

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

Курсовой Проект
по дисциплине «Программирование»
на тему «Приложение C++. Классы: прямоугольник, треугольник»

ПГУ 09.03.04 – 02КП232. 18 ПЗ
Направление подготовки – 09.03.04 «Программная инженерия.»

Выполнил студент:	Осин Сергей Михайлович
Группа:	23ВП2
Руководитель:	
к.т.н., доцент	Гурьянов Л. В.

Проект защищен с оценкой
Преподаватели

отлично
Л. В. Гурьянов

Дата защиты

31.05.2024

Пенза, 2024.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Математическое обеспечение и применение ЭВМ»

«УТВЕРЖДАЮ»

Зав. кафедрой МОиПЭВМ

 А.Ю.Козлов

ЗАДАНИЕ

на курсовой проект


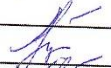
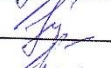


по дисциплине «Программирование»

на тему: «Приложение С++. Классы: прямоугольник, треугольник»

1. Студент гр.	23ВП2	факультета ВТ	направления 09.03.04
Осин С.М.			
2. Руководитель работы	Гурьянов Лев Вячеславович		
3. Время проектирования	с «15» февраля 2024	по «30» мая 2024	
4. Тема проекта:	Приложение С++. Классы: прямоугольник, треугольник		
5. Техническое задание на курсовую работу (назначение, технические требования)			
<u>Назначение:</u> создание и визуализация следующих типов фигур:			
прямоугольник, треугольник, сложная фигура			
<u>Основные функции приложения:</u>			
а) создать геометрическую фигуру;			
б) показать/скрыть фигуру;			
в) переместить фигуру;			
г) добавить фигуру в контейнер;			
д) показать фигуры из контейнера:			
е) удалить контейнер			
<u>Структура данных и программных средств:</u> очередь как двусторонняя очередь,			
базовый класс, композиция			
<u>Технология разработки:</u> ООП			
<u>Язык программирования:</u> С++			
<u>Среда исполнения:</u> консольное приложение 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	1.03.2024	
2) Проектирование	20	28.03.2024	
3) Реализация	30	20.04.2024	
4) Тестирование	20	1.05.2024	
5) Оформление пояснительной записки	20	15.05.2024	

Дата выдачи задания «15» февраля 2024г.

Руководитель курсового проекта Гурьянов Лев Вячеславович

Задание к исполнению принял «15» февраля 2024г.

Студент Осин Сергей Михайлович

Оглавление

Введение.....	5
1 ПРИЛОЖЕНИЕ C++. КЛАССЫ: ПРЯМОУГОЛЬНИК, ТРЕУГОЛЬНИК.....	6
1.1 Анализ предметной области.....	6
1.2 Анализ функциональных требований.....	8
1.3 Проектирование.....	10
1.4. Реализация.....	14
1.5 Тестирование.....	16
Заключение	23
Список использованных источников.....	24
Приложение А.....	25
Приложение Б.	32

Введение.

Основной задачей курсового проекта является разработка консольного приложения для рисования фигур с использованием Объектно-ориентированного программирования на Языке Программирования C++ в приложении Microsoft Visual Studio 2022.

Целью курсового проектирования является формирование и овладение умениями и навыками самостоятельной организации учебно-исследовательской работы и применения полученных знаний для решения профессиональных задач.

Основные этапами разработки приложения являются:

- анализ предметной области;
- анализ функциональных требований;
- проектирование программного обеспечения;
- реализация;
- тестирование;

В процессе разработки консольного приложения был применён язык моделирования **UML** (Unified Modeling Language), используемый в приложении Visual Paradigm 17.1 Community Edition.

Средствами разработки были: Язык Программирования C++ 20 и среда разработки – Visual Studio 2022.

В конце курсового проекта присутствуют 2 приложения: А, В. Приложение А содержит код программ, из которых строится приложение. Приложение В является графической частью приложения, в котором содержатся диаграммы классов и компонентов, которые были разработаны в ходе проектирования приложения.

1 ПРИЛОЖЕНИЕ C++. КЛАССЫ: ПРЯМОУГОЛЬНИК, ТРЕУГОЛЬНИК.

1.1 Анализ предметной области.

Предметная область разработки включает геометрические фигуры, указанные в варианте технического задания: треугольник, прямоугольник и сложная фигура. Предметная область приведена на рисунке 1.

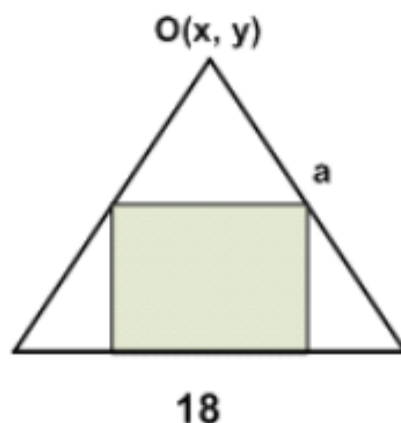


Рисунок 1 – Заданная Геометрическая Фигура

Проанализировав предметную область, получаем следующую сложную фигуру, представленную на рисунке 2.

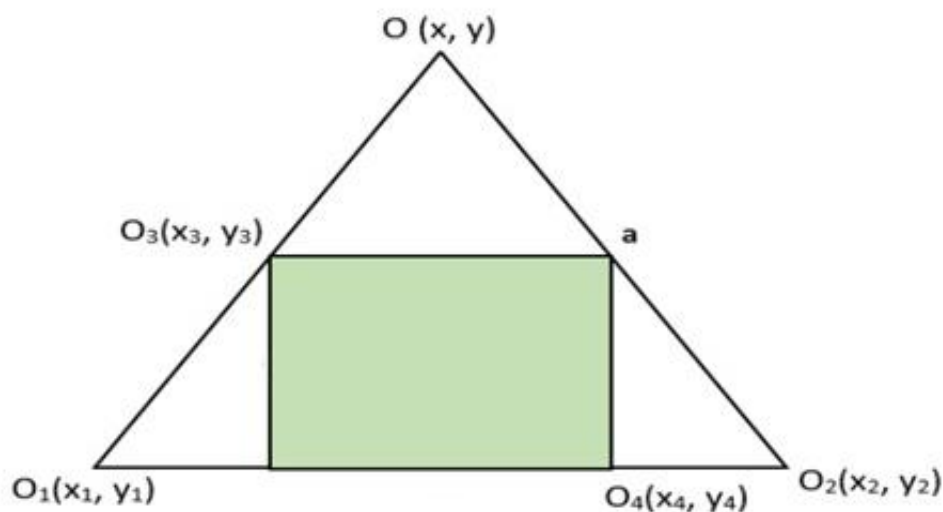


Рисунок 2 - Анализ предметной области.

Сложная фигура состоит из равнобедренного треугольника и вписанного в него прямоугольника. Равнобедренный треугольник представлен 3 вершинами: $O(x, y)$, $O_1(x_1, y_1)$, $O_2(x_2, y_2)$, а также длиной боковых сторон a . Вписанный прямоугольник строится из координат верхнего левого угла $O_3(x_3, y_3)$ и правого нижнего угла $O_4(x_4, y_4)$.

Проведём вычисления координат вершин треугольника и прямоугольника, исходя из заданных параметров:

а) Вычислим координаты двух оставшихся вершин треугольника, используя заданные координаты точки $O(x, y)$ и длиной боковых сторон a :

$$O_1(x_1, y_1) = O_1(x + a, y + a) \quad (1)$$

$$O_2(x_2, y_2) = O_2(x - a, y + a) \quad (2)$$

б) Для построения прямоугольника проведём вычисления координат левой верхней и правой нижней вершин:

$$O_3(x_3, y_3) = O_3\left(x - \frac{a}{2}, y + \frac{a}{2}\right) \quad (3)$$

$$O_4(x_4, y_4) = O_4\left(x + \frac{a}{2}, y + a\right) \quad (4)$$

Модель предметной области была построена на основе анализа предметной области (рисунок 2.), представлена на рисунке 3.



Рисунок 3 – Модель предметной области

В данной модели сложная фигура состоит из равнобедренного треугольника и вписанного в него прямоугольника. Треугольник определяется вершинами $O(x, y)$, $O_1(x_1, y_1)$, $O_2(x_2, y_2)$, по которым строится треугольник и длиной боковых сторон a . Вписанный прямоугольник строится по левой верхней вершине $O_3(x_3, y_3)$ и правой нижней вершине $O_4(x_4, y_4)$.

1.2 Анализ функциональных требований.

Основными функциями приложения являются:

- Создать геометрическую фигуру;
- Показать/Скрыть фигуру;
- Переместить фигуру;
- Добавить фигуру в контейнер;
- Показать фигуру из контейнера;
- Удалить контейнер.

Была построена Диаграмма Вариантов Использования. На диаграмме варианты «Скрыть треугольник» и «Показать треугольник» являются расширением варианта «Переместить треугольник». Аналогичные варианты использования и у Прямоугольника, и у Сложной Фигуры

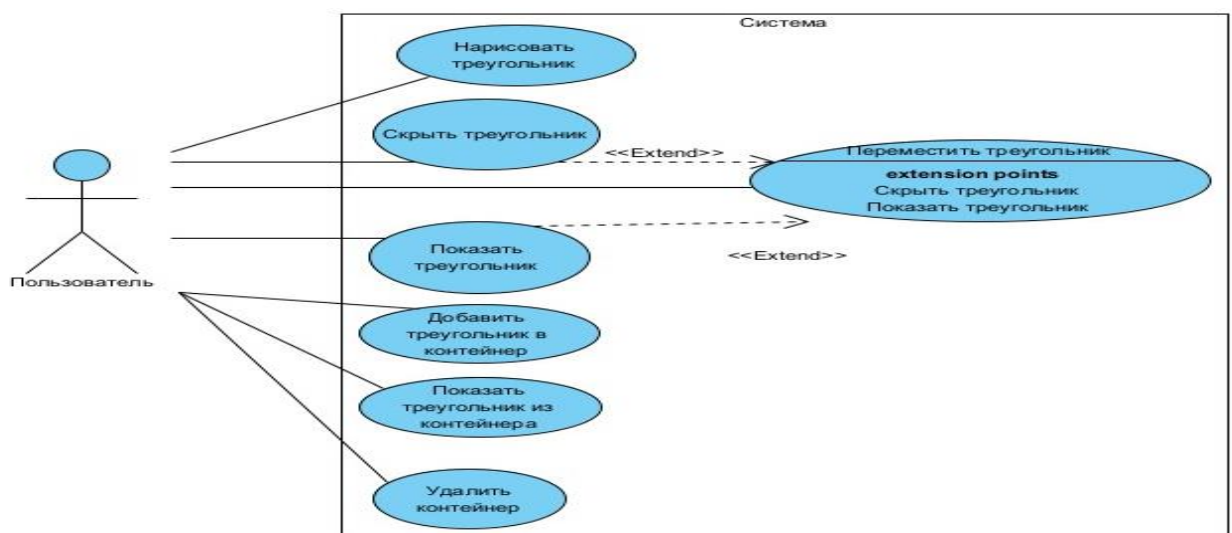


Рисунок 4 – Диаграмма Вариантов Использования

Сценарии вариантов «Переместить треугольник» и «Показать фигуру из контейнера» приведены в таблицах 1,2.

Таблица 1 – Сценарий варианта использования «Переместить треугольник»

Наименование: переместить треугольник.
ID: 3
Краткое описание: пользователь задаёт координаты для перемещения треугольника, которые сначала система проверяет корректность введенных значений.
Действующие лица: пользователь
Предусловие: треугольник создан и нарисован
Основной поток: <ol style="list-style-type: none"> 1. Пользователь задаёт координаты перемещения треугольника. <ol style="list-style-type: none"> 1.1. Если координаты выходят за пределы окна ИЛИ координаты меньше 0, то фигура не сможет переместиться на данные координаты. 1.2. Иначе: <ol style="list-style-type: none"> 1.2.1. Треугольник перемещается на заданные координаты с тем же размером фигуры.
Постусловие: треугольник перемещён на заданные координаты

Таблица 2 – Сценарий варианта использования «Показать фигуры из контейнера»

Наименование: показать фигуры из контейнера
ID: 6.
Краткое описание: система проверяет, пуст ли контейнер, затем выводит на экран консоли все имеющиеся в контейнере фигуры
Действующие лица: система
Предусловие: контейнер не пуст
Основной поток: <ol style="list-style-type: none"> 1. Система проверяет контейнер на заполненность. <ol style="list-style-type: none"> 1.1. Если контейнер пуст, то никакая фигура не выводится на экран, а в консоли появляется надпись «Queue is empty». 1.2. Иначе: <ol style="list-style-type: none"> 1.2.1. Система выводит на экран консоли фигуры с записью их координат в левом верхнем углу
Постусловие: фигура вывелась на экран

1.3 Проектирование.

Спроектируем диаграмму классов, созданные при разработки консольного приложения, показанных на рисунке 5.

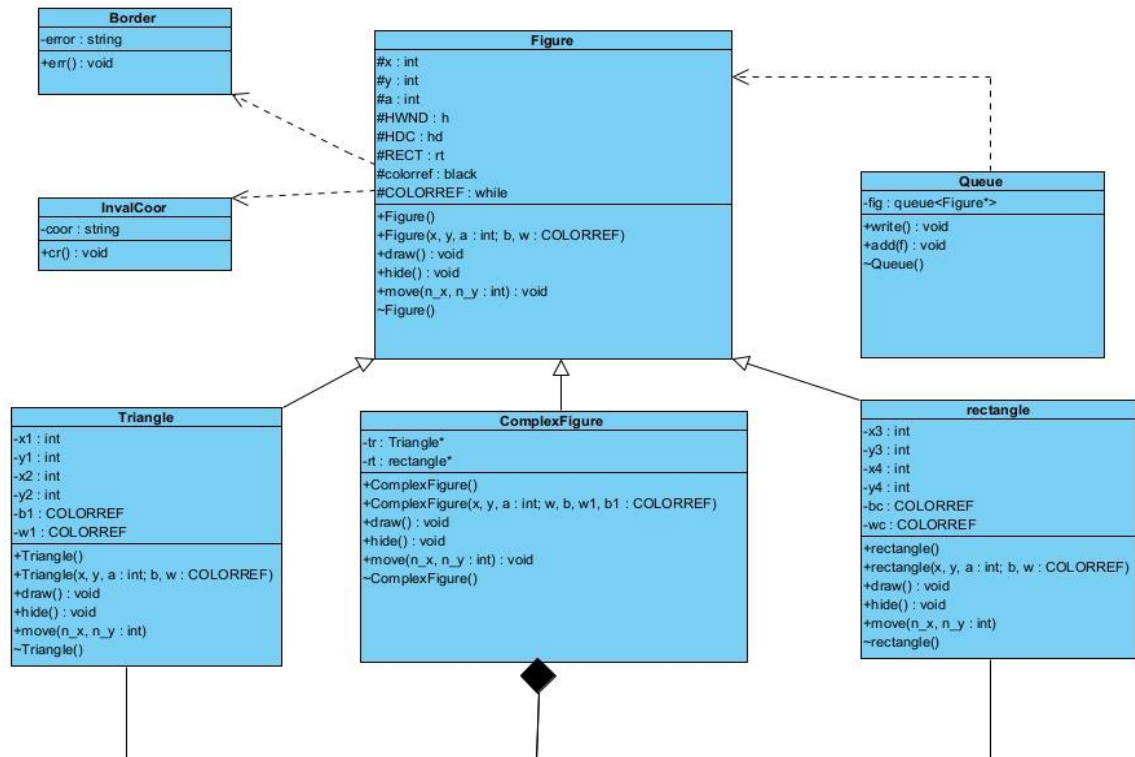


Рисунок 5 – Диаграмма Классов

Данная диаграмма классов представляет собой множество классов, главный из которых – **Figure**. Он является базовым, в нём содержится общие для всех классов объекты, функции. Классы **Triangle** и **rectangle**, **ComplexFigure** являются соответствующими фигурами: Треугольник и Прямоугольник, Сложная фигура и наследуют от класса **Figure** атрибуты и операции. Класс **ComplexFigure** включает объекты классов **Triangle** и **rectangle**. Классы **Border** и **InvalCoor** являются классами обработки исключения, первый в данном случае следит, чтобы нарисованная пользователем фигура не выходила за границы окна консольного приложения, а вторая – чтобы координаты и длины сторон не были отрицательными. Класс **Queue** представлен контейнерным классом **queue**, который используется для хранения фигур. Пользователь может добавить фигуру в контейнер, вывести содержимое контейнера и удалить контейнер.

Спецификация класса **Figure**.

Название класса: **Figure**.

Назначение: базовый класс Фигура.

Члены класса:

- x, y: int** – координаты вершины.
- a: int** – размер фигуры.
- h: HWND** – идентификатор окна;
- hd: HDC** – контекст изображения;
- rt: RECT** – координаты прямоугольного окна;
- black: COLORREF** – заливка фигуры;
- white: COLORREF** – цвет границы фигуры;
- Border: Figure** – класс исключения
- InvalCoor: Figure** – класс исключения.

Функции класса:

- Figure()** – конструктор по умолчанию;
- Figure(int,int,int,COLORREF,COLORREF)** – конструктор с координатами вершины и стороной фигуры, цветом заливки и границ;
- draw()** – нарисовать фигуру.
- hide()** – скрыть фигуру.
- move(int, int)** – переместить фигуру.
- ~Figure()** – деструктор.

Спецификация класса **Triangle**.

Название класса: **Triangle**.

Назначение: класс Треугольник, наследник класса **Figure**.

Члены класса:

- x1,y1: int** – координаты левой нижней вершины Треугольника.
- x2,y2: int** – координаты правой нижней вершины Треугольника.
- b1,w1: COLORREF** – цвета линии и заливки Треугольника.

Функции класса:

- Triangle()** – конструктор по умолчанию;

-Triangle(int, int,int, COLORREF, COLORREF) – конструктор с координатами вершины и стороной треугольника, цветом границ и заливки;

-draw() – нарисовать фигуру.

-hide() – скрыть фигуру.

-move(int, int) – переместить фигуру на заданные координаты.

--Triangle() – деструктор.

Спецификация класса rectangle.

Название класса: rectangle.

Назначение: класс прямоугольник, наследник класса **Figure**.

Члены класса:

-x3, y3: int – координаты левой верхней вершины прямоугольника.

-x4,y4: int – координаты правой нижней вершины прямоугольника.

-bc, wc: COLORREF – цвета границы и заливки.

Функции класса:

-rectangle() – конструктор по умолчанию.

- rectangle (int,int, int, COLORREF, COLORREF) – конструктор с координатами левого верхнего угла и стороной прямоугольника, цветом границ и заливки.

-draw() – нарисовать прямоугольник.

-hide() – скрыть прямоугольник.

-move(int, int) – переместить прямоугольник на заданные координаты.

--rectangle() – деструктор.

Спецификация класса ComplexFigure.

Название класса: ComplexFigure.

Назначение: класс Сложная фигура, наследник класса **Figure**.

Члены класса:

-tr: Triangle* – треугольник;

-rt: rectangle* – прямоугольник;

Функции класса:

-ComplexFigure() – конструктор по умолчанию.

-ComplexFigure(int,int, int, COLORREF, COLORREF, COLORREF, COLORREF) – конструктор с координатами вершинами сложной фигуры, размером фигуры, цветом границ и заливки треугольника и прямоугольника.

-draw() – нарисовать сложную фигуру;

-hide() – скрыть сложную фигуру;

-move(int, int) – переместить сложную фигуру на заданные координаты.

--ComplexFigure () – деструктор.

Спецификация класса Queue

Название класса: Queue.

Назначение: класс хранения фигур.

Члены класса:

-fig: Queue<Figure*> - динамический массив.

Функции класса:

-write() – вывод всех фигур, находящихся в контейнере, на экран.

-add(f) – добавить фигуру f в контейнер.

--Queue() – деструктор.

Спецификация класса Border.

Название класса: Border.

Назначение: класс обработки исключения выхода фигуры за границы консоли.

Члены класса:

- error: string – сообщение об нарушении.

Функции класса:

-Border() – инициализация переменной **error** сообщением об ошибке;

-error() – вывод сообщения об ошибке;

--Border() – деструктор.

Спецификация класса InvalCoor.

Название класса: InvalCoor.

Назначение: класс обработки исключения инициализации переменных отрицательными значениями.

Члены класса:

-**coor: string** – сообщение об ошибке.

Функции класса:

-**cr()** – инициализация переменной **coor** сообщением об ошибке;

-**coor()** – вывод сообщения об ошибке;

~**InvalCoor()** – деструктор.

1.4. Реализация.

На данном этапе реализуем диаграмму компонентов, содержащую классы, спроектированные ранее. Она определяет, как проектные элементы развертываются на узлах вычислительной системы.Arteфакты представляют описания реальных программных сущностей, а узлы представляют описания или сред выполнения, в которых эти сущности развёртываются. Диаграмма компонентов представлена на рисунке 6.

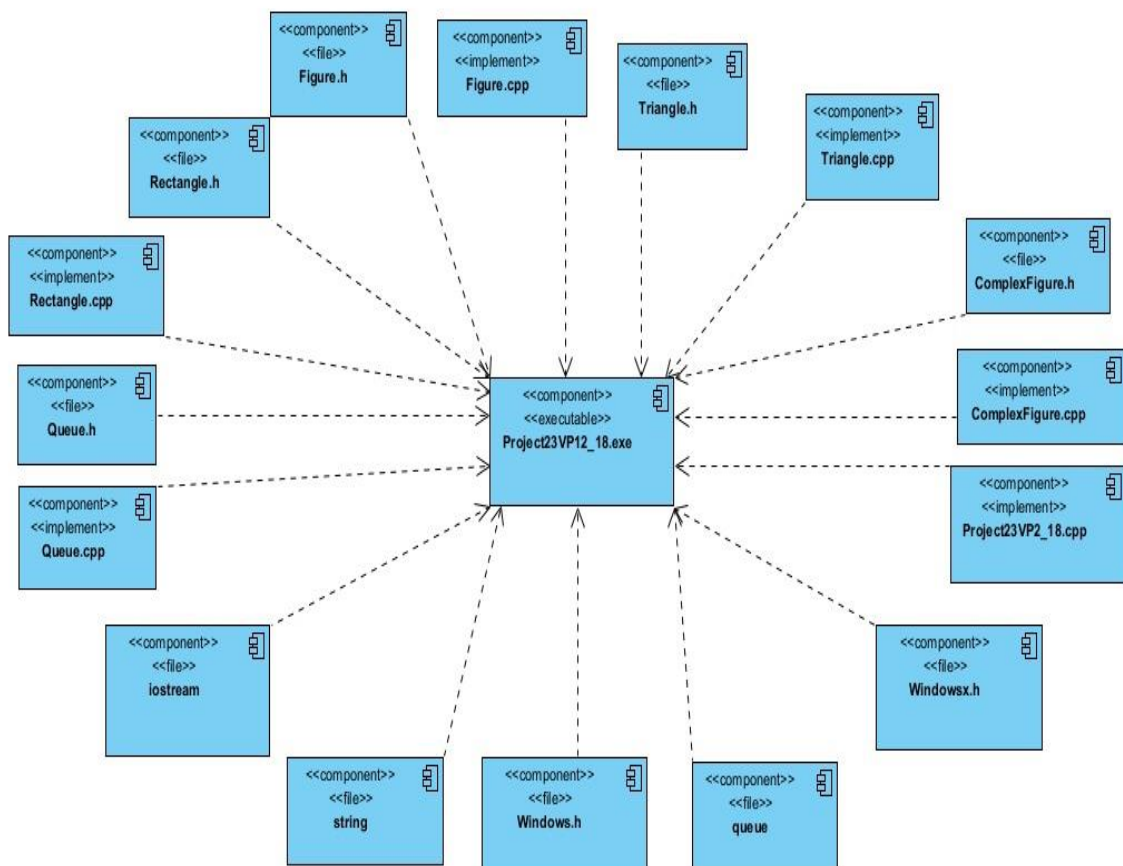


Рисунок 6 – Диаграмма компонентов

Рассмотрим каждый компонент в таблице 3:

Таблица 3 – Компоненты приложения.

Компонент	Назначение
Figure.h	Заголовочный файл класса Figure
Figure.cpp	Исходный файл класса Figure
Triangle.h	Заголовочный файл класса Triangle
Triangle.cpp	Исходный файл класса Triangle
Rectangle.h	Заголовочный файл класса rectangle
Rectangle.cpp	Исходный файл класса rectangle
Queue.h	Заголовочный файл класса Queue
Queue.cpp	Исходный файл класса Queue
ComplexFigure.h	Заголовочный файл класса ComplexFigure
ComplexFigure.cpp	Исходный файл класса ComplexFigure
Windows.h	Системный заголовочный файл
Windowsx.h	Системный заголовочный файл
iostream	Системный заголовочный файл
string	Системный заголовочный файл

Окончание таблицы 3

Queue	Системный заголовочный файл.
Project23VP2_18.cpp	Главная программа
Project23VP2_18.exe	Исполняемый файл приложения

1.5 Тестирование.

Завершающий этап разработки консольного приложения – **тестирование**. Тестирование пройдёт у всех фигур, заданных в модели предметной области. Главная задача тестирования – показать функции, указанные на Рисунке 2, в действии в консольном приложении. Пусть будут заданы цвета у Треугольника: **RGB** (0,0,0), **RGB**(255, 255, 255) и у Прямоугольника: **RGB**(0, 0, 0), **RGB**(0, 255, 0). Результаты тестирования будут представлены в таблице 4 и на рисунках 7–13.

Таблица 4 – Тестирование.

Вариант использования	Тест	Результат
Показать треугольник	Тест 1. Координаты вершины: 380,250. Длина стороны: 100.	Тест пройден. Треугольник создан. (Рисунок 7.)
	Тест 2. Координаты вершины: 850,250 Длина стороны: 100	Тест пройден. Выход за пределы окна. (Рисунок. 8)
	Тест 3. Координаты вершины: 600,540. Длина стороны: -150	Тест пройден. Ошибка: отрицательные координаты/размеры фигуры.
Скрыть треугольник	Тест 1. Координаты вершины: 380,250. Длина стороны: 100.	Тест пройден. Треугольник был скрыт. (Рисунок 11.).

Продолжение таблицы 4

	Тест 2. Координаты вершины: 850,250 Длина стороны: 100	Тест пройден. Выход за пределы окна.
	Тест 3. Координаты вершины: 600,540 Длина стороны: -150	Тест пройден. Отрицательные координаты/длины сторон.
Переместить треугольник	Тест 1. Координаты перемещения: 380,350. Длина стороны: 100.	Тест пройден. Треугольник был перемещён на заданные координаты. (Рисунок 12).
	Тест 2 Координаты вершины: 720,290. Длина стороны: 150	Тест пройден. Треугольник был перемещён на выбранные координаты.
	Тест 3 Координаты вершины: -520,790. Длина стороны: 90	Тест пройден. Выход за пределы границы окна и отрицательная координата.
Показать прямоугольник	Тест 1. Координаты вершины: 540,250. Длина стороны: 100.	Тест пройден. Прямоугольник был нарисован. (Рисунок 7).
	Тест 2. Координаты вершины: 950,150 Длина стороны: 130	Тест пройден. Выход за границы окна. (Рисунок. 9)

Продолжение таблицы 4

Скрыть прямоугольник	Тест 1. Координаты вершины: 450,100 Длина стороны: 80	Тест пройден. Прямоугольник был скрыт. (Рисунок. 12)
	Тест 2. Координаты вершины: 700,250. Длина стороны: 100	Тест пройден. Прямоугольник перемещен правильно.
	Тест 3. Координаты вершины: 800,500. Длина стороны:150	Тест пройден. Ошибка: Выход за пределы окна.
Переместить прямоугольник	Тест 1 Координаты вершины: 545, 350. Длина стороны: 100.	Тест пройден. Прямоугольник был перемещён на нужные координаты.
	Тест 2 Координаты вершины: 720,290. Длина стороны: 150	Тест пройден. Фигура была нарисована и через некоторое время была скрыта.
	Тест 3 Координаты вершины: -520,790. Длина стороны: 90	Тест пройден. Ошибка: отрицательные координаты/длины сторон.

Продолжение таблицы 4

Показать сложную фигуру	Тест 1 Координаты вершины: 690,250. Длина стороны: 100.	Тест пройден. Сложная фигура была нарисована. (Рисунок 7.)
	Тест 2 Координаты вершины: 800,200. Длина стороны: 150	Тест пройден. Ошибка: выход за пределы окна. (Рисунок. 10)
	Тест 3 Координаты вершины: 750,130. Длина стороны: -100.	Тест пройден. Ошибка: отрицательные координаты/длина.
Скрыть сложную фигуру	Тест 1 Координаты вершины: 750,150. Длина стороны: 100.	Тест пройден. Сложная фигура была скрыта. (Рисунок. 12)
	Тест 2 Координаты вершины: 800,200. Длина стороны: 150	Тест пройден. Ошибка: выход за пределы окна.
	Тест 3 Координаты вершины: 750,130. Длина стороны: -100.	Тест пройден. Ошибка: отрицательные координаты/длины сторон.
Переместить сложную фигуру	Тест 1 Точки перемещения: 695,360. Длины стороны: 100.	Тест пройден. Сложная фигура была перемещена на данные координаты. (Рисунок 13.)
	Тест 2 Точки перемещения: 980,220. Длины стороны: 100.	Тест пройден. Ошибка: выход за пределы окна.

Окончание таблицы 4

	Тест 3 Точки перемещения: -100,200. Длины стороны: 100.	Тест пройден. Ошибка: отрицательные координаты/длины сторон.
--	---	---

Результаты тестирования (рисунок 7–13):

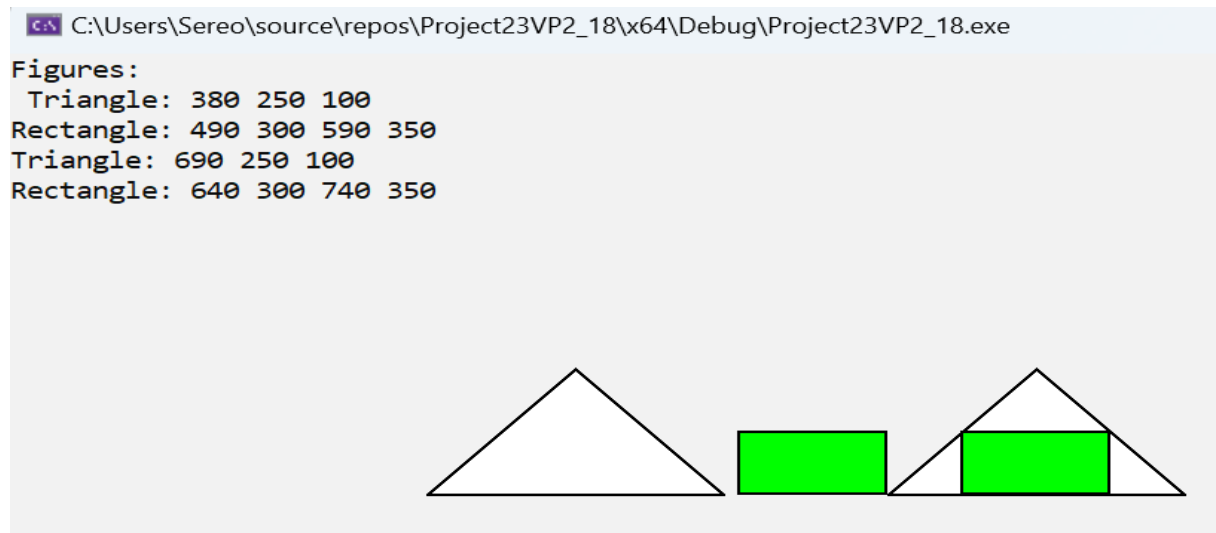


Рисунок 7 - Тест 1 «Показать фигуры».

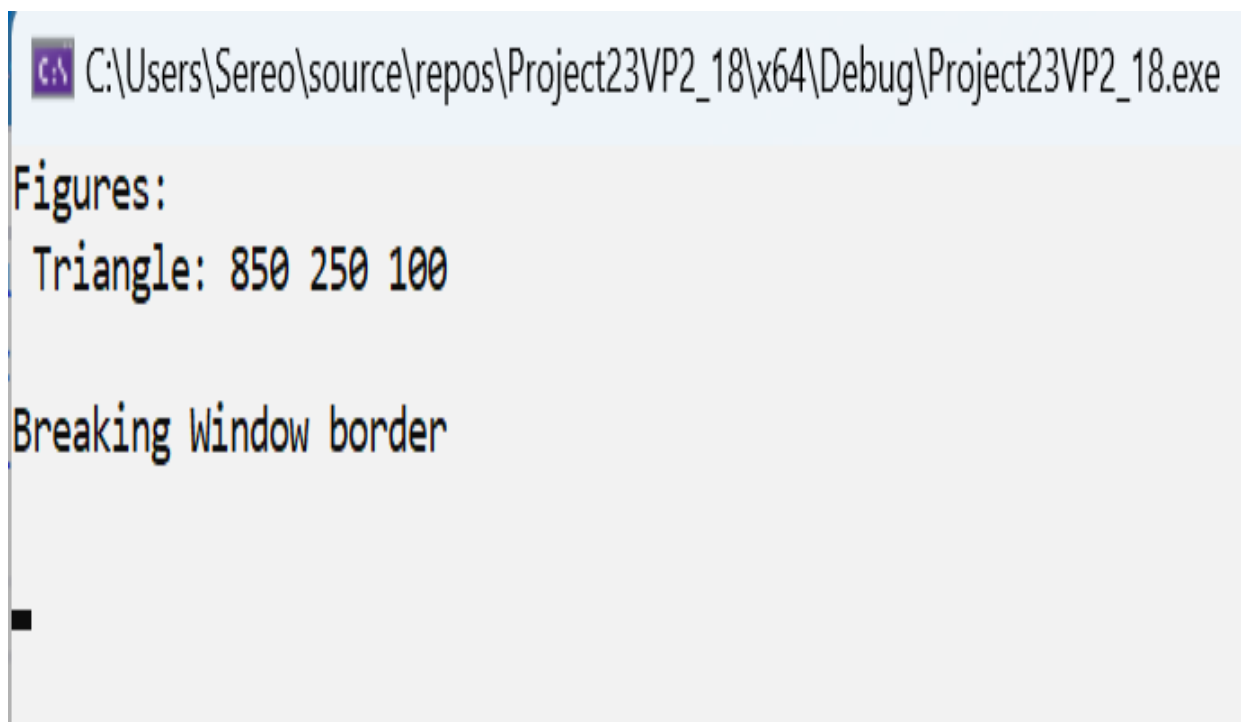


Рисунок 8 - Тест 2 у Треугольника «Показать фигуру»

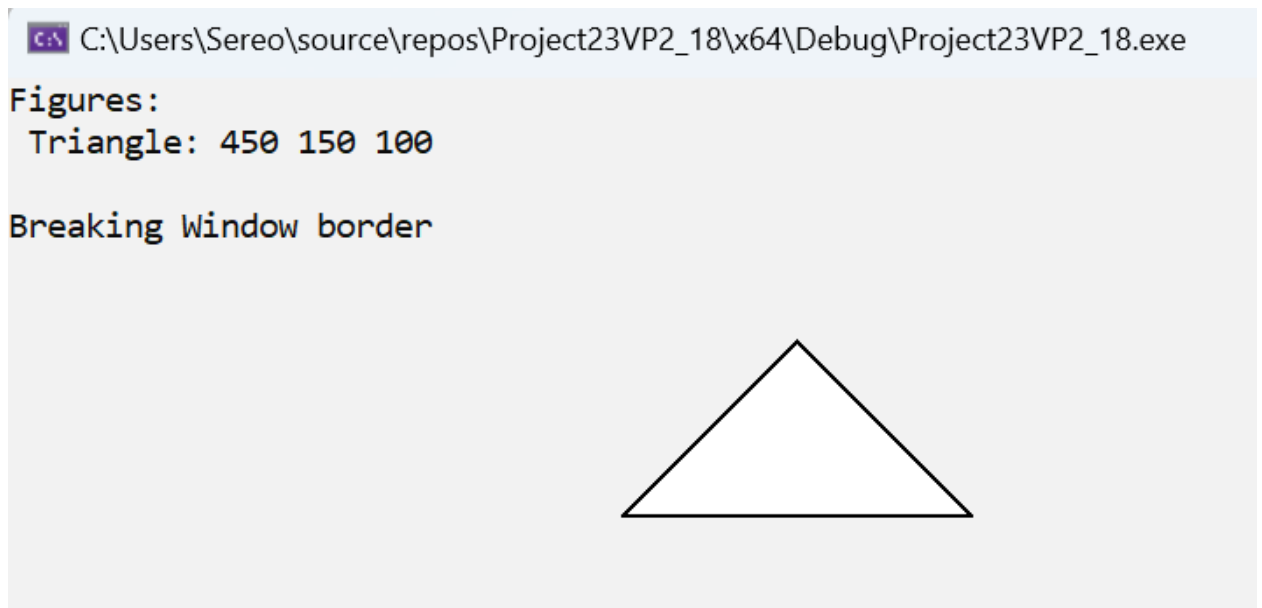


Рисунок 9 – Тест 2 у Прямоугольника «Показать фигуру»

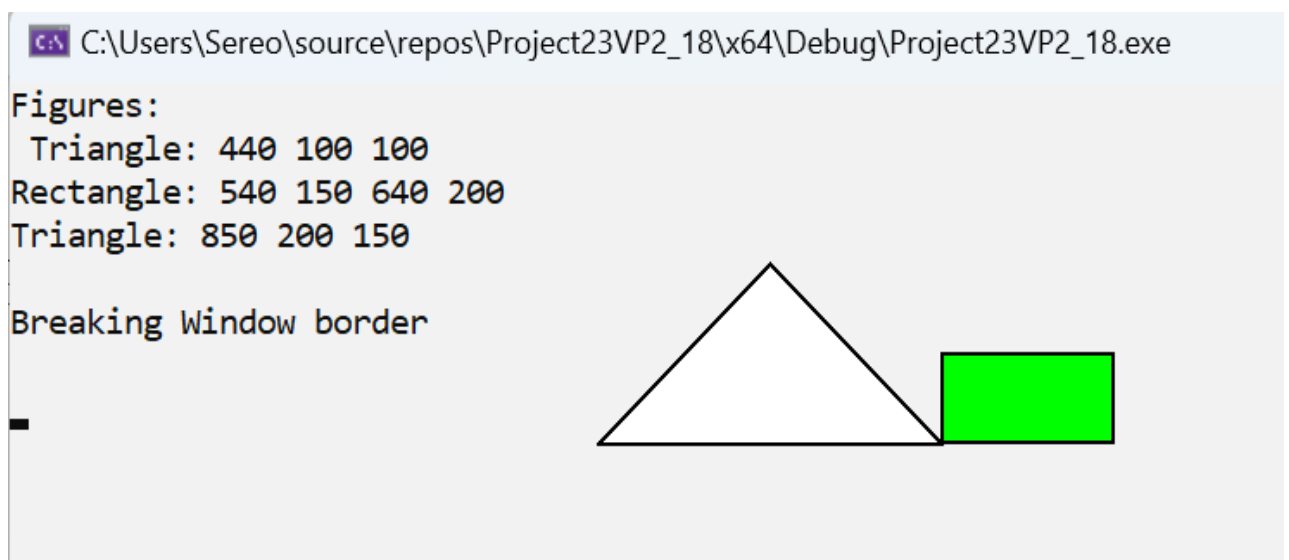


Рисунок 10 – Тест 2 у Сложной Фигуры «Показать фигуру»

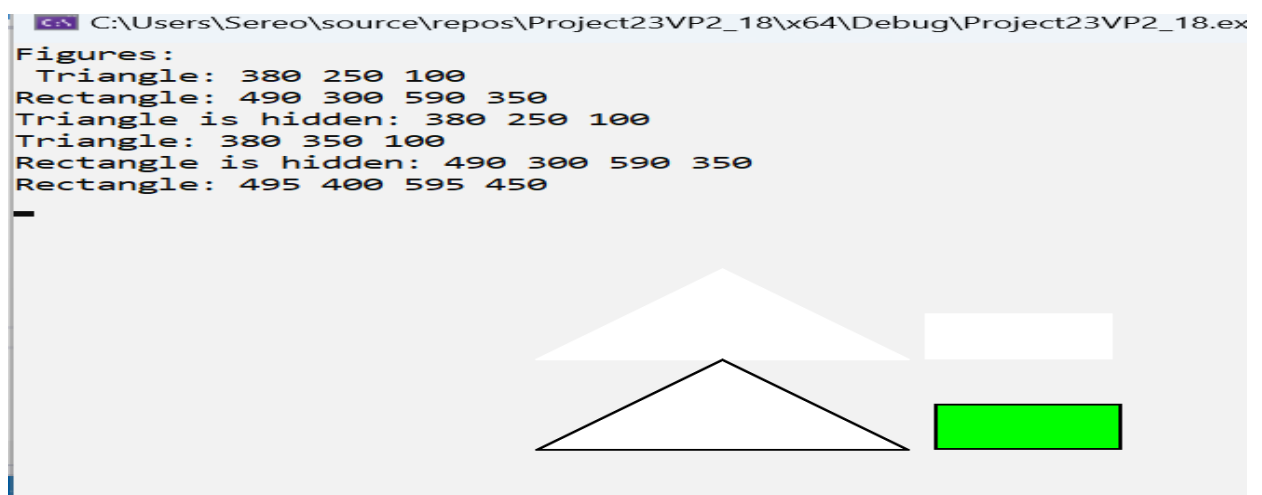


Рисунок 11 – Тест 1 у Треугольник и Прямоугольника «Переместить фигуру»

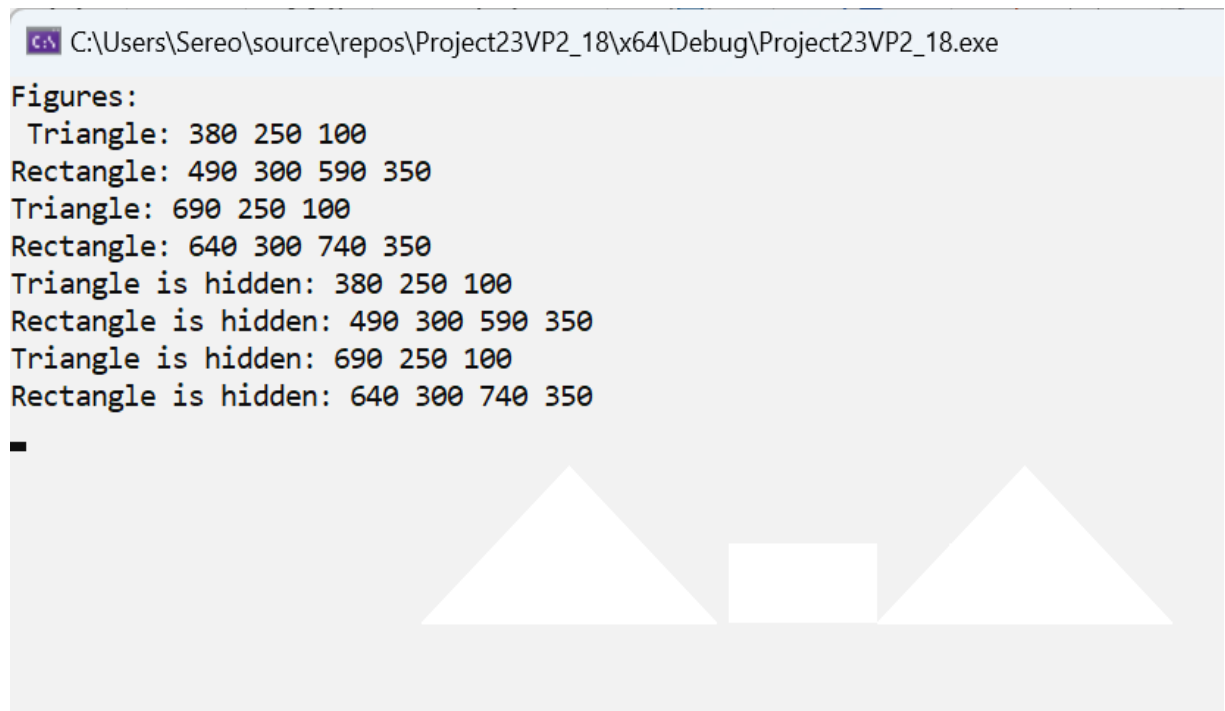


Рисунок 12 – Тест 1 «Скрыть фигуры»

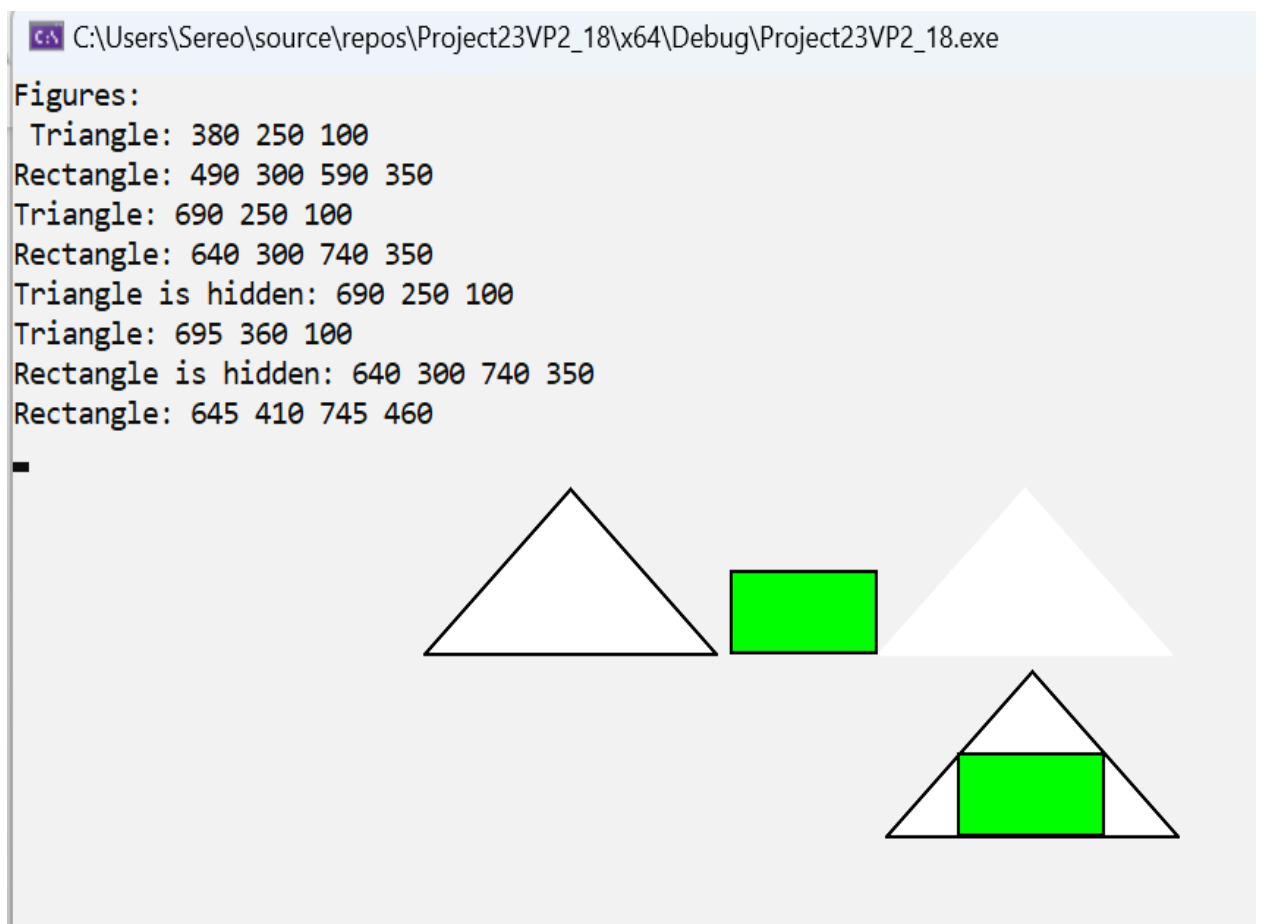


Рисунок 13 - Тест 1 у Сложной фигуры «Переместить фигуру»

Заключение

В результате выполнения курсового проекта удалось успешно разработать консольное приложение, используя объектно-ориентированный подход к программированию.

На этапе «анализ предметной области» была проведена детальная работа по изучению сложной фигуры, состоящей из треугольника и вписанного в него прямоугольника, и на основе этого была спроектирована модель предметной области. Была построена диаграмма вариантов использования, которая позволила чётко определить функциональность консольного приложения: нарисовать, показать/скрыть, переместить, добавить в контейнер, показать из контейнера, удалить из контейнера фигуры.

В процессе проектирования была разработана диаграмма классов с атрибутами и конструкторами: Figure, Triangle, rectangle, ComplexFigure, Border, InvalCoor. Оно способствовало более эффективной имплементации кода.

На этапе «реализация» была разработана диаграмма компонентов, которая даёт полное представление обо всех файлах, присутствующих в процессе разработки приложения: Figure, Triangle, rectangle, ComplexFigure, Queue, Project23VP2_18.

Завершающий этап тестирования позволил проверить функционал программы в различных сценариях. Были проведены тесты с различными вариантами построения треугольника, сложной фигуры и прямоугольника: показать, скрыть, переместить. В результате фигуры либо успешно появлялись на консоли, либо не создавались из-за возникновения исключительной ситуации.

Список использованных источников

1. Арлоу Дж., Нейштадт А. UML2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование. – СПб. : Символ-Плюс, 2007. – 624 с.
2. Л. В. Гурьянов, Л. С. Гурьянова, Е. А. Дзюба [и др.]. Программирование на языке C++. Курсовое проектирование : учеб. пособие. – Пенза : Изд-во ПГУ, 2023. – 76 с.
3. Павловская Т. А. C/C++. Программирование на языке высокого уровня. – СПб. : Питер, 2003. – 461 с.
4. Руководство по языку программирования C++. – URL: <https://metanit.com/cpp/tutorial/> (дата обращения: 15.04.2024)

Приложение А.
КОД ПРОГРАММЫ
(обязательное)

Figure.h

```
#pragma once
#include <iostream>
#include <Windows.h>
#include <windowsx.h>
#include <string>
using namespace std;
class Figure {
protected:
    int x, y;
    int a;
    HWND h;
    HDC hd;
    RECT rt;
    COLORREF black;
    COLORREF white;
public:
    Figure() : a(0), x(0), y(0), h(NULL), hd(NULL), black(0), white(0) {};
    Figure(int _x, int _y, int sh, COLORREF b, COLORREF w) : x(_x), y(_y),
a(sh), black(b), white(w) {
        h = GetConsoleWindow();
        hd = GetDC(h);
        GetClientRect(h, &rt);
    }
    ~Figure() {
        ReleaseDC(h, hd);
    }
    virtual void draw();
    virtual void hide();
    virtual void move(int n_x, int n_y) {
        hide();
        x = n_x;
        y = n_y;
        draw();
    }
    class Border {
    private:
        string error;
    public:
        Border() : error("\nBreaking Window border\n") {};
        void err() {
            cout << error << '\n';
        }
    };
    class InvalCoor {
    private:
        string coor;
    public:
        InvalCoor() : coor("\nCoordinates or sides are less than 0\n") {};
        void cr() {
            cout << coor << '\n';
        }
    };
};
```

Figure.cpp

```
#include "Figure.h"
void Figure::draw() {}
void Figure::hide() {}
```

Triangle.h

```

#pragma once
#include "Figure.h"
class Triangle : public Figure {
public:
    Triangle(int x, int y, int a, COLORREF b, COLORREF w);
    void draw() override;
    void hide() override;
    void move(int n_x, int n_y) override;
private:
    int x1, y1;
    int x2, y2;
    COLORREF b1, w1;
};

```

Triangle.cpp

```

#include "Triangle.h"
#include "Figure.h"
Triangle::Triangle(int x, int y, int a, COLORREF b, COLORREF w) : Figure(x, y, a,
b, w), b1(b), w1(w) {
    x1 = x + a, y1 = y + a; x2 = x - a, y2 = y + a;
}
void Triangle::hide() {
    cout << "Triangle is hidden: " << x << " " << y << " " << a << '\n';
    GetClientRect(h, &rt);
    HPEN p = CreatePen(PS_SOLID, 2, RGB(255,255,255));
    HBRUSH rus = CreateSolidBrush(RGB(255,255,255));
    SelectObject(hd, p);
    SelectObject(hd, rus);
    POINT points[3] = { {x, y}, {x1, y1}, {x2, y2} };
    Polygon(hd, points, 3);
    DeleteObject(p);
    DeleteObject(rus);
}
void Triangle::draw() {
    cout << "Triangle: " << x << " " << y << " " << a << '\n';
    if (x + a > rt.right || y + a > rt.bottom) {
        throw Border();
    }
    if (x < 0 || y < 0 || a <= 0) {
        throw InvalCoor();
    }
    GetClientRect(h, &rt);
    HPEN p = CreatePen(PS_SOLID, 2, b1);
    HBRUSH b = CreateSolidBrush(w1);
    SelectObject(hd, p);
    SelectObject(hd, b);
    POINT points[] = { {x,y}, {x1,y1}, {x2,y2} };
    Polygon(hd, points, 3);
    DeleteObject(p);
    DeleteObject(b);
}
void Triangle::move(int n_x, int n_y) {
    this->hide();
    x = n_x;
    y = n_y;
    x1 = x + a, y1 = y + a;
    x2 = x - a, y2 = y + a;
    this->draw();
}

```

Rectangle.h

```

#pragma once

```

```
#include "Figure.h"
class rectangle : public Figure {
public:
    rectangle(int x, int y, int a, COLORREF b, COLORREF);
    void draw() override;
    void hide() override;
    void move(int n_x, int n_y) override;
private:
    int x3, y3;
    int x4, y4;
    COLORREF bc, wc;
};
```

Rectangle.cpp

```
#include "Rectangle.h"
#include "Figure.h"
rectangle::rectangle(int x, int y, int a, COLORREF b, COLORREF w) : bc(b), wc(w),
    Figure(x, y, a, b, w) {
    int x1 = x + a, y1 = y + a; x2 = x - a, y2 = y + a;
    x3 = x - a / 2; y3 = y + a / 2; x4 = x + a / 2; y4 = y + a;
    y4 = y + a;
}
void rectangle::hide() {
    cout << "Rectangle is hidden: " << x3 << " " << y3 << " " << x4 << " " << y4
    << '\n';
    HPEN p = CreatePen(PS_SOLID, 2, RGB(255,255,255));
    HBRUSH b = CreateSolidBrush(RGB(255,255,255));
    SelectObject(hd, p);
    SelectObject(hd, b);
    Rectangle(hd, x3, y3, x4, y4);
    DeleteObject(p);
    DeleteObject(b);
}
void rectangle::draw() {
    if (x + a > rt.right || x - a < rt.left || y + a > rt.bottom || y - a <
    rt.top) {
        throw Border();
    }
    if (x3 < 0 || y3 < 0 || x4 < 0 || y4 < 0)
        throw InvalCoor();
    if (x < 0 || y < 0 || a <= 0) {
        throw InvalCoor();
    }
    cout << "Rectangle: " << x3 << " " << y3 << " " << x4 << " " << y4 << '\n';
    HPEN p = CreatePen(PS_SOLID, 2, bc);
    HBRUSH b = CreateSolidBrush(wc);
    GetClientRect(h, &rt);
    SelectObject(hd, p);
    SelectObject(hd, b);
    Rectangle(hd, x3, y3, x4, y4);
    DeleteObject(p);
    DeleteObject(b);
}
void rectangle::move(int n_x, int n_y) {
    this->hide();
    x = n_x, y = n_y;
    int x1 = x + a, y1 = y + a; x2 = x - a, y2 = y + a;
    x3 = x - a / 2; y3 = y + a / 2; x4 = x + a / 2; y4 = y + a;
    this->draw();
}
```

ComplexFigure.h

```
#pragma once
#include "Figure.h"
#include "Triangle.h"
```

```

#include "Rectangle.h"
class ComplexFigure : public Figure {
private:
    Triangle* tr;
    rectangle* rc;
public:
    ComplexFigure(int x, int y, int a, COLORREF w, COLORREF c, COLORREF w1,
COLORREF b1);
    ~ComplexFigure();
    void draw() override;
    void hide() override;
    void move(int n_x, int n_y) override;
};

```

ComplexFigure.cpp

```

#include "ComplexFigure.h"
ComplexFigure::ComplexFigure(int x, int y, int a, COLORREF c, COLORREF w,COLORREF
w1,COLORREF b1) {
    tr = new Triangle(x, y, a, c, w);
    rc = new rectangle(x, y, a, b1,w1);
}

void ComplexFigure::hide() {
    tr->hide();
    rc->hide();
}

void ComplexFigure::draw() {
    tr->draw();
    rc->draw();
}

void ComplexFigure::move(int new_x, int new_y) {
    tr->move(new_x, new_y);
    rc->move(new_x, new_y);
}

ComplexFigure::~~ComplexFigure() {
    delete tr;
    delete rc;
}

```

Queue.h

```

#pragma once
#include "Figure.h"
#include <queue>
class Queue {

```



```

private:
    queue<Figure*>fig;
public:
    void write();
    void add(Figure* f);
    ~Queue();
};

```

Queue.cpp

```

#include "Queue.h"
void Queue::add(Figure* f) {
    fig.push(f);
}
void Queue::write() {
    if (fig.empty()) {
        cout << "Queue is empty!";
        return;
    }
    while (!fig.empty()) {
        fig.front()->draw();
        fig.pop();
    }
}
Queue::~Queue() {
    while (!fig.empty())
        fig.pop();
}

```

Project23VP2_18.cpp

```
#include "Rectangle.h"
#include "Triangle.h"
#include "Queue.h"
#include "ComplexFigure.h"
#include "Figure.h"
#include <iostream>
using namespace std;
const int NU = system("color F0");
int main() {
    Figure* s[3];
    s[0] = new Triangle(380, 250, 100, RGB(0,0,0), RGB(255, 255, 255));
    s[1] = new rectangle(540, 250, 100, RGB(0, 0, 0), RGB(0, 255, 0));
    s[2] = new ComplexFigure(750, 250, 100, RGB(0, 0, 0), RGB(255, 255, 255),
    RGB(0, 255, 0), RGB(0,0,0));
    Queue deq;
    cout << "Figures: \n ";
    try {
        for (int i = 0; i < 3; i++) {
            deq.add(s[i]);
        }
        deq.write();
        s[0]->hide();
        Sleep(3500);
        s[2]->move(750, 350);
        Sleep(3500);
    }
    catch (Figure::Border br) {
        br.err();
    }
    catch (Figure::InvalCoor ic) {
        ic.cr();
    }
    Sleep(5000);
    system("pause");
    return 0;
}
```

Приложение Б.
ГРАФИЧЕСКАЯ ЧАСТЬ
(обязательное)

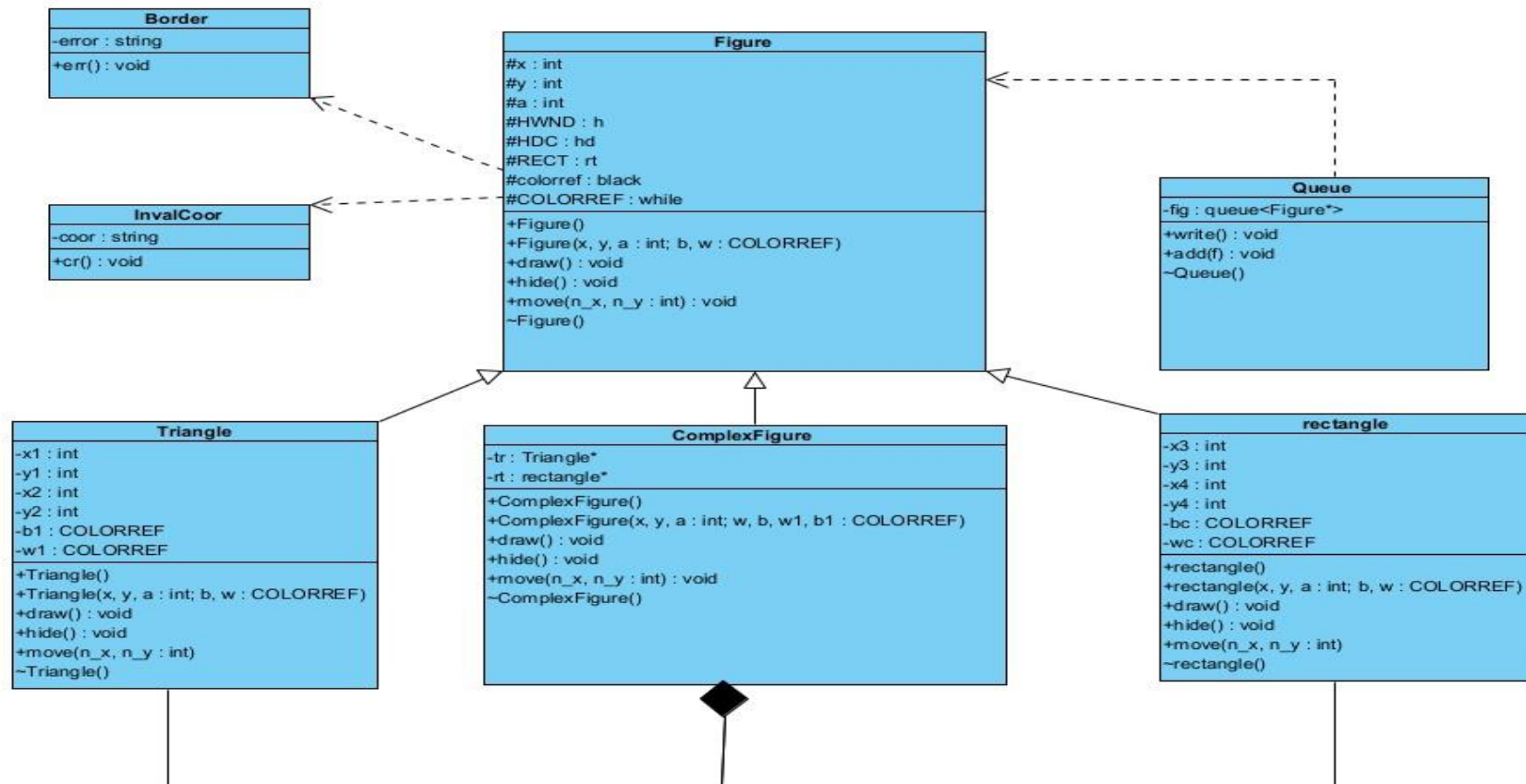


Рисунок Б1 – Диаграмма классов

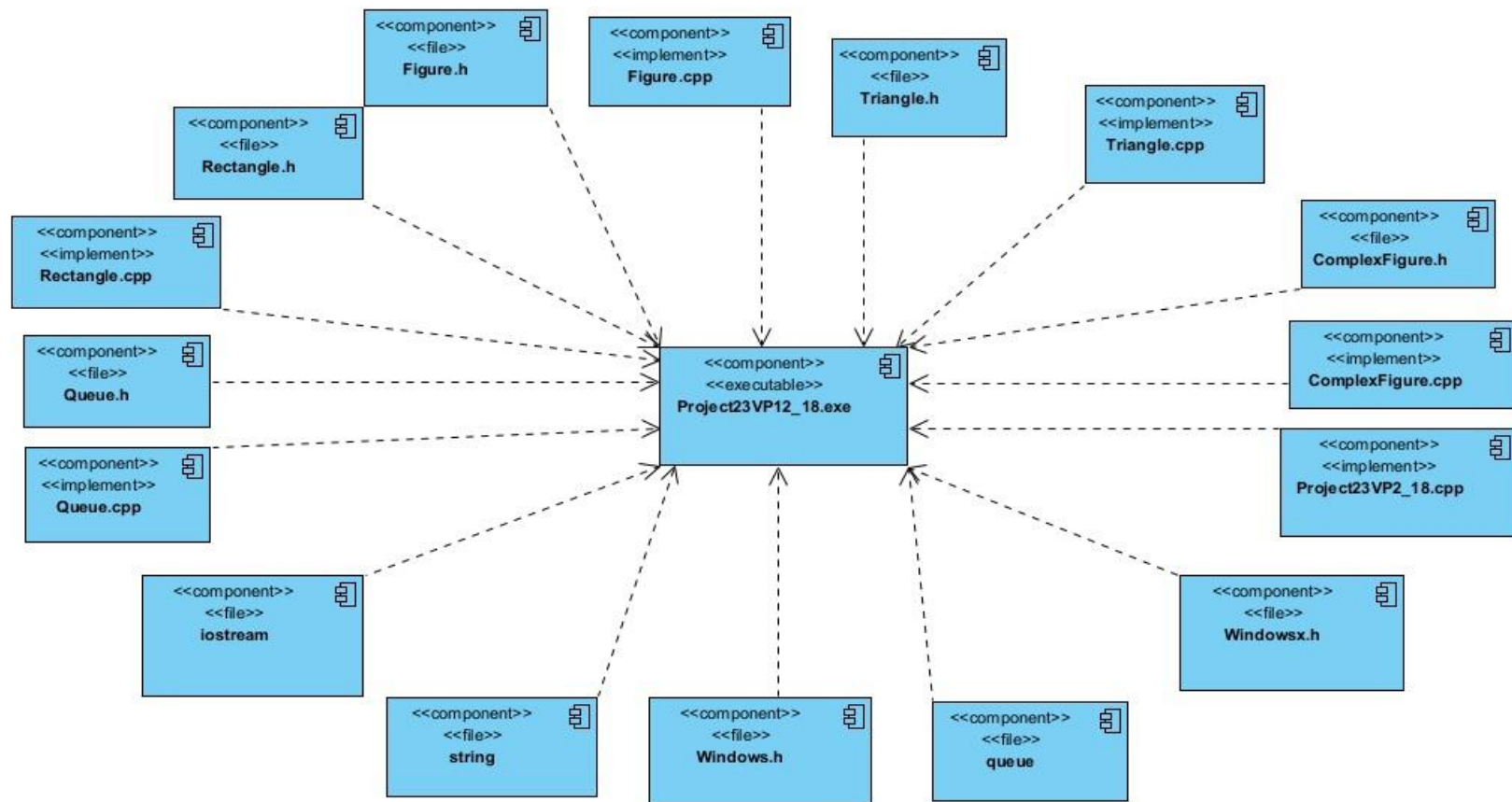


Рисунок Б2 - Диаграмма компонентов