МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Кафедра «Математическое обеспечение и применение ЭВМ»

Курсовой проект

по дисциплине «Программирование»

на тему: Разработка объектно-ориентированного приложения. Классы окружность и квадрат.

ПГУ 09.03.04 – 02КП201.12 ПЗ

Направление подготовки – 09.03.04 «Программная инженерия»

Выполнил студент: Кудашов Александр Сергеевич
Группа: 20ВП1

Руководитель:
к.т.н., доцент _____ Гурьянов Л.В.

Проект защищен с оценкой _____
Преподаватели _____
_____
_____
Дата защиты _____

Пенза 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Математическое обеспечение и применение ЭВМ»

«УТВЕРЖДАЮ»

заведующий кафедрой

_____П.П. Макарычев

«___»_____ 2021г.

ЗАДАНИЕ

на курсовой проект

по дисциплине «Программирование»

на тему: Разработка объектно-ориентированного приложения. Классы окружность и квадрат.

1.	**Студент гр.** *20ВП1* **факультета** ВТ **направления** 09.03.04
	Кудашов Александр Сергеевич
2.	**Руководитель работы** *Гурьянов Лев Вячеславович*
3.	**Время проектирования** с «15» февраля 2021 по «30» мая 2021
4.	**Тема проекта**: *Разработка объектно-ориентированного приложения. Классы окружность и квадрат.*
5.	**Техническое задание на курсовую работу (назначение, технические требования)**
	Назначение: *создание и визуализация следующих типов фигур:*
	квадрат, окружность, сложная фигура (квадрат, вписанный в окружность)
	Основные функции приложения*:*
	а) показать геометрическую фигуру (квадрат, окружность, сложная фигура (квадрат, вписанный в окружность));
	б) скрыть фигуру;
	в) переместить фигуру;
	Структура данных: *динамический массив*
	Технология разработки*: ООП*
	Язык программирования*: C++*
	Среда исполнения*: Windows*
6.	**Содержание работы**

6.1. **Пояснительная записка (перечень вопросов, подлежащих разработке, расчетов, обоснований, описаний)**

1) *Анализ предметной области*
2) *Анализ функциональных требований*
3) *Проектирование*
4) *Реализация*
5) *Тестирование*
6) *Оформление пояснительной записки*

7. Календарный график по выполнению работы

Наименование этапов работы	Объем работы (%)	Срок выполнения	Подпись руководителя
1) *Анализ предметной области и требований*	10	25.02.2021	
2) *Проектирование*	20	17.03.2021	
3) *Реализация*	30	20.04.2021	
4) *Тестирование*	20	10.05.2021	
5) *Оформление пояснительной записки*	20	30.05.2021	

Дата выдачи задания «15»февраля 2021г.

Руководитель курсового проекта _____ Гурьянов Л.В.

Задание к исполнению принял «15 »февраля 2021г.

Студент _____ Кудашов А.С.

Оглавление

Введение

В курсовом проекте требуется разработать объектно-ориентированной приложение для создания и визуализации геометрических фигур: квадрата и окружности.

Для моделирования программных средств используется язык UML. Разработка осуществляется на языке C++ на платформе Microsoft Visual Studio 19.

Процесс создания ПС включает следующие этапы: анализ предметной области и требований к программным средствам, проектирование, реализация и тестирование.

Графическая часть проекта включает диаграмму классов и диаграмму компонентов, выполненных в нотации UML.

1. Разработка объектно-ориентированного приложения. Классы окружность и квадрат.

1.1 Анализ предметной области

В техническом задании представлена предметная область разработки, изображенная на рисунке 1.

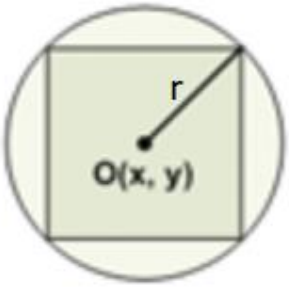

Рисунок 1 – Заданная геометрическая фигура

Модель предметной области для приведенной выше фигуры приведена на рисунке 2

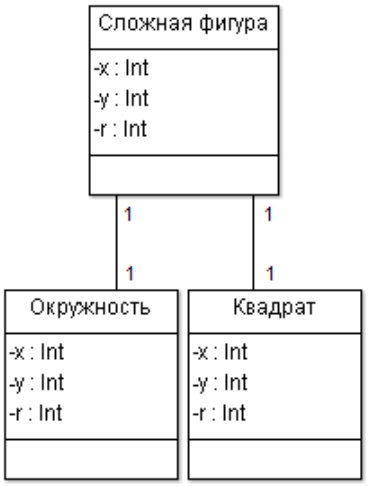

Рисунок 2 – Модель предметной области

Для предметной области, приведенной на рисунке 1, сложная фигура представляет квадрат, вписанный в окружность, и определяется координатами центра (x, y) и радиусом окружности (r). Простая фигура окружность определяется аналогично сложной фигуре. Простая фигура квадрат определяется координатами центра (x, y) и длиной половины диагонали (r), численно равной радиусу окружности при составлении сложной фигуры.

1.2 Анализ функциональных требований

В техническом задании на курсовой проект определены следующие функциональные требования:

- показать геометрическую фигуру (квадрат, окружность, сложная фигура (квадрат, вписанный в окружность));

- скрыть фигуру;

- переместить фигуру.

Диаграмма вариантов использования, соответствующая этим требованиям, приведена на рисунке 3. На диаграмме варианты «Показать квадрат» и «Скрыть квадрат» расширяют функциональность варианта «Переместить квадрат» (отношение «extend» – способ введения нового поведения в существующий вариант использования). Аналогичное отношение расширения используется для вариантов окружности и сложной фигуры. Это расширение обосновано алгоритмом функции «Переместить фигуру», который сначала удаляет геометрическую фигуру, а потом показывает ее на новом месте (относительно новой точки «привязки»).

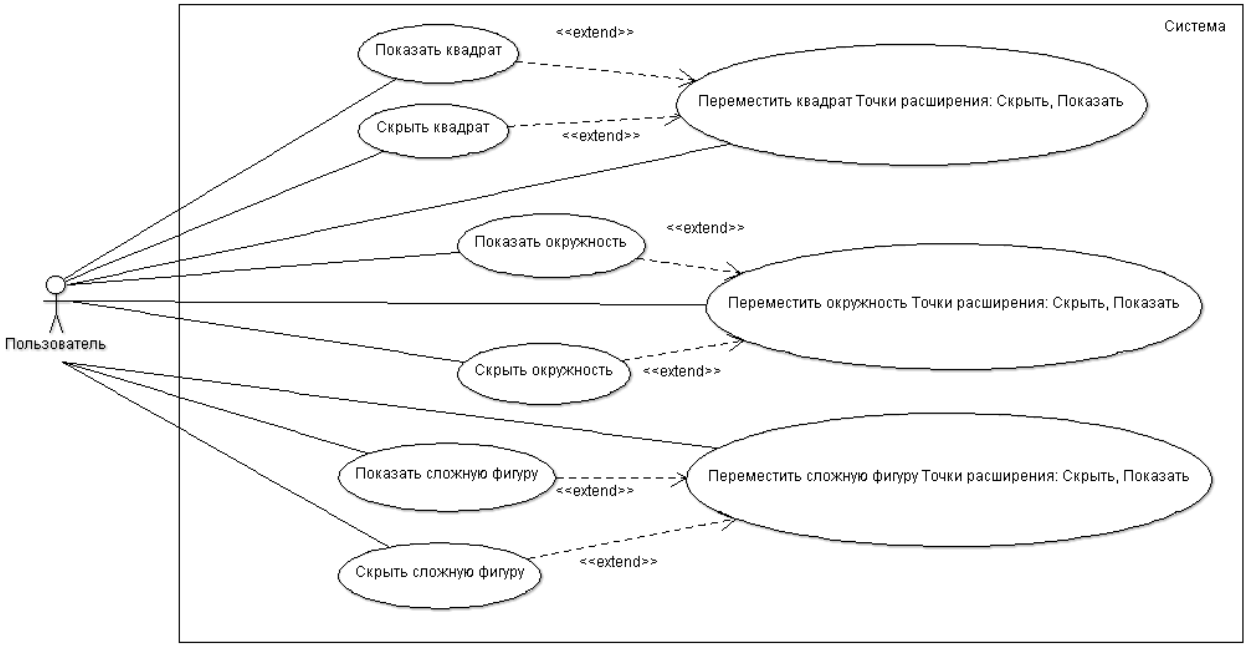

Рисунок 3 – Диаграмма вариантов использования

Сценарии варианта использования «Переместить квадрат» и расширяющих его вариантов приведены в таблицах 1 – 3.

Таблица 1 – Сценарий варианта использования «Переместить квадрат»

Наименование: Переместить квадрат
ID: 4
Краткое описание: система удаляет изображение квадрата и рисует его в новых координатах
Действующие лица: пользователь
Предусловие: квадрат создан и нарисован
Основной поток: 1. Пользователь задает новые координаты центра (x, y) и длину половины диагонали (r) квадрата Точка расширения «Скрыть» 2 Система переопределяет координаты квадрата Точка расширения «Нарисовать»
Постусловие: квадрат перемещен

Таблица 2 – Сценарий варианта использования «Скрыть квадрат»

Наименование: Скрыть квадрат
ID: 3
Краткое описание: система удаляет изображение квадрата
Действующие лица: пользователь
Предусловие: квадрат создан и нарисован
Основной поток: 1. Пользователь инициирует удаление изображения квадрата 2. Система рисует изображение квадрата цветом фона
Постусловие: изображение квадрата скрыто

Таблица 3 – Сценарий варианта использования «Нарисовать квадрат»

Наименование: Нарисовать квадрат
ID: 2
Краткое описание: система рисует изображение квадрата с координатами центра (x, y) и заданной длиной половины диагонали (r)
Действующие лица: пользователь
Предусловие: квадрат создан
Основной поток: 1. Пользователь инициирует рисование квадрата (задает координаты центра (x, y) и длину половины диагонали (r) квадрата) 2. Система рисует изображение квадрата с координатами центра (x, y) и заданной длиной половины диагонали (r)
Постусловие: квадрат нарисован

1.3 Проектирование

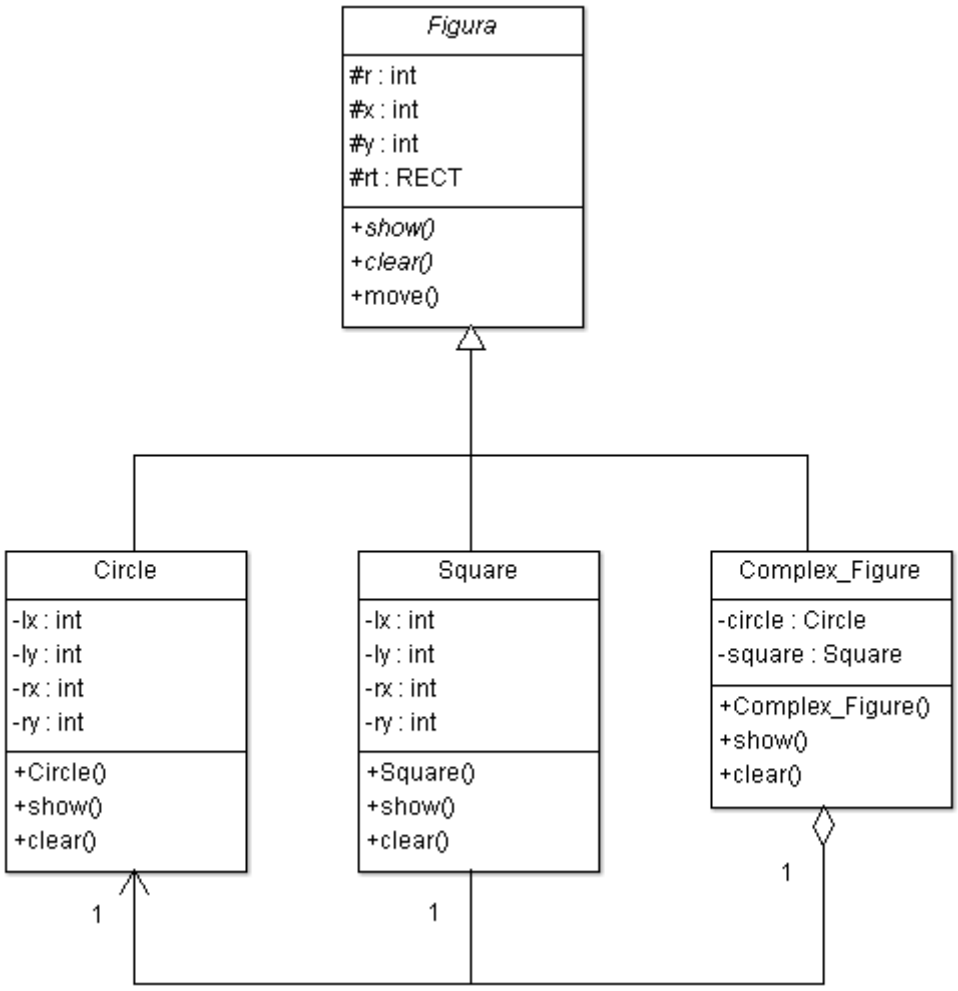

Рисунок 4 – Диаграмма классов

На приведенной диаграмме классы окружность (Circle), квадрат (Square) и сложная фигура (Complex_Figure) наследуют из базового абстрактного класса *Figure* радиус и координаты центра фигуры (r, x, y); класс сложной фигуры (ComplexFigure) содержит агрегацию из объектов классов Circle и Square.

Спецификация классов

Таблица 4 – Спецификация класса *Figura*

Название класса:	*Figura*
Назначение класса:	Абстрактный родительский класс, объединяющий поля и методы, свойственные всем типам фигур
Члены класса:	r: int – длина половины диагонали для квадрата (для окружности – радиус) фигуры; x: int – абсцисса центра фигур; y: int – ордината центра фигуры; rt: RECT – прямоугольное окно консольного прил-я
Функции класса:	*show()* – показать фигуру; *clear()* – скрыть фигуру; move(int, int) – переместить фигуру, принимает новые координаты центра фигуры.

Таблица 5 – Спецификация класса Circle

Название класса:	Circle
Назначение класса:	Класс сущности фигуры круг, наследник класса Figure
Члены класса:	lx: int – абсцисса верхней левой вершины квадрата, описанного вокруг окружности; ly: int – ордината верхней левой вершины квадрата, описанного вокруг окружности; rx: int – абсцисса нижней правой вершины квадрата, описанного вокруг окружности; ry: int – ордината нижней правой вершины квадрата, описанного вокруг окружности.
Функции класса:	Circle(int, int, int) – конструктор с радиусом и координатами центра окружности; show() – показать окружность; clear() – скрыть окружность.

Таблица 6 – Спецификация класса Square

Название класса:	Square
Назначение класса:	Класс сущности фигуры квадрат, наследник класса Figure
Члены класса:	lx: int – абсцисса верхней левой вершины квадрата; ly: int – ордината верхней левой вершины квадрата; rx: int – абсцисса нижней правой вершины квадрата; ry: int – ордината нижней правой вершины квадрата.
Функции класса:	Square(int, int, int) – конструктор с длиной половины диагонали и координатами центра квадрата; show() – показать окружность; clear() – скрыть окружность.

Таблица 7 – Спецификация класса Complex_Figure

Название класса:	Complex_Figure
Назначение класса:	Класс сущности сложной фигуры, наследник класса Figure
Члены класса:	Circle:Circle – объект класса Circle, составляющая сложной фигуры Square:Square – объект класса Square, составляющая сложной фигуры
Функции класса:	Comlex_Figure(int, int, int) – конструктор с длиной половины диагонали и координатами центра сложной фигуры; show() – показать сложную фигуру; clear() – скрыть сложную фигуру.

1.4 Реализация

Диаграмма компонентов приведена на рисунке 5.

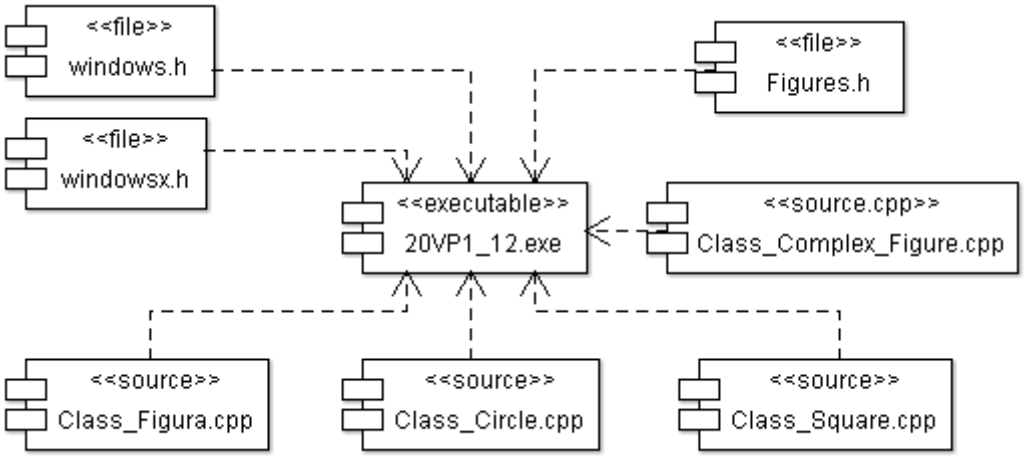

Рисунок 5 – Диаграмма компонентов

Описание компонентов приведено в таблице 8.

Таблица 8 - Компоненты

Компоненты	Назначение
20VP1_12.exe	Исполняемый файл приложения
Figures.h	Заголовочный файл проекта
Class_Figura.cpp	Исходный файл класса Figure
Class_Circle.cpp	Исходный файл класса Circle
Class_Square.cpp	Исходный файл класса Square
Class_Complex_Figure.cpp	Исходный файл класса ComplexFigure
windows.h	Системный заголовочный файл
windowsx.h	Системный заголовочный файл

1.5 Тестирование

Результаты функционального тестирования представлены в таблице 9.

Таблица 9 – План тестирования

Вариант использования	**Тест**	**Результат**
Показать окружность	**Тест1** Координаты центра: 300, 200 Радиус: 50	Тест пройден. Окружность нарисована правильно (рисунок 6)
	Тест2 Координаты центра: 100, 200 Радиус: 150	Тест пройден. Окружность нарисована правильно
	Тест 3 Координаты центра: 275, 325 Радиус: 75	Тест пройден. Окружность нарисована правильно
Скрыть окружность	**Тест1** Координаты центра: 300, 200 Радиус: 50	Тест пройден Окружность скрыта (рисунок 7)
	Тест2 Координаты центра: 100, 200 Радиус: 150	Тест пройден Окружность скрыта
	Тест 3 Координаты центра: 275, 325 Радиус: 75	Тест пройден Окружность скрыта
Переместить окружность	**Тест1** Новые координаты центра: 100, 400	Тест пройден. Окружность перемещена в новые координаты центра (рисунок 8)

	Тест2 Новые координаты центра: 400, 350	Тест пройден. Успешно обработано исключение выхода окружности за границы окна (рисунок 9)
	Тест3 Новые координаты центра: 700, 225	Тест пройден. Окружность перемещена в новые координаты центра
Показать квадрат	**Тест1** Координаты центра: 710, 170 Длина половины диагонали: 40	Тест пройден. Квадрат нарисован правильно (рисунок 6)
	Тест2 Координаты центра: 200, 300 Длина половины диагонали: 125	Тест пройден. Квадрат нарисован правильно
	Тест 3 Координаты центра: 50, 75 Длина половины диагонали: 105	Тест пройден. Успешно обработано исключение выхода квадрата за границы окна
Скрыть квадрат	**Тест1** Координаты центра: 710, 170 Длина половины диагонали: 40	Тест пройден Квадрат скрыт (рисунок 7)
	Тест2 Координаты центра: 200, 300 Длина половины диагонали: 125	Тест пройден Квадрат скрыт
Переместить квадрат	**Тест1** Новые координаты центра: 600, 300	Тест пройден. Квадрат перемещен в новые координаты центра (рисунок 8)

	Тест2 Новые координаты центра: 700, 225	Тест пройден. Квадрат перемещен в новые координаты центра
Показать сложную фигуру	**Тест1** Координаты центра: 340, 340 Длина половины диагонали: 50	Тест пройден. Сложная фигура нарисована правильно (рисунок 6)
	Тест2 Координаты центра: 517, 276 Длина половины диагонали: 123	Тест пройден. Сложная фигура нарисована правильно
	Тест 3 Координаты центра: 200, 300 Длина половины диагонали: 70	Тест пройден. Сложная фигура нарисована правильно
Скрыть сложную фигуру	**Тест1** Координаты центра: 340, 340 Длина половины диагонали: 50	Тест пройден Сложная фигура скрыта (рисунок 7)
	Тест2 Координаты центра: 517, 276 Длина половины диагонали: 123	Тест пройден Сложная фигура скрыта
	Тест 3 Координаты центра: 200, 300 Длина половины диагонали: 70	Тест пройден Сложная фигура скрыта
Переместить сложную фигуру	**Тест1** Новые координаты центра: 100, 200	Тест пройден. Сложная фигура перемещена в новые координаты центра (рисунок 8)

	Тест2 Новые координаты центра: 350, 358	Тест пройден. Успешно обработано исключение выхода сложной фигуры за границы окна
	Тест3 Новые координаты центра: 250, 400	Тест пройден. Сложная фигура перемещена в новые координаты центра

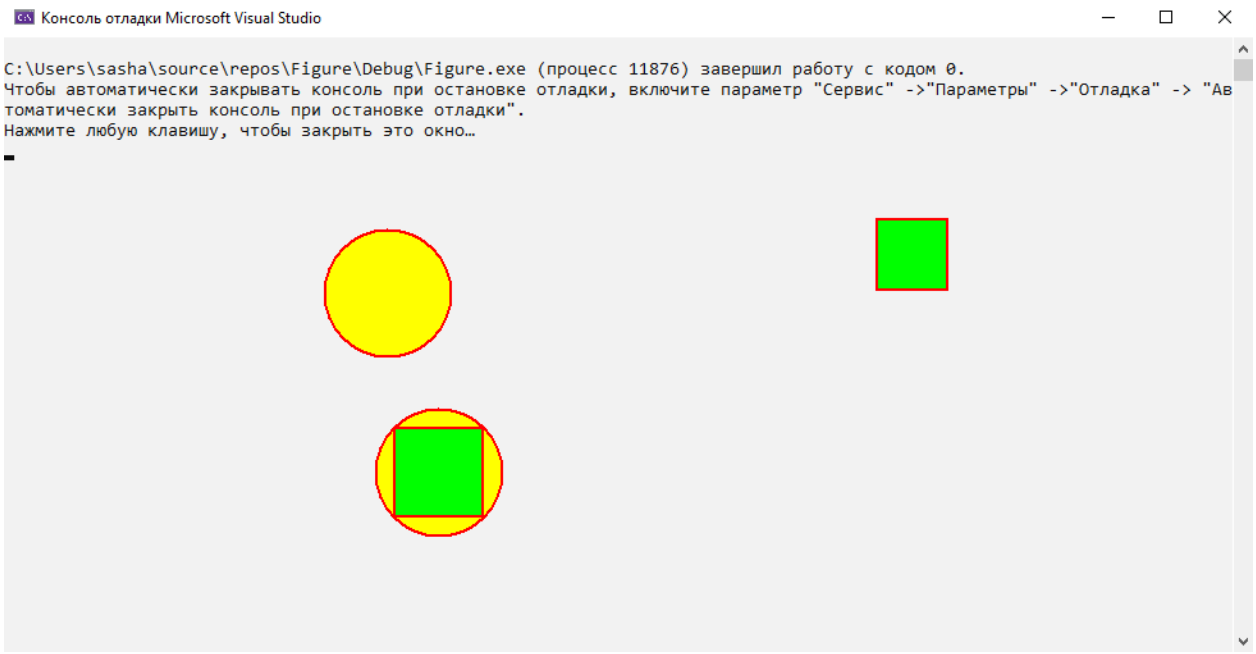

Рисунок 6 – Тестирование функции show ()

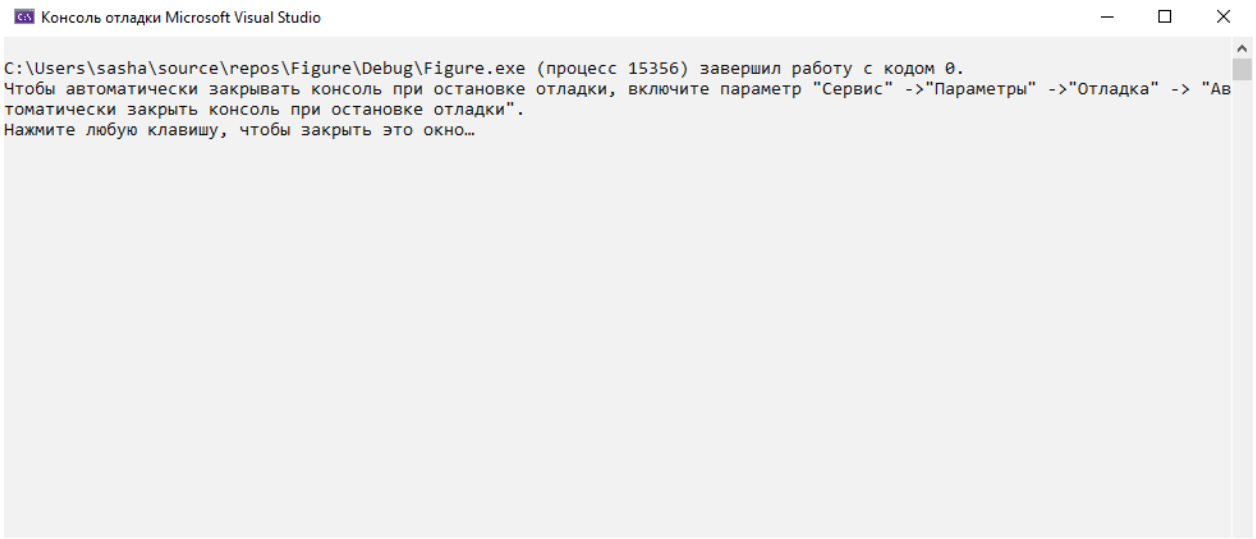

Рисунок 7 - Тестирование функции clear ()

Рисунок 8 - Тестирование функции move (int, int)

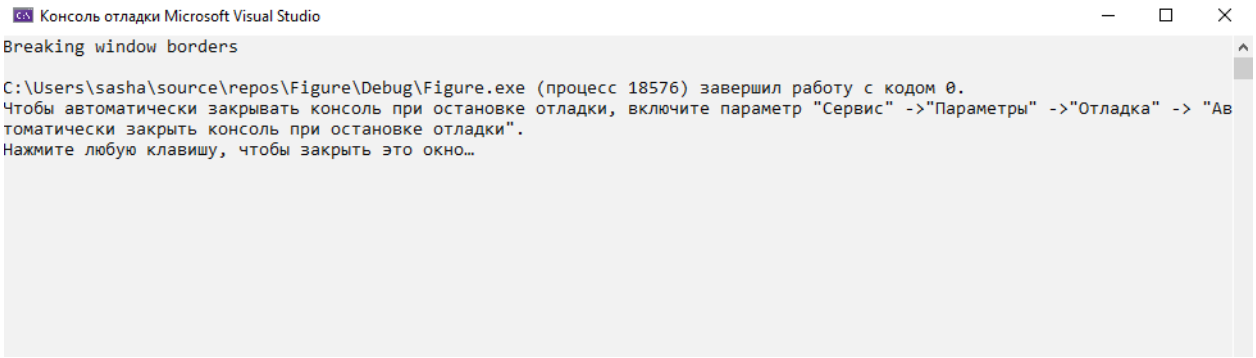

Рисунок 9 - Тестирование функции move (int, int). Обработка исключения

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 варианта использования «Переместить квадрат» и расширяющих его вариантов). В процессе проектирования построена диаграмма классов, спроектированы и реализованы классы: *Figura*, Circle, Square, Complex_Figure; описаны их спецификации. Результатом разработки стало приложение «20VP1_12.exe» для работы с геометрическими фигурами: окружность, квадрат, 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. Л.В.Гурьянов. Введение в программирование на языке С++/Л.В.Гурьянов, Л.С. Гурьянова, Е.А.Дзюба, Д.В.Такташкин. – Лабораторный практикум: Издательство ПГУ, 2010. – 91с

2. Джим Арлоу. UML2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование/Джим Арлоу, Айла Нейштадт. – Санкт-Петербург, Издательство Символ-Плюс, 2007. – 624с

3. Т. А. Павловская. C/C++. Программирование на языке высокого уровня. – СПб.: Питер, 2003. – 461 с

Приложение А. Код приложения

Главная программа Figure.cpp

```
#include "Figures.h"

int main() {
    SetConsoleTitle((LPCWSTR)L"20VP1_12");
// заголовок консоли
    vector <Figura*> figures(2);
// создание вектора для хранения фигур
    figures[0] = reinterpret_cast<Figura*>(new Circle(75, 250, 225));
// заполнение вектора фигурами
    figures[1] = reinterpret_cast<Figura*>(new Square(75, 250, 380));
// заполнение вектора фигурами
    figures.push_back(reinterpret_cast<Figura*>(new Complex_Figure(75, 450, 380)));
// добавление фигуры в вектор
    figures.push_back(reinterpret_cast<Figura*>(new Complex_Figure(55, 70, 280)));
// добавление фигуры в вектор
    try {
        for (int i = 0; i < figures.size(); ++i) figures[i]->show();
// отрисовка фигур из вектора
        figures[2]->move(700, 225);
// перемещение фигуры
        figures[1]->clear();
// удаление фигуры
    }
    catch (Figura::Border) {
        cout << "Breaking window borders" << endl;
// обработка исключения выхода за границы окна
    }
    //system("pause");
// требуется для работы exe-файла
    return 0;
}
```

Figures.h

```
#pragma once
#include <iostream>
#include <windows.h>
#include <windowsx.h>
#include <vector>
using namespace std;

const int NotUsed = system("color F0");
// цвет фона окна

class Figura {
protected:
    int r = 0;
// радиус фигуры
    int x = 0;
// абсцисса центра фигуры
    int y = 0;
// ордината центра фигура
    RECT rt;
// rt-прямоугольник
public:
    virtual void show() = 0;
// отрисовка фигуры
    virtual void clear() = 0;
// удаление фигуры
    void move(int, int);
// перемещение фигуры
```

```
    class Border {};
// класс для обработки исключений
};

class Circle :public Figura {
private:
    int lx, ly, rx, ry;
// координаты верхнего левого и нижнего правого углов
public:
    void show() override;
// отрисовка круга
    void clear() override;
// удаление круга
    Circle(int new_r = 0, int new_x = 0, int new_y = 0);
// конструтор с параметрами - координатами, задающими круг
};

class Square :public Figura {
private:
    int lx, ly, rx, ry;
// координаты верхнего левого и нижнего правого углов
public:
    void show() override;
// отрисовка квадрата
    void clear() override;
// удаление квадрата
    Square(int new_r = 0, int new_x = 0, int new_y = 0);
// конструтор с параметрами - координатами, задающими квадрат
};

class Complex_Figure : Figura {
private:
    Circle circle;
    Square square;
public:
    void show() override;
// отрисовка сложной фигуры
    void clear() override;
// удаление сложной фигуры
    Complex_Figure(int new_r = 0, int new_x = 0, int new_y = 0);
// конструтор с параметрами - координатами, задающими сложную фигуру
};
```

Class_Figura.cpp

```
#include "Figures.h"

// перемещение фигуры
void Figura::move(int new_x, int new_y) {
    clear();
// удаление фигуры
// обновление координат
    x = new_x;
    y = new_y;
    show();
// отрисовка фигуры по новым координатам
}
```

Class_Circle.cpp

```
#include "Figures.h"

// отрисовка круга
void Circle::show() {
    HWND hwnd = GetConsoleWindow();
// получаем идентификатор окна
```

```
    HDC hdc = GetDC(hwnd);
// получаем контекст отображения
    HPEN pen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
// создаем перо
    HBRUSH brush = CreateSolidBrush(RGB(255, 255, 0));
// создаем кисть
    GetClientRect(hwnd, &rt);
// получаем размер окна
    SelectObject(hdc, pen); SelectObject(hdc, brush);
// назначаем перо и кисть для рисования
    lx = x - r;
// обновляем координаты для построения
    ly = y - r;
    rx = x + r;
    ry = y + r;
    if ((lx <= rt.left)||(ly <= rt.top)||(rx >= rt.right)||(ry >= rt.bottom)) throw
Border();   // отлавливание выхода за границы окна
    Ellipse(hdc, lx, ly, rx, ry);
// рисуем окружность
    DeleteObject(pen);
// освобождаем ресурсы
    DeleteObject(brush);
    DeletePen(pen);
// удаляем перо
    DeleteBrush(brush);
// удаляем кисть
    ReleaseDC(hwnd, hdc);
// освобождаем контекст отображения
}
// удаление круга
void Circle::clear() {
    HWND hwnd = GetConsoleWindow();
// получаем идентификатор окна
    HDC hdc = GetDC(hwnd);
// получаем контекст отображения
    HPEN pen = CreatePen(PS_SOLID, 2, RGB(242, 242, 242));
// создаем перо
    HBRUSH brush = CreateSolidBrush(RGB(242, 242, 242));
// создаем кисть
    GetClientRect(hwnd, &rt);
// получаем размер окна
    SelectObject(hdc, pen); SelectObject(hdc, brush);
// назначаем перо и кисть для рисования
    Ellipse(hdc, lx, ly, rx, ry);
// стираем окружность
    DeleteObject(pen);
// освобождаем ресурсы
    DeleteObject(brush);
    DeletePen(pen);
// удаляем перо
    DeleteBrush(brush);
// удаляем кисть
    ReleaseDC(hwnd, hdc);
// освобождаем контекст отображения
}
// конструтор с параметрами - координатами, задающими круг
Circle::Circle(int new_r, int new_x, int new_y) {
    r = new_r;
// объявление начальных координат
    x = new_x;
    y = new_y;
    lx = x - r;
// расчёт координат для построения
    ly = y - r;
```

```
    rx = x + r;
    ry = y + r;
}
```

Class_Square.cpp

```
#include "Figures.h"

// отрисовка квадрата
void Square::show() {
    HWND hwnd = GetConsoleWindow();
// получаем идентификатор окна
    HDC hdc = GetDC(hwnd);
// получаем контекст отображения
    HPEN hRedPen = CreatePen(PS_SOLID, 2, RGB(255, 0, 0));
// создаем красное перо
    HBRUSH hGreenBrush = CreateSolidBrush(RGB(0, 255, 0));
// создаем зеленую кисть
    GetClientRect(hwnd, &rt);
// получаем размер окна
    SelectObject(hdc, hRedPen); SelectObject(hdc, hGreenBrush);
// назначаем перо и кисть для рисования
    lx = x - 0.7 * r;
// обновляем координаты для построения
    ly = y - 0.7 * r;
    rx = x + 0.7 * r;
    ry = y + 0.7 * r;
    if ((lx <= rt.left)||(ly <= rt.top)||(rx >= rt.right)||(ry >= rt.bottom)) throw
Border();   // отлавливание выхода за границы окна
    Rectangle(hdc, lx, ly, rx, ry);
// рисуем квадрат
    DeleteObject(hRedPen);
// освобождаем ресурсы
    DeleteObject(hGreenBrush);
    DeletePen(hRedPen);
// удаляем красное перо
    DeleteBrush(hGreenBrush);
// удаляем зеленую кисть
    ReleaseDC(hwnd, hdc);
// освобождаем контекст отображения
}
// удаление квадрата
void Square::clear() {
    HWND hwnd = GetConsoleWindow();
// получаем идентификатор окна
    HDC hdc = GetDC(hwnd);
// получаем контекст отображения
    HPEN hRedPen = CreatePen(PS_SOLID, 2, RGB(242, 242, 242));
// создаем красное перо
    HBRUSH hGreenBrush = CreateSolidBrush(RGB(242, 242, 242));
// создаем зеленую кисть
    GetClientRect(hwnd, &rt);
// получаем размер окна
    SelectObject(hdc, hRedPen); SelectObject(hdc, hGreenBrush);
// назначаем перо и кисть для рисования
    Rectangle(hdc, lx, ly, rx, ry);
// стираем квадрат
    DeleteObject(hRedPen);
// освобождаем ресурсы
    DeleteObject(hGreenBrush);
    DeletePen(hRedPen);
// удаляем красное перо
    DeleteBrush(hGreenBrush);
// удаляем зеленую кисть
    ReleaseDC(hwnd, hdc);
// освобождаем контекст отображения
```

```
}
// конструтор с параметрами - координатами, задающими квадрат
Square::Square(int new_r, int new_x, int new_y) {
    r = new_r;
// объявление начальных координат
    x = new_x;
    y = new_y;
    lx = x - 0.7 * r;
// расчёт координат для построения
    ly = y - 0.7 * r;
    rx = x + 0.7 * r;
    ry = y + 0.7 * r;
}
```

Class_Complex_Figure.cpp

```
#include "Figures.h"

// отрисовка сложной фигуры
void Complex_Figure::show() {
    circle = Circle(r, x, y);
// необходимо обновлять координаты для вызова из move()
    square = Square(r, x, y);
    circle.show();
// отрисовка круга
    square.show();
// отрисовка квадрата

}
// удаление сложной фигуры
void Complex_Figure::clear() {
    circle.clear();
// удаление квадрата
    square.clear();
// удаление круга
}
// конструтор с параметрами - координатами, задающими сложную фигуру
Complex_Figure::Complex_Figure(int new_r, int new_x, int new_y) {
    r = new_r;
// объявление начальных координат
    x = new_x;
    y = new_y;
    circle = Circle(r, x, y);
    square = Square(r, x, y);
}
```

Приложение Б. Графическая часть

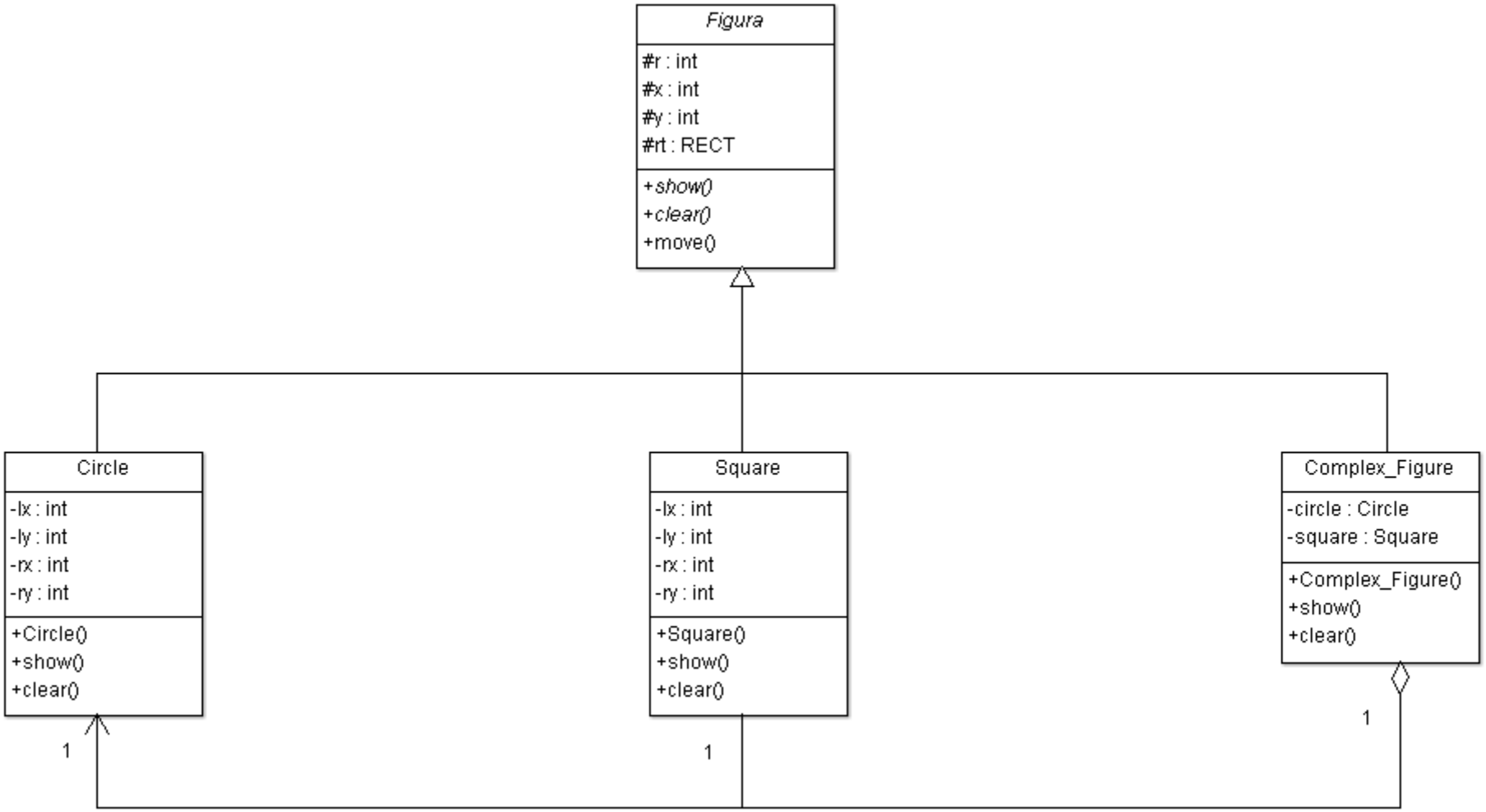

Рисунок Б1 – Диаграмма классов

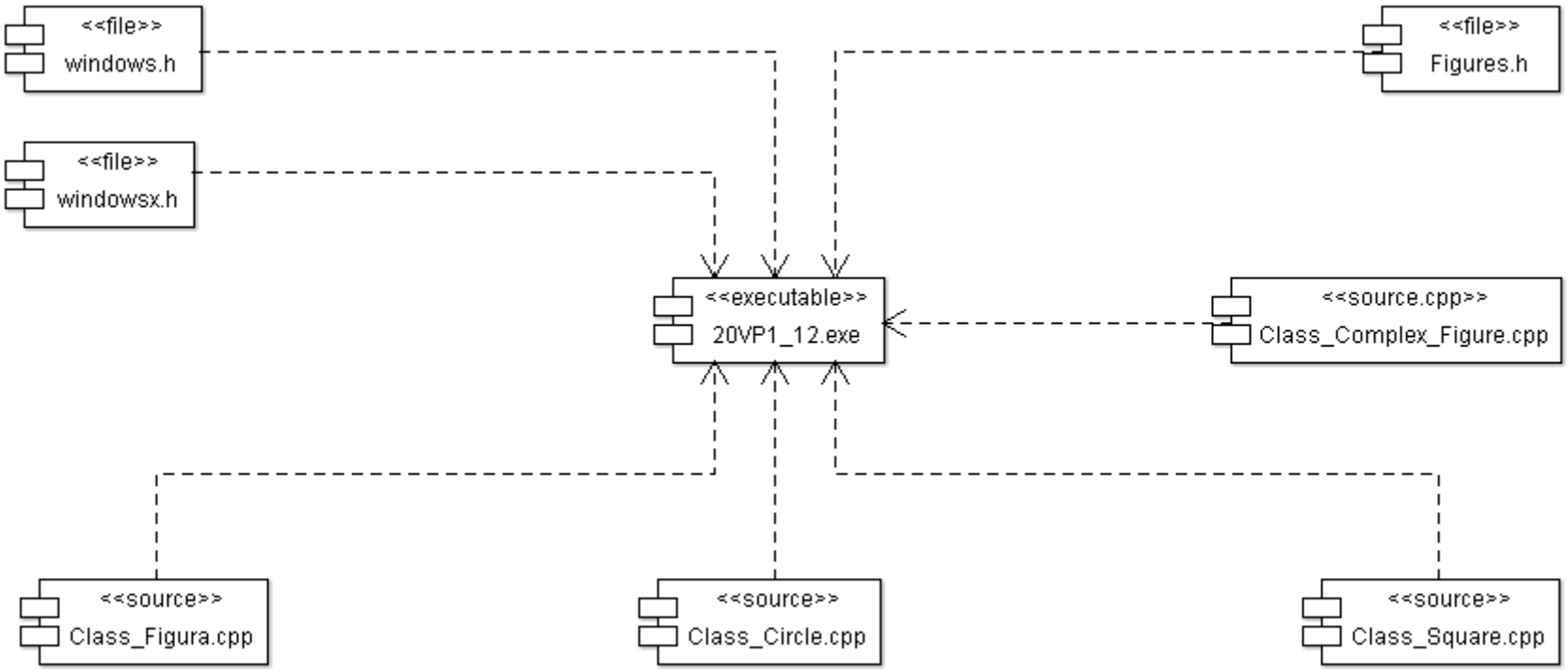

Рисунок Б2 – Диаграмма компонентов