МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Математическое обеспечение и применение ЭВМ»

**Курсовой проект**

по дисциплине «Программирование»

на тему: «Разработка консольного приложения

с элемента»

ПГУ 09.03.04 – 02КП241. 14 ПЗ

Направление подготовки – 09.03.04 «Программная инженерия»

|  |  |
| --- | --- |
| Выполнил студент: | Мишин А.Д. |
| Группа: | 24ВП1 |
| Руководитель: |  |
| к.т.н., доцент | Гурьянов Л.В. |

|  |  |
| --- | --- |
| Проект защищен с оценкой | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Преподаватели | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата защиты | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Пенза 2025

Оглавление

[Введение 3](#_Toc198468061)

[1. РАЗРАБОТКА ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРИЛОЖЕНИЯ НА ЯЗЫКЕ С++. ПЯТИУГОЛЬНИК, ТРЕУГОЛЬНИК 4](#_Toc198468062)

[1.1. Анализ предметной области 4](#_Toc198468063)

[1.2. Анализ функциональных требований 5](#_Toc198468064)

[1.3. Проектирование 8](#_Toc198468065)

[1.4. Реализация 10](#_Toc198468066)

[1.5. Тестирование 12](#_Toc198468067)

[Заключение 15](#_Toc198468068)

[Список использованных источников 17](#_Toc198468069)

[Приложение А. Код программы 18](#_Toc198468070)

[Приложение Б. Код программы 26](#_Toc198468071)

# Введение

Данный курсовой проект посвящен разработке программных инструментов, предназначенных для создания и графического отображения базовых геометрических форм, включая квадрат, окружность, а также составной объект, представляющий собой квадрат, вписанный в окружность.

Работа над проектом разделена на ключевые этапы: изучение предметной области и формулирование требований к программному обеспечению, проектирование структуры программы, непосредственная разработка и тестирование готового продукта.

Для проектирования архитектуры программного решения применяется стандарт UML (Unified Modeling Language). Реализация проекта выполняется с использованием языка программирования C++ в среде разработки Microsoft Visual Studio 2022.

В качестве графической составляющей проекта представлены диаграммы классов и компонентов, выполненные в соответствии с UML-нотацией. Эти диаграммы позволяют наглядно отобразить структуру и взаимодействие элементов разрабатываемого программного обеспечения.

Курсовой проект содержит два приложения: Приложение А содержит код главной программы, Приложение Б – графическую часть описания этапов разработки (диаграмма классов, диаграмма компонентов).

# РАЗРАБОТКА ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРИЛОЖЕНИЯ НА ЯЗЫКЕ С++. КЛАССЫ ФИГУР: РОМБ В ОКРУЖНОСТИ.

## Анализ предметной области

Предметная область курсового проекта – геометрические фигуры (рисунок 1).

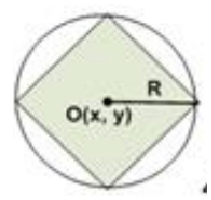


Рисунок 1 - Заданная геометрическая фигура

В данной модели предметной области (рисунок 2) представлены такие сущности, как Complex Figure (Сложная фигура), Square (Квадрат), Circle (Окружность).

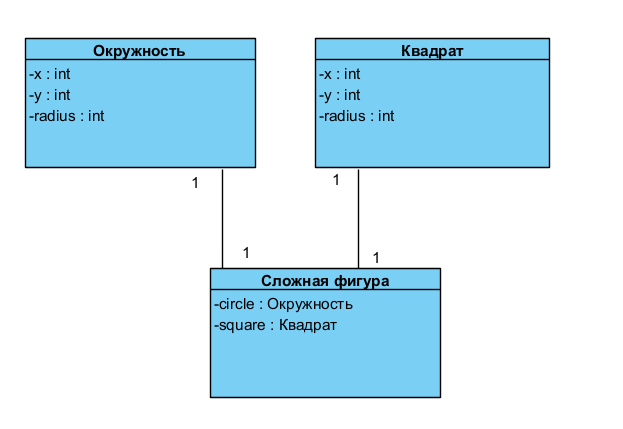


Рисунок 2 - Модель предметной области

Атрибутами сложной фигуры являются квадрат и окружность. Атрибутами окружности являются x, y, radius, где (x, y) – координаты центра окружности, radius – радиус окружности. Атрибутами квадрата являются x, y, radius, где radius – расстояние от центра квадрата до его вершин, а (x, y) – координаты точки пересечения диагоналей (центр квадрата).

Диагонали квадрата пересекаются под прямым углом, и вершины находятся на равном расстоянии от центра. Пусть центр квадрата (x, y) тогда координаты вершин можно вычислить, используя расстояние до вершин.

Вершины квадрата:

* Верхняя (x, y + d)
* Правая (x + d, y)
* Левая (x - d, y)
* Нижняя (x, y – d)

## Анализ функциональных требований

В техническом задании на курсовой проект определены следующие функциональные требования:

* создать геометрическую фигуру (квадрат, окружность, сложная фигура);
* показать фигуру;
* скрыть фигуру;
* переместить фигуру;
* добавить фигуру в контейнер хранения;
* показать фигуры из контейнера;
* удалить контейнер фигур.

Диаграмма вариантов использования, соответствующая этим требованиям, приведена на рисунке 3. На диаграмме варианты «Показать окружность» и «Спрятать окружность» расширяют функциональность варианта «Переместить окружность» (отношение «extends» – способ введения нового поведения в существующий вариант использования). Отношение расширение обосновано алгоритмом функции «Переместить окружность», который сначала рисует геометрическую окружность цветом фона, а потом рисует ее на новом месте (относительно новой точки «привязки»). Аналогичные диаграммы вариантов использования строятся для фигуры «Квадрат» и «Сложная фигура».

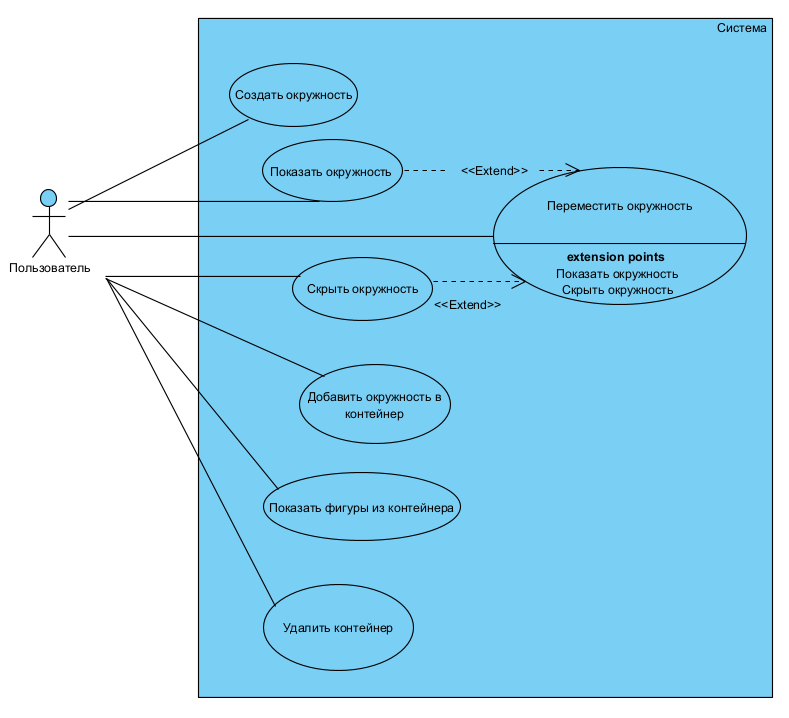


Рисунок 3 - Диаграмма вариантов использования

Сценарий варианта использования «Создать окружность» приведен в таблице 1.

Сценарии варианта использования «Переместить окружность» и расширяющих его вариантов приведены в таблицах 2 – 4.

*Таблица 1*

**Сценарий варианта использования «Создать окружность»**

|  |
| --- |
| **Наименование: Создать окружность** |
| **ID: 1** |
| **Краткое описание:** система создает окружность с центром в точке O с координатами x, y и радиусом R |
| **Действующие лица:** пользователь |
| **Предусловие:** - |
| **Основной поток:**   1. Пользователь задает атрибуты окружности (координаты центра окружности X, Y и радиус окружности R) 2. Система создает объект «Окружность» с заданными атрибутами |
| **Постусловие:** фигура создана |

*Таблица 2*

**Сценарий варианта использования «Показать окружность»**

|  |
| --- |
| **Наименование: Показать окружность** |
| **ID: 2** |
| **Краткое описание:** система рисует изображение окружности с координатами центра (x, y) и заданным радиусом (R) |
| **Действующие лица:** пользователь |
| **Предусловие:** окружность создана |
| **Основной поток:**   1. Пользователь инициирует рисование окружности с координатами вершины (x, y) и радиусом (R) 2. Система рисует изображение окружности с заданными координатами вершины (x, y) и заданным радиусом (R) |
| **Постусловие:** окружность нарисована |

*Таблица 3*

**Сценарий варианта использования «Скрыть окружность»**

|  |
| --- |
| **Наименование: Скрыть окружность** |
| **ID: 3** |
| **Краткое описание:** система удаляет изображение окружности |
| **Действующие лица:** пользователь |
| **Предусловие:** окружность создана и нарисована |
| **Основной поток:**   1. Пользователь инициирует удаление изображения окружности. 2. Система закрашивает область с окружностью в цвет фона |
| **Постусловие:** изображение окружности скрыто |

*Таблица 4*

**Сценарий варианта использования «Переместить окружность»**

|  |
| --- |
| **Наименование: Переместить окружность** |
| **ID: 4** |
| **Краткое описание:** система удаляет изображение окружности и рисует её в новых координатах |
| **Действующие лица:** пользователь |
| **Предусловие:** окружность создана и нарисован |
| **Основной поток:**   1. Пользователь задает новые координаты X, Y окружности. Точка расширения «Спрятать». 2. Система переопределяет координаты окружности. Точка расширения «Нарисовать» |
| **Постусловие:** окружность перемещена |

## Проектирование

На рисунке 4 и в Приложении Б на рисунке Б.1 приведена диаграмма классов для данной предметной области.

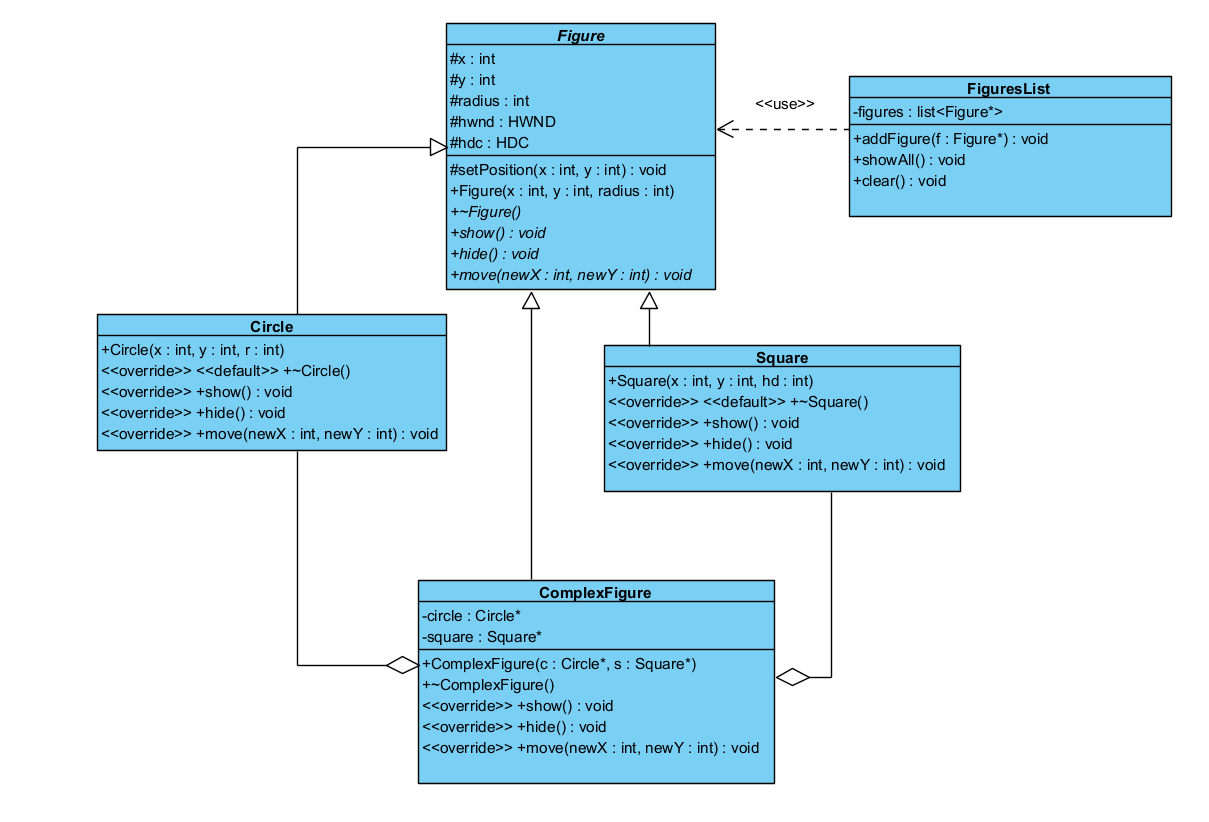


Рисунок 4 - Диаграмма классов

Классы спроектированы на основе результатов анализа предметной области и анализа функциональных требований. На приведенной диаграмме класс Figure является базовым и содержит следующие параметры: координаты центра фигуры (x, y), радиус (radius), дескриптор окна (HWND), контекст устройства (HDC), а также методы для установки позиции (setPosition), отображения (show), скрытия (hide) и перемещения фигуры (move).

Классы Circle (окружность) и Square (квадрат) наследуют от базового класса Figure координаты центра фигуры, радиус, дескриптор окна, контекст устройства, а также переопределяют методы отображения (show), скрытия (hide) и перемещения (move).

Класс ComplexFigure (сложная фигура) содержит агрегацию из объектов классов Circle и Square, что позволяет создавать составные фигуры. Он также переопределяет методы отображения, скрытия и перемещения для управления составной фигурой как единым целым.

Класс FiguresList предназначен для управления коллекцией фигур. Он содержит список объектов типа Figure и предоставляет методы для добавления фигур (addFigure), отображения всех фигур (showAll) и очистки списка (clear).

Отношение <<use>> между FiguresList и Figure указывает на то, что FiguresList использует объекты класса Figure для выполнения своих функций.

Таблица 5 - Спецификация класса Figure

|  |  |
| --- | --- |
| Назначение класса: | Базовый класс для представления геометрических фигур в консольном приложении. Инкапсулирует общие свойства и методы всех фигур. |
| Члены класса: | x : int, y : int — координаты центра фигуры;  radius : int — радиус фигуры;  hwnd : HWND — дескриптор окна;  hdc : HDC — контекст устройства. |
| Функции класса: | Figure(int, int, int) — конструктор с параметрами (координаты и радиус);  ~Figure() — деструктор;  setPosition(int, int) — установить позицию;  show() — отобразить фигуру;  hide() — скрыть фигуру;  move(int, int) — переместить фигуру. |

Таблица 6 - Спецификация класса Circle

|  |  |
| --- | --- |
| Назначение класса: | Класс, представляющий окружность. Наследует свойства и методы Figure. |
| Члены класса: | (Наследует x, y, radius, hwnd, hdc от Figure). |
| Функции класса: | Circle(int, int, int) — конструктор с параметрами (координаты и радиус);  ~Circle() — деструктор;  show() — отобразить окружность;  hide() — скрыть окружность;  move(int, int) — переместить окружность.hide() – скрыть фигуру. |

Таблица 7 - Спецификация класса Square

|  |  |
| --- | --- |
| Назначение класса: | Класс, представляющий квадрат. Наследует свойства и методы Figure. |
| Члены класса: | (Наследует x, y, radius, hwnd, hdc от Figure). |
| Функции класса: | Square(int, int, int) — конструктор с параметрами (координаты и длина стороны);  ~Square() — деструктор;  show() — отобразить квадрат;  hide() — скрыть квадрат;  move(int, int) — переместить квадрат. |

Таблица 8 - Спецификация класса ComplexFigure

|  |  |
| --- | --- |
| Назначение класса: | Класс, связывающий Circle и Square через агрегацию. |
| Члены класса: | circle : Circle\* — указатель на окружность;  square : Square\* — указатель на квадрат. |
| Функции класса: | ComplexFigure(Circle\*, Square\*) — конструктор с параметрами;  ~ComplexFigure() — деструктор;  show() — отобразить сложную фигуру;  hide() — скрыть сложную фигуру;  move(int, int) — переместить сложную фигуру. |

Таблица 9 - Спецификация класса FiguresList

|  |  |
| --- | --- |
| Назначение класса: | Класс для хранения и управления коллекцией фигур. |
| Члены класса: | figures : list<Figure\*> — список указателей на фигуры. |
| Функции класса: | addFigure(Figure\*) — добавить фигуру в список;  showAll() — отобразить все фигуры;  clear() — очистить список. |

## Реализация

Программный код реализации для предметной области «Квадрат в окружности» приведен в Приложении А. Диаграмма компонентов приведена на рисунке 5 и рисунке Б.2

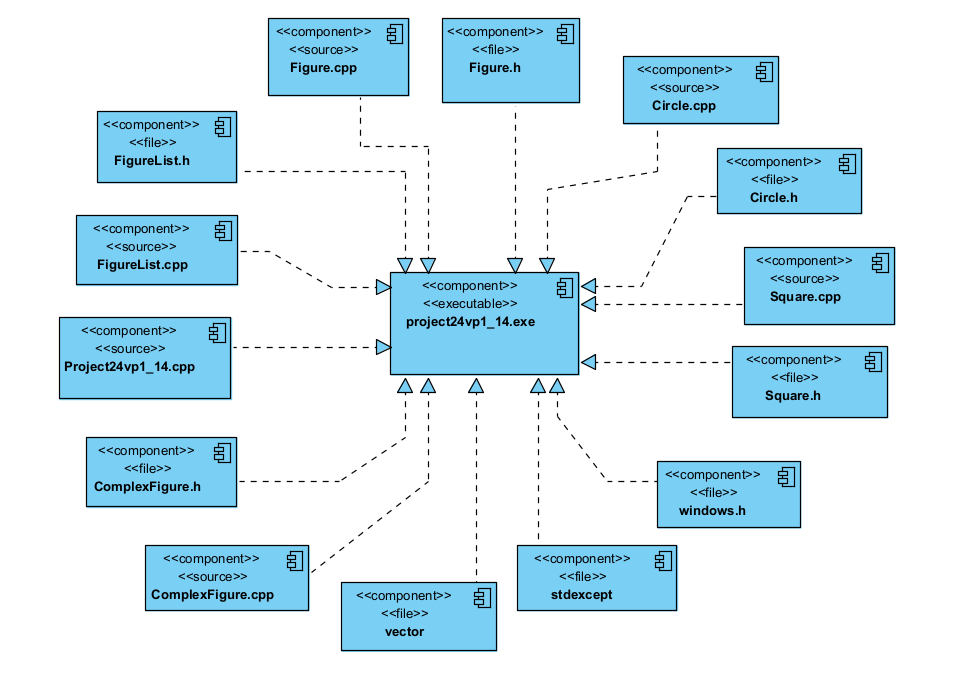


Рисунок 5 - Диаграмма компонентов

Описание компонентов приведено в таблице 10.

*Таблица 10 - Компоненты данной предметной области*

|  |  |
| --- | --- |
| **Компоненты** | **Назначение** |
| Figure.h | Заголовочный файл класса Figure |
| Figure.cpp | Исходный файл класса Figure |
| Circle.h | Заголовочный файл класса Circle |
| Circle.cpp | Исходный файл класса Circle |
| Square.h | Заголовочный файл класса Square |
| Square.cpp | Файл реализации класса Square |
| ComplexFigure.h | Заголовочный файл класса ComplexFigure |
| ComplexFigure.cpp | Исходный файл класса ComplexFigure |
| FigureList.h | Заголовочный файл класса FigureList |
| FigureList.cpp | Исходный файл класса FigureList |
| vector | Системный заголовочный файл |
| stdexept.h | Системный заголовочный файл |
| windows.h | Системный заголовочный файл |
| Project24vp1\_14.cpp | Главная программа |
| Project24vp1\_14.exe | Исполняемый файл приложения |

## Тестирование

В данном разделе представлены результаты тестирования проекта. Примеры тестирования представлены в таблице 11.

*Таблица 11 - Примеры тестирования*

|  |  |  |
| --- | --- | --- |
| **Вариант использования** | **Тест** | **Результат** |
| Создать окружность | Тест 1: координаты центра 100, 200, радиус 80, цвет контура RGB (0, 0, 0). | Тест пройден. Окружность создана (рисунок 6). |
| Тест 2: координаты центра 100, 200, радиус -80, цвет контура RGB (0, 0, 0). | Тест пройден. Сгенерировано и обработано исключение. (рисунок 7). |
| Переместить окружность | Тест 3: координаты центра 100, 200, радиус 80, цвет контура RGB (0, 0, 0). Новые координаты центра 150, 250. | Тест пройден. Окружность перемещена (рисунок 8). |
| Тест 4: координаты центра 100, 200, радиус 80, цвет контура RGB (0, 0, 0). Новые координаты центра -150, -250. | Тест пройден. Сгенерировано и обработано исключение (рисунок 9). |
| Создать сложную фигуру | Тест 5: координаты центра окружности 100, 200, радиус 80, цвет заливки RGB (137, 172, 118), цвет контура RGB(0, 0, 0). Координаты центра пятиугольника 100, 200, радиус 80, цвет заливки RGB (137, 172, 118), цвет контура RGB (0, 0, 0). | Тест пройден. Сложная фигура создана (рисунок 10). |
| Тест 6: координаты центра окружности 1000, 20000, радиус 80, цвет заливки RGB (137, 172, 118), цвет контура RGB(0, 0, 0). Координаты центра пятиугольника 100, 200, радиус 80, цвет заливки RGB (137, 172, 118), цвет контура RGB (0, 0, 0). | Тест пройден. Сгенерировано и обработано исключение (рисунок 11). |



Рисунок 6.



Рисунок 7.



Рисунок 8.



Рисунок 9.



Рисунок 10.

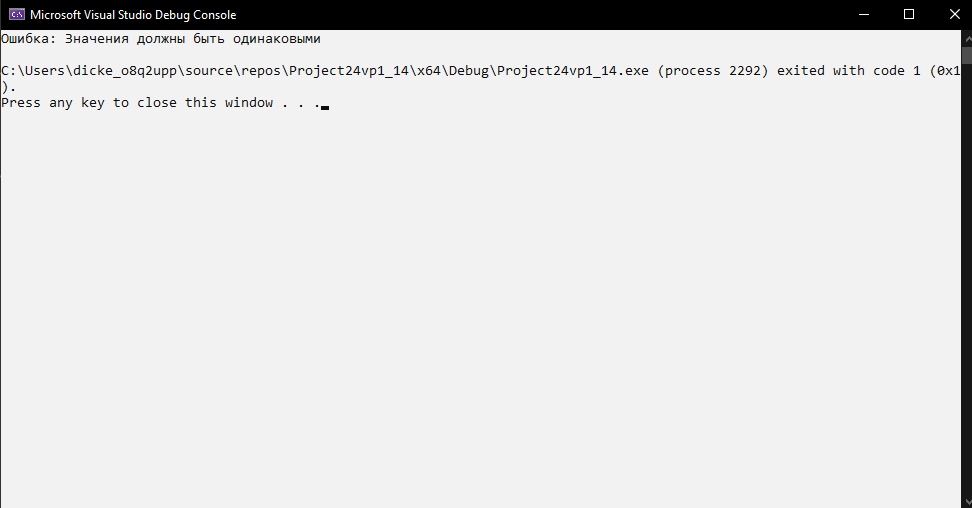


Рисунок 11.

# Заключение

В процессе разработки курсового проекта произведён анализ предметной области (составлена модель предметной области) и функциональных требований к проекту (разработана диаграмма вариантов использования, описаны сценарии нескольких вариантов использования).

В процессе проектирования построена диаграмма классов, спроектированы и реализованы классы: Figure, Circle, Square, ComplexFigure, FigureList; описаны их спецификации.

Результатом разработки стало приложение «project24vp1\_14.exe» для работы с геометрическими фигурами: квадрат, окружность, сложная фигура (квадрат, вписанный в окружность). Структура приложения отражена на диаграмме компонентов.

Заключительным этапом разработки приложения стало его тестирование, которое было пройдено успешно.

# Список использованных источников

1. Программирование на языке С++. Курсовое проектирование: учеб. пособие / Л. В. Гурьянов, Л. С. Гурьянова, Е. А. Дзюба [и др.]. – Пенза: Изд-во ПГУ, 2023. – 76 с.
2. Т. А. Павловская. C/C++. Программирование на языке высокого уровня. – СПб.: Питер, 2003. – 461 с.
3. Джим Арлоу. UML2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование/Джим Арлоу, Айла Нейштадт. – Санкт-Петербург, Издательство Символ-Плюс, 2007. – 624с.
4. Руководство по языку программирования C++. – URL: https:// metanit.com/cpp/tutorial/ (дата обращения: .05.2025).

# 

# Приложение А. Код программы

(обязательное)

**Circle.cpp**

#include <iostream>

#include <windows.h>

#include <stdexcept>

#include "Circle.h"

Circle::Circle(int x, int y, int r, COLORREF c) : Figure(x, y, r, c) {}

void Circle::show()

{

HPEN hPen = CreatePen(PS\_SOLID, 2, color);

SelectObject(hdc, hPen);

Ellipse(hdc, x - radius, y - radius, x + radius, y + radius);

DeleteObject(hPen);

}

void Circle::hide()

{

COLORREF bgColor = RGB(255, 255, 255);

HPEN hPen = CreatePen(PS\_SOLID, 2, bgColor);

SelectObject(hdc, hPen);

Ellipse(hdc, x - radius, y - radius, x + radius, y + radius);

DeleteObject(hPen);

}

void Circle::move(int newX, int newY)

{

if (newX <= 0 || newY <= 0) {

throw std::invalid\_argument("Попытка выхода за границы!");

}

hide();

setPosition(newX, newY);

show();

}

**Circle.h**

#pragma once

#include <iostream>

#include <windows.h>

#include "Figure.h"

class Circle : public Figure {

public:

Circle(int x, int y, int r, COLORREF c);

~Circle() override = default;

void show() override;

void hide() override;

void move(int newX, int newY) override;

};

**ComplexFigure.cpp**

#include "ComplexFigure.h"

ComplexFigure::ComplexFigure(Circle\* circle, Square\* square)

{

if (circle->getX() == square->getX() &&

circle->getY() == square->getY() &&

circle->getRadius() == square->getRadius())

{

this->circle = circle;

this->square = square;

}

else {

throw std::invalid\_argument("Значения должны быть одинаковыми");

}

}

ComplexFigure::~ComplexFigure()

{

delete circle;

delete square;

circle = nullptr;

square = nullptr;

}

void ComplexFigure::show()

{

circle->show();

square->show();

}

void ComplexFigure::hide() {

circle->hide();

square->hide();

}

void ComplexFigure::move(int newX, int newY)

{

circle->move(newX, newY);

square->move(newX, newY);

}

**ComplexFigure.h**

#pragma once

#include <iostream>

#include <windows.h>

#include "Figure.h"

#include "Circle.h"

#include "Square.h"

class ComplexFigure : public Figure {

private:

Circle\* circle;

Square\* square;

public:

ComplexFigure(Circle\* circle, Square\* square);

~ComplexFigure();

void show() override;

void hide() override;

void move(int newX, int newY) override;

};

**Figure.cpp**

#include <iostream>

#include <windows.h>

#include "Figure.h"

Figure::Figure() : x(0), y(0), radius(0), color(RGB(0, 0, 0))

{

hwnd = GetConsoleWindow();

hdc = GetDC(hwnd);

}

Figure::Figure(int x, int y, int radius, COLORREF color) : Figure()

{

this->x = x;

this->y = y;

this->radius = radius;

this->color = color;

if (radius <= 0) {

throw std::invalid\_argument("Радиус должен быть положительным");

}

if (x <= 0 || y <= 0) {

throw std::invalid\_argument("Координаты должны быть положительными");

}

}

Figure::~Figure()

{

ReleaseDC(hwnd, hdc);

}

void Figure::setPosition(int x, int y)

{

this->x = x;

this->y = y;

}

int Figure::getX()

{

return x;

}

int Figure::getY()

{

return y;

}

int Figure::getRadius()

{

return radius;

}

**Figure.h**

#pragma once

#include <iostream>

#include <windows.h>

class Figure {

protected:

int x, y, radius;

HWND hwnd;

HDC hdc;

COLORREF color;

void setPosition(int x, int y);

public:

Figure(int x, int y, int radius, COLORREF color);

Figure();

virtual ~Figure();

virtual void show() = 0;

virtual void hide() = 0;

virtual void move(int newX, int newY) = 0;

int getX();

int getY();

int getRadius();

};

**FigureList.cpp**

#include "FigureList.h"

FiguresList::~FiguresList()

{

clear();

}

void FiguresList::addFigure(Figure\* figure)

{

if (figure) {

figures.push\_back(figure);

}

}

void FiguresList::showAll()

{

for (auto figure : figures) {

figure->show();

}

}

void FiguresList::hideAll()

{

for (auto figure : figures) {

figure->hide();

}

}

void FiguresList::clear()

{

figures.clear();

}

**FigureList.h**

#pragma once

#include <vector>

#include "Figure.h"

class FiguresList {

private:

std::vector<Figure\*> figures;

public:

~FiguresList();

void addFigure(Figure\* f);

void showAll();

void hideAll();

void clear();

};

**Square.cpp**

#include "Square.h"

Square::Square(int x, int y, int r, COLORREF c, COLORREF cb) : Figure(x, y, r, c)

{

colorBrush = cb;

}

void Square::show()

{

POINT vertices[4];

vertices[0] = { x, y - radius };

vertices[1] = { x + radius, y };

vertices[2] = { x, y + radius };

vertices[3] = { x - radius, y };

HPEN hPen = CreatePen(PS\_SOLID, 2, color);

HBRUSH hBrush = CreateSolidBrush(colorBrush);

HGDIOBJ oldPen = SelectObject(hdc, hPen);

HGDIOBJ oldBrush = SelectObject(hdc, hBrush);

Polygon(hdc, vertices, 4);

SelectObject(hdc, oldPen);

SelectObject(hdc, oldBrush);

DeleteObject(hPen);

DeleteObject(hBrush);

}

void Square::hide()

{

POINT vertices[4];

vertices[0] = { x, y - radius };

vertices[1] = { x + radius, y };

vertices[2] = { x, y + radius };

vertices[3] = { x - radius, y };

HPEN hPen = CreatePen(PS\_SOLID, 2, RGB(255, 255, 255));

HBRUSH hBrush = CreateSolidBrush(RGB(255, 255, 255));

HGDIOBJ oldPen = SelectObject(hdc, hPen);

HGDIOBJ oldBrush = SelectObject(hdc, hBrush);

Polygon(hdc, vertices, 4);

SelectObject(hdc, oldPen);

SelectObject(hdc, oldBrush);

DeleteObject(hPen);

DeleteObject(hBrush);

}

void Square::move(int newX, int newY)

{

if (newX <= 0 || newY <= 0) {

throw std::invalid\_argument("Попытка выхода за границы!");

}

hide();

setPosition(newX, newY);

show();

}

**Square.h**

#pragma once

#include <iostream>

#include <windows.h>

#include "Figure.h"

class Square : public Figure {

private:

COLORREF colorBrush;

public:

Square(int x, int y, int r, COLORREF c, COLORREF cb);

~Square() override = default;

void show() override;

void hide() override;

void move(int newX, int newY) override;

};

**Project24vp1\_14.cpp**

#include <iostream>

#include <windows.h>

#include "Circle.h"

#include "Square.h"

#include "ComplexFigure.h"

#include "FigureList.h"

using namespace std;

// ВНЕС ИЗМЕНЕНИЕ

int main() {

setlocale(LC\_ALL, "RU");

system("color F0");

try {

COLORREF colorBrush = RGB(137, 172, 118);

COLORREF color = RGB(0, 0, 0);

Figure\* circle = new Circle(1000, 20000, 80, color);

Figure\* square = new Square(100, 200, 80, color, colorBrush);

Figure\* complexFig = new ComplexFigure((Circle\*)circle, (Square\*)square);

Circle\* circle1 = new Circle(300, 200, 80, color);

Figure\* square1 = new Square(300, 200, 80, color, colorBrush);

FiguresList list;

list.addFigure(complexFig);

list.addFigure(circle1);

list.showAll();

cin.get();

complexFig->move(500, 200);

cin.get();

circle1->move(300, 400);

cin.get();

list.hideAll();

cin.get();

list.clear();

list.addFigure(square1);

list.showAll();

cin.get();

list.hideAll();

cin.get();

}

catch (const std::exception& e) {

std::cerr << "Ошибка: " << e.what() << std::endl;

return 1;

}

return 0;

}

# Приложение Б. Код программы

(обязательное)

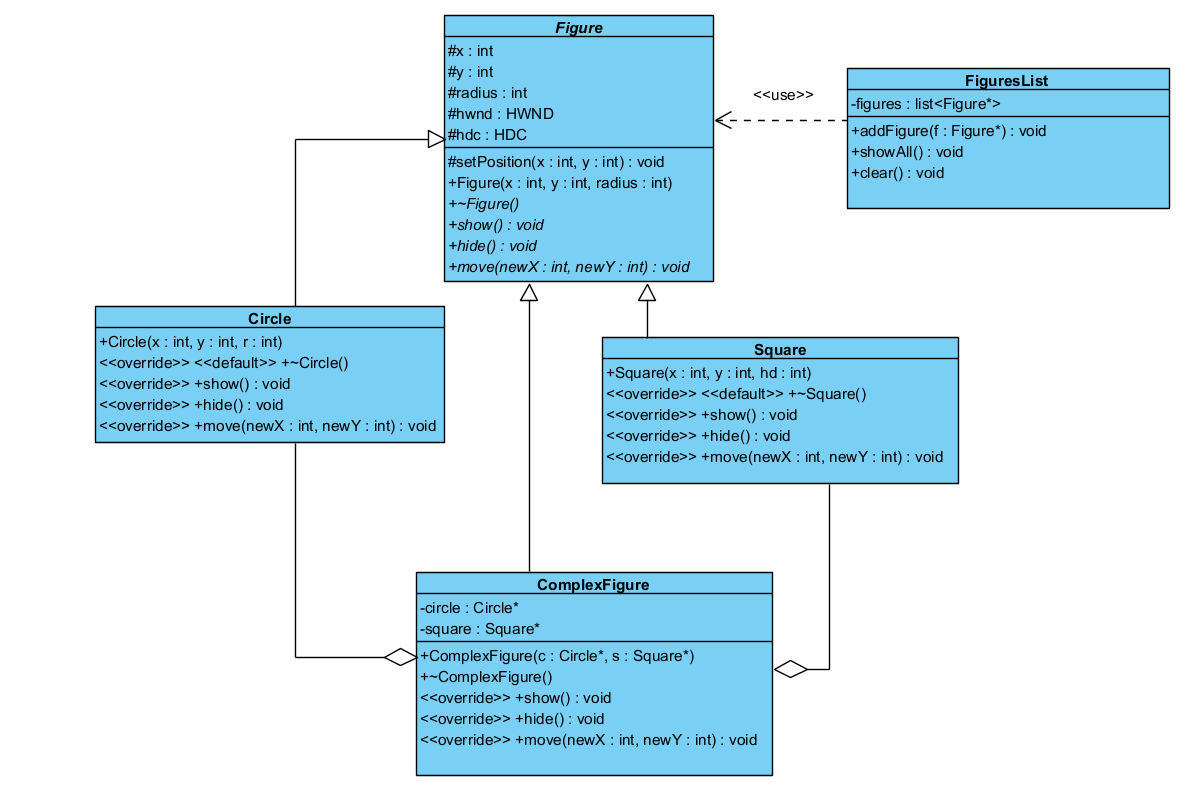


Рисунок Б.1 – Диаграмма классов

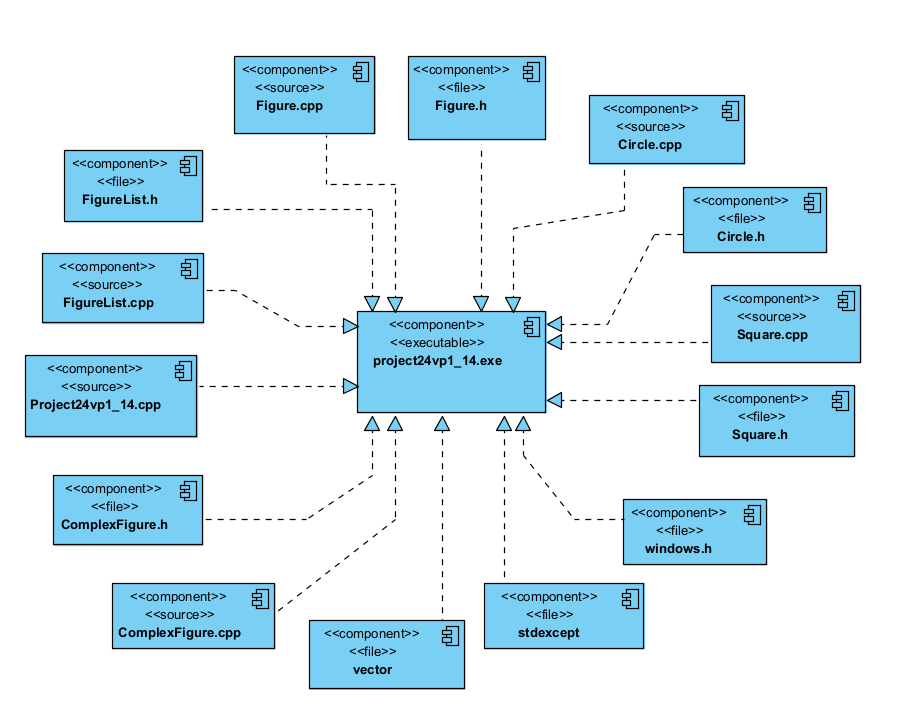


Рисунок Б.2 – Диаграмма компонентов