечить однозначную идентификацию каждого объекта.

**Ход работы.**

Был реализован класс Color\_RGBA\_8bit для хранения цвета с альфа-каналом в формате RGBA 8-битной разрядности.

Был реализован абстрактный класс Shape, который хранит сведения о координатах фигуры, угле поворота и цвете (объект класса Color\_RGBA\_8bit) и имеет методы для перемещения фигуры, поворота и виртуальный метод для масштабирования.

Классы Square, Ellips, Regular\_Pentagon наследуются от класса Shape и используются для работы с квадратами, эллипсами и правильными пятиугольниками соответственно.

Была написана небольшая демонстрация работы с классами. Код программы представлен в приложении Б.

**UML диаграмма разработанных классов.**

UML диаграмма разработанных классов представлена в приложении А.

**Вывод.**

В ходе выполнения лабораторной работы была спроектирована система классов для работы с геометрическими фигурами. Получены навыки работы с абстрактными классами и виртуальными функциями.

**Приложение А. Тестирование.**

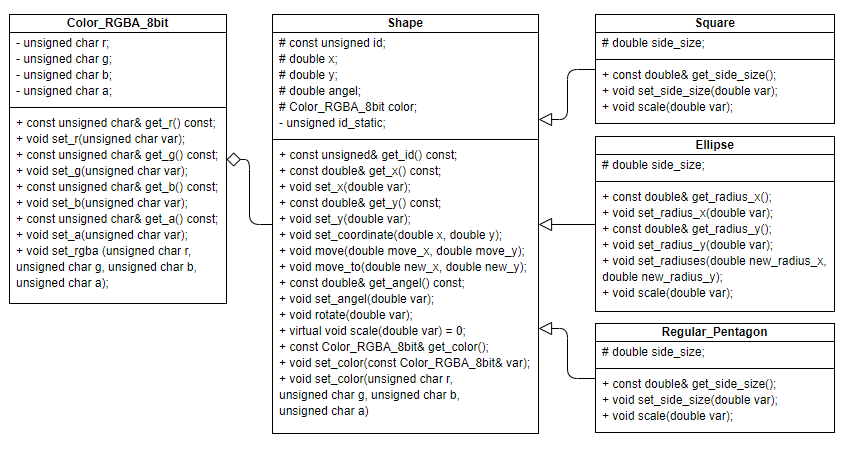
****

Рисунок 1 – UML диаграмма разработанных классов

**Приложение Б. Файл main.cpp.**

|  |
| --- |
|  |
|  | #include <iostream>  #include <cmath> |
|  |  |
|  |  |
|  |  |
|  | class Color\_RGBA\_8bit { |
|  | public: |
|  | // Constructors and destructor |
|  | Color\_RGBA\_8bit(unsigned char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a = 255) |
|  | : r(r), g(g), b(b), a(a) {} |
|  |  |
|  | Color\_RGBA\_8bit(const Color\_RGBA\_8bit& var) |
|  | : r(var.r), g(var.g), b(var.b), a(var.a) {} |
|  |  |
|  | // Methods |
|  | const unsigned char& get\_r() const noexcept { return r; } |
|  | void set\_r(unsigned char var) noexcept { r = var; } |
|  |  |
|  | const unsigned char& get\_g() const noexcept { return g; } |
|  | void set\_g(unsigned char var) noexcept { g = var; } |
|  |  |
|  | const unsigned char& get\_b() const noexcept { return b; } |
|  | void set\_b(unsigned char var) noexcept { b = var; } |
|  |  |
|  | const unsigned char& get\_a() const noexcept { return a; } |
|  | void set\_a(unsigned char var) noexcept { a = var; } |
|  |  |
|  | void set\_rgba (unsigned char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a = 0) noexcept { |
|  | this->r = r; |
|  | this->g = g; |
|  | this->b = b; |
|  | this->a = a; |
|  | } |
|  |  |
|  | // Friend functions and classes |
|  | friend std::ostream& operator<<(std::ostream&, const Color\_RGBA\_8bit&); |
|  | friend class Shape; |
|  |  |
|  | // Operators |
|  | Color\_RGBA\_8bit& operator = (const Color\_RGBA\_8bit& var) { |
|  | r = var.r; |
|  | g = var.g; |
|  | b = var.b; |
|  | a = var.a; |
|  | return \*this; |
|  | } |
|  |  |
|  | private: |
|  | unsigned char r; |
|  | unsigned char g; |
|  | unsigned char b; |
|  | unsigned char a; |
|  | }; |
|  |  |
|  | std::ostream& operator << (std::ostream& stream, const Color\_RGBA\_8bit& color) { |
|  | std::cout << "RGBA(" << unsigned(color.r) << ", " << unsigned(color.g) << ", " << unsigned(color.b) << ", " << unsigned(color.a) << ")"; |
|  | return stream; |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | class Shape { |
|  | public: |
|  | // Constructors and destructor |
|  | Shape(double x = 0, double y = 0, double angel = 0, unsigned char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a = 255) |
|  | : id(id\_static), x(x), y(y), color(r, g, b, a) { set\_angel(angel); id\_static++; } |
|  |  |
|  | Shape(double x, double y, double angel, const Color\_RGBA\_8bit& color) |
|  | : id(id\_static), x(x), y(y), angel(angel), color(color) { set\_angel(angel); id\_static++; } |
|  |  |
|  | // Methods |
|  | const unsigned& get\_id() const noexcept { return id; } |
|  |  |
|  | const double& get\_x() const noexcept { return x; } |
|  | void set\_x(double var) noexcept { x = var; } |
|  |  |
|  | const double& get\_y() const noexcept { return y; } |
|  | void set\_y(double var) noexcept { y = var; } |
|  |  |
|  | void set\_coordinate(double x, double y) noexcept { |
|  | this->x = x; |
|  | this->y = y; |
|  | } |
|  |  |
|  | void move(double move\_x = 0, double move\_y = 0) noexcept { |
|  | x += move\_x; |
|  | y += move\_y; |
|  | } |
|  |  |
|  | void move\_to(double new\_x = 0, double new\_y = 0) noexcept { |
|  | x = new\_x; |
|  | y = new\_y; |
|  | } |
|  |  |
|  | const double& get\_angel() const noexcept { return angel; } |
|  | void set\_angel(double var) noexcept { |
|  | angel = var; |
|  | while (angel < 0) |
|  | angel += 360.0; |
|  | while (angel >= 360.0) |
|  | angel -= 360; |
|  | } |
|  |  |
|  | void rotate(double var) noexcept { |
|  | set\_angel(angel + var); |
|  | } |
|  |  |
|  | virtual void scale(double var) = 0; |
|  |  |
|  | // Color |
|  | const Color\_RGBA\_8bit& get\_color() const noexcept { return color; } |
|  | void set\_color(const Color\_RGBA\_8bit& var) noexcept { color = var; } |
|  | void set\_color(unsigned char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a = 255) noexcept { color.set\_rgba(r, g, b, a); } |
|  | const unsigned char& get\_r() const noexcept { return color.r; } |
|  | void set\_r(unsigned char var) noexcept { color.r = var; } |
|  | const unsigned char& get\_g() const noexcept { return color.g; } |
|  | void set\_g(unsigned char var) noexcept { color.g = var; } |
|  | const unsigned char& get\_b() const noexcept { return color.b; } |
|  | void set\_b(unsigned char var) noexcept { color.b = var; } |
|  | const unsigned char& get\_a() const noexcept { return color.a; } |
|  | void set\_a(unsigned char var) noexcept { color.a = var; } |
|  |  |
|  | protected: |
|  | const unsigned id; |
|  | double x; |
|  | double y; |
|  | double angel; |
|  | Color\_RGBA\_8bit color; |
|  |  |
|  | private: |
|  | static unsigned id\_static; |
|  |  |
|  | }; |
|  |  |
|  | unsigned Shape::id\_static = 0; |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | class Square : public Shape { |
|  | public: |
|  | // Constructors and destructor |
|  | Square(double side\_size = 0, double x = 0, double y = 0, double angel = 0, unsigned char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a = 255) |
|  | : side\_size(side\_size < 0 ? -side\_size : side\_size), Shape(x, y, angel, r, g, b, a) {} |
|  |  |
|  | Square(double side\_size, double x, double y, double angel, const Color\_RGBA\_8bit color) |
|  | : side\_size(side\_size < 0 ? -side\_size : side\_size), Shape(x, y, angel, color) {} |
|  |  |
|  | Square(const Square& var) |
|  | : side\_size(var.side\_size), Shape(var.x, var.y, var.angel, var.color) {} |
|  |  |
|  | // Methods |
|  | const double& get\_side\_size() const noexcept { return side\_size; } |
|  | void set\_side\_size(double var) noexcept { side\_size = var < 0 ? -var : var; } |
|  |  |
|  | void scale(double var) noexcept override { |
|  | side\_size \*= var < 0 ? -var : var; |
|  | } |
|  |  |
|  | // Friend functions |
|  | friend std::ostream& operator<<(std::ostream&, const Square&); |
|  |  |
|  | protected: |
|  | double side\_size; |
|  | }; |
|  |  |
|  | std::ostream& operator << (std::ostream& stream, const Square& square) { |
|  | std::cout << "Square[id: " << square.id << ", side size: " << square.side\_size << ", coordinate(" << square.x << ", " << square.y << "), angel: " << square.angel << ", " << square.color << ']'; |
|  | return stream; |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | class Ellipse : public Shape { |
|  | public: |
|  | // Constructors and destructor |
|  | Ellipse(double radius\_x = 0, double radius\_y = 0, double x = 0, double y = 0, double angel = 0, unsigned char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a = 255) |
|  | : radius\_x(radius\_x < 0 ? -radius\_x : radius\_x), radius\_y(radius\_y < 0 ? -radius\_y : radius\_y), Shape(x, y, angel, r, g, b, a) {} |
|  |  |
|  | Ellipse(double radius\_x, double radius\_y, double x, double y, double angel, const Color\_RGBA\_8bit& color) |
|  | : radius\_x(radius\_x < 0 ? -radius\_x : radius\_x), radius\_y(radius\_y < 0 ? -radius\_y : radius\_y), Shape(x, y, angel, color) {} |
|  |  |
|  | Ellipse(const Ellipse& var) |
|  | : radius\_x(var.radius\_x), radius\_y(var.radius\_y), Shape(var.x, var.y, var.angel, var.color) {} |
|  |  |
|  | // Methods |
|  | const double& get\_radius\_x() const noexcept { return radius\_x; } |
|  | void set\_radius\_x(double var) noexcept { radius\_x = var < 0 ? -var : var; } |
|  |  |
|  | const double& get\_radius\_y() const noexcept { return radius\_y; } |
|  | void set\_radius\_y(double var) noexcept { radius\_y = var < 0 ? -var : var; } |
|  |  |
|  | void set\_radiuses(double new\_radius\_x, double new\_radius\_y){ |
|  | set\_radius\_x(new\_radius\_x); |
|  | set\_radius\_y(new\_radius\_y); |
|  | } |
|  |  |
|  | void scale(double var) noexcept override { |
|  | double temp = var < 0 ? -var : var; |
|  | radius\_x \*= temp; |
|  | radius\_y \*= temp; |
|  | } |
|  |  |
|  | // Friend functions |
|  | friend std::ostream& operator<<(std::ostream&, const Ellipse&); |
|  |  |
|  | protected: |
|  | double radius\_x; |
|  | double radius\_y; |
|  | }; |
|  |  |
|  | std::ostream& operator << (std::ostream& stream, const Ellipse& ellipse) { |
|  | std::cout << "Ellipse[id: " << ellipse.id << ", radius x: " << ellipse.radius\_x << ", radius y: " << ellipse.radius\_y << ", coordinate(" << ellipse.x << ", " << ellipse.y << "), angel: " << ellipse.angel << ", " << ellipse.color << ']'; |
|  | return stream; |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | class Regular\_Pentagon : public Shape { |
|  | public: |
|  | // Constructors and destructor |
|  | Regular\_Pentagon(double side\_size = 0, double x = 0, double y = 0, double angel = 0, unsigned char r = 0, unsigned char g = 0, unsigned char b = 0, unsigned char a = 255) |
|  | : side\_size(side\_size < 0 ? -side\_size : side\_size), Shape(x, y, angel, r, g, b, a) {} |
|  |  |
|  | Regular\_Pentagon(double side\_size, double x, double y, double angel, const Color\_RGBA\_8bit& color) |
|  | : side\_size(side\_size < 0 ? -side\_size : side\_size), Shape(x, y, angel, color) {} |
|  |  |
|  | Regular\_Pentagon(const Regular\_Pentagon& var) |
|  | : side\_size(var.side\_size), Shape(var.x, var.y, var.angel, var.color) {} |
|  |  |
|  | // Methods |
|  | const double& get\_side\_size() const noexcept { return side\_size; } |
|  | void set\_side\_size(double var) noexcept { side\_size = var < 0 ? -var : var; } |
|  |  |
|  | void scale(double var) noexcept override { |
|  | side\_size \*= var < 0 ? -var : var; |
|  | } |
|  |  |
|  | // Friend functions |
|  | friend std::ostream& operator<<(std::ostream&, const Regular\_Pentagon&); |
|  |  |
|  | protected: |
|  | double side\_size; |
|  | }; |
|  |  |
|  | std::ostream& operator << (std::ostream& stream, const Regular\_Pentagon& pentagon) { |
|  | std::cout << "Regular\_Pentagon[id: " << pentagon.id << ", side size: " << pentagon.side\_size << ", coordinate(" << pentagon.x << ", " << pentagon.y << "), angel: " << pentagon.angel << ", " << pentagon.color << ']'; |
|  | return stream; |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | int main() { |
|  | std::cout << "Demonstration.\n" << std::endl; |
|  |  |
|  | Color\_RGBA\_8bit color\_1(45, 15, 115); |
|  | std::cout << color\_1 << "\n" << std::endl; |
|  |  |
|  | Square square\_1(5, 6.24, -56.11, 14, color\_1); |
|  | std::cout << square\_1 << std::endl; |
|  |  |
|  | Square square\_2(square\_1); |
|  | square\_2.move(415.15, -100); |
|  | square\_2.rotate(600.15); |
|  | square\_2.scale(1.5); |
|  | std::cout << square\_2 << std::endl; |
|  |  |
|  | std::cout << std::endl; |
|  |  |
|  | Ellipse ellipse\_1(25, 50, -100, -150.15, 0); |
|  | std::cout << ellipse\_1 << std::endl; |
|  |  |
|  | std::cout << std::endl; |
|  |  |
|  | Regular\_Pentagon pentagon\_1(25, 15, 15, 25.234, 54, 23, 14); |
|  | pentagon\_1.move\_to(-pentagon\_1.get\_x(), -pentagon\_1.get\_y()); |
|  | pentagon\_1.set\_angel(0); |
|  | std::cout << pentagon\_1 << std::endl; |
|  | pentagon\_1.set\_r(70); |
|  | std::cout << pentagon\_1.get\_color() << std::endl; |
|  |  |
|  |  |
|  | return 0; |
|  | } |