# Содержание

[Содержание 4](#_Toc104319514)

[Введение 5](#_Toc104319515)

[1 РАЗРАБОТКА ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРИЛОЖЕНИЯ. ОКРУЖНОСТЬ В РАВНОБЕДРЕННОМ ТРЕУГОЛЬНИКЕ. 6](#_Toc104319516)

[1.1 Анализ предметной области и функциональных требований 6](#_Toc104319517)

[1.2 Анализ функциональных требований 7](#_Toc104319518)

[1.3. Проектирование 10](#_Toc104319519)

[1.4. Реализация 12](#_Toc104319520)

[1.5. Тестирование 13](#_Toc104319521)

[Заключение 21](#_Toc104319522)

[Список использованных источников 23](#_Toc104319523)

[Приложение А 24](#_Toc104319524)

[Приложение Б 30](#_Toc104319525)

## Введение

В курсовом проекте требуется разработать программные средства (ПС) создания и визуализации таких геометрических фигур как окружность, треугольник и сложная фигура (равнобедренный треугольник, окружность, вписанная в равнобедренный треугольник).

Разработка осуществляется на языке С++ на платформе Microsoft Visual Studio 22. Для моделирования программных средств используется язык UML.

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

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

# 1 РАЗРАБОТКА ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРИЛОЖЕНИЯ. ОКРУЖНОСТЬ В РАВНОБЕДРЕННОМ ТРЕУГОЛЬНИКЕ.

## 1.1 Анализ предметной области и функциональных требований

Анализ предметной области разработки включает рисунок, указанный в варианте технического задания на курсовой проект (рисунок 1) и соответствующую ему модель предметной области.

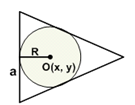


Рисунок 1 – Заданная геометрическая фигура

Модель предметной области для приведенной выше фигуры приведена на рисунке 2.

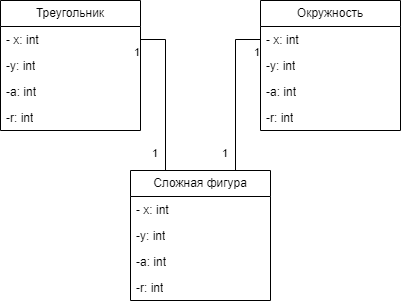


Рисунок 2 – Модель предметной области

Для предметной области, приведенной на рисунке 1, сложная фигура представляет собой окружность в равнобедренном треугольнике, с центром в точке О с координатами (X, Y), радиусом(R), проведённый из центра окружности к стороне равнобедренного треугольника длиной(а).

## 1.2 Анализ функциональных требований

В техническом задании на курсовой проект определены следующие функциональные требования:

* создать геометрическую фигуру (равнобедренный треугольник, окружность, вписанная в равнобедренный треугольник);
* показать/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 фигуру» и «Скрыть фигуру» расширяют функциональность варианта «Переместить фигуру».

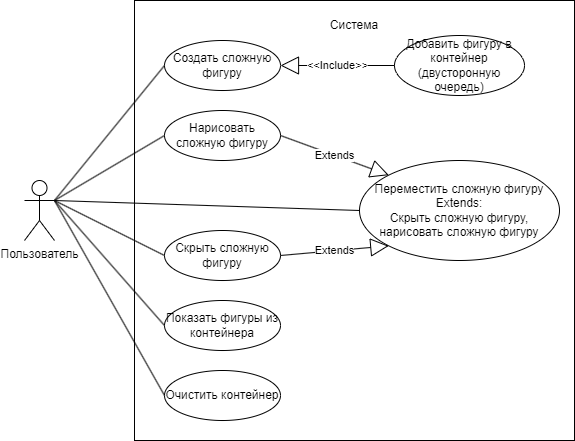


Рисунок 3 – Диаграмма вариантов использования

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

Таблица 1 – Сценарий варианта использования «Переместить фигуру»

|  |
| --- |
| **Наименование: Переместить фигуру** |
| **ID: 3** |
| **Краткое описание:** система удаляет изображение фигуры и рисует его в новых координатах |
| **Действующие лица:** пользователь |
| **Предусловие:** фигура создана и нарисована |
| **Основной поток:**   1. Пользователь задает новые координаты X, Y фигуры   Точка расширения «Скрыть»  2 Система переопределяет координаты фигуры  Точка расширения «Нарисовать» |
| **Постусловие:** фигура перемещена |

Таблица 2 – Сценарий варианта использования «Скрыть фигуру»

|  |
| --- |
| **Наименование: Скрыть фигуру** |
| **ID: 2** |
| **Краткое описание:** система удаляет изображение фигуры |
| **Действующие лица:** пользователь |
| **Предусловие:** фигура создана и нарисована |
| **Основной поток:**   1. Пользователь инициирует удаление изображения фигуры 2. Система рисует изображение фигуры цветом фона |
| **Постусловие:** изображение фигуры скрыто |

Таблица 3 – Сценарий варианта использования «Нарисовать фигуру»

|  |
| --- |
| **Наименование: Нарисовать фигуру** |
| **ID: 1** |
| **Краткое описание:** система рисует изображение фигуру с центром в точке О координатами (X, Y), радиусом(R) и заданной стороной (a) |
| **Действующие лица:** пользователь |
| **Предусловие:** фигура создана |
| **Основной поток:**   1. Пользователь инициирует рисование с центром в точке О координатами (X, Y), радиусом(R) и стороной (a) 2. Система рисует изображение фигуры с заданными координатами точки О (X, Y), заданным радиусом (R) и заданной стороной (a) |
| **Постусловие:** фигура нарисована |

## 1.3. Проектирование

В основе моделирования структуры ПС лежит модель предметной области («Окружность, вписанная в равнобедренный треугольник») и модель хранения информации о создаваемых фигурах (стек как динамический массив). В общем случае структуру ПС можно отобразить как следующую диаграмму классов (рисунок 4).

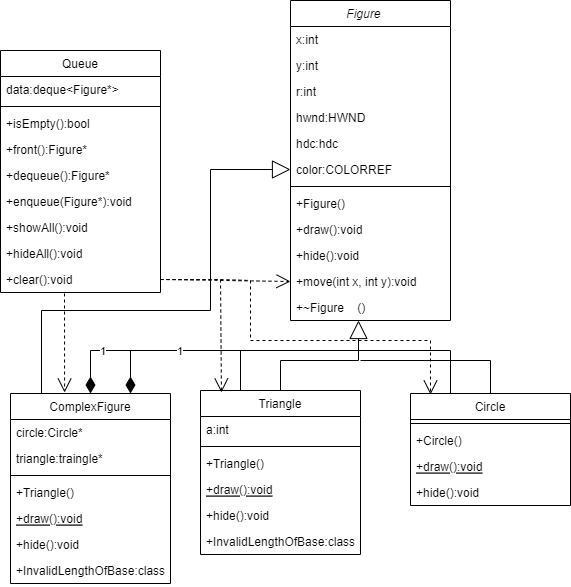


Рисунок 4 – Диаграмма классов для предметной области

«Окружность, вписанная в равнобедренный треугольник»

**Спецификация классов**

**Название** класса: **Figure**

**Назначение:** абстрактный класс

**Члены класса:**

* x: int – координата центра;
* y: int – координата центра;
* r: int – радиус окружности;
* hwnd: HWND – идентификатор окна;
* hdc: HDC – контекст изображения;
* color: COLORREF – цвет фигуры.

**Функции класса:**

* Figure() –конструктор класса;
* draw() – показать фигуру;
* hide() – скрыть фигуру;
* move(int, int) – переместить фигуру.

**Название** класса: **Circle**

**Назначение:** класс сущности фигуры окружность, наследник класса Figure.

**Члены класса:**

**Функции класса:**

* Circle() – конструктор класса;
* draw() – показать фигуру;
* hide() – скрыть фигуру;

**Название** класса: **Triangle**

**Назначение:** класс сущности фигуры треугольник, наследник класса Figure.

**Члены класса:**

* a:int — длина основания треугольника

**Функции класса:**

* Triangle() – конструктор класса;
* draw() – показать фигуру;
* hide() – скрыть фигуру.

**Название** класса: **ComplexFigure**

**Назначение:** класс сущности фигуры сложная фигура, наследник класса Figure.

**Члены класса:**

* Circle: circle\* – указатель на объект класса circle;
* Triangle: triangle\* - указатель на объект класса triangle.

**Функции класса:**

* ComplexFigure() – конструктор класса;
* draw() – показать фигуру;
* hide() – скрыть фигуру;
* move(int, int) – переместить фигуру.

**Название класса:** Queue

**Назначение:**  класс хранения фигур

**Члены класса:**

* deque<Figure\*> data – двусторонняя очередь, как очередь, хранящая фигуры.

**Функции класса:**

* bool isEmpty() - проверка, пуста ли очередь;
* void enqueue(Figure\*)- добавление элемента в конец очереди;
* Figure\* dequeue()-удаление и возврат элемента из начала очереди;
* Figure\* front() - получение элемента из начала очереди без его удаления;
* void showAll() - Отрисовка всех фигур;
* void hideAll() - Функция прячет все фигуры;
* void clear() – очистка очереди;

## 1.4. Реализация

Код программы для предметной области «Окружность, вписанная в равнобедренный треугольник» с графическим выводом фигур в окне консольного приложения приведен в «Приложение А», диаграмма компонентов – на рисунке 5. Описание компонентов приведено в таблице 4.

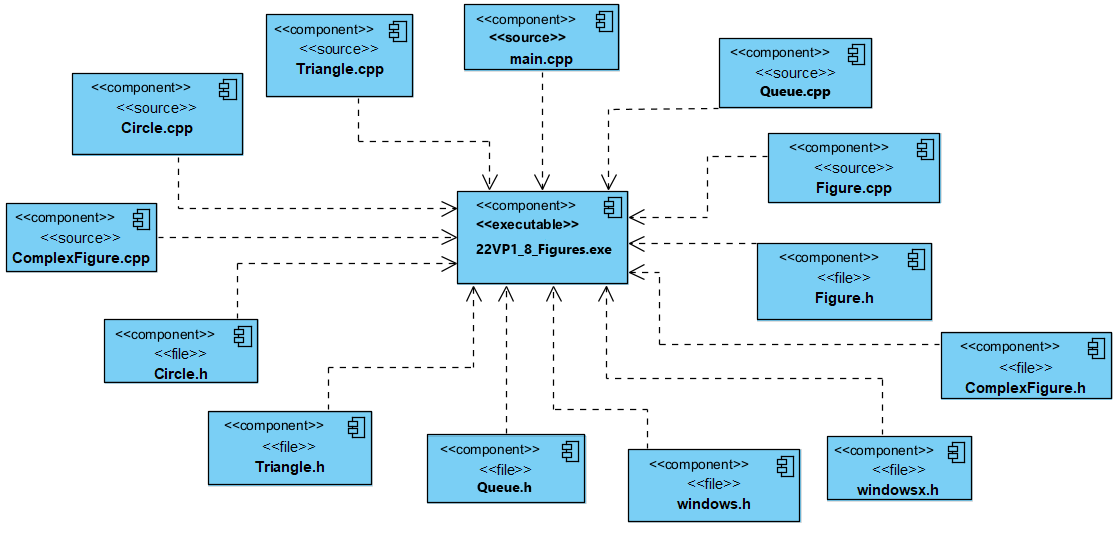


Рисунок 5 - Диаграмма компонентов для предметной области «Окружность, вписанная в равнобедренный треугольник»

Таблица 4 – Компоненты для предметной области «Окружность и треугольник»

|  |  |
| --- | --- |
| **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 |
| Triangle.h | Заголовочный файл класса Triangle |
| Triangle.cpp | Исходный файл класса Triangle |
| ComplexFigure.h | Заголовочный файл класса ComplexFigure |
| ComplexFigure.cpp | Исходный файл класса ComplexFigure |
| Queue.h | Заголовочный файл класса Stack |
| Queue.cpp | Исходный файл класса Stack |
| main.cpp | Главная программа |
| windows.h | Системный заголовочный файл |
| windowsx.h | Системный заголовочный файл |
| 22VP1\_8\_Figures.exe | Исполняемый файл приложения |

## 1.5. Тестирование

В рамках курсового проекта основной методологией тестирования является функциональное тестирование.

План функционального тестирования формируется на основе вариантов использования. План включает варианты использования с описанием тестов, соответствующих классам эквивалентности для этого варианта, и результатам выполнения тестов (таблица 5).

Таблица 5 − План тестирования

|  |  |  |
| --- | --- | --- |
| **Вариант использования** | **Тест** | **Результат** |
| **Создать окружность** | **Тест1**  Координаты центра: 100, 200  Радиус: 50 | Тест пройден.  Объект создан |
| **Тест2**  Координаты центра: 300, 300  Радиус: 110 | Тест пройден.  Объект создан |
| **Тест 3**  Координаты центра: 100, 200  Радиус: 100 | Тест пройден.  Объект создан |
| **Тест 4**  Координаты центра: 400, 400  Радиус 100 | Тест пройден  Объект создан |
| **Тест 5**  Координаты центра: 400, -20  Радиус 100 | Тест пройден  Объект создан |
| **Показать окружность** | **Тест1**  Координаты центра: 100, 200  Радиус: 50 | Тест пройден.  Окружность нарисована правильно |
| **Тест2**  Координаты центра: 250, 400  Радиус: 70 | Тест пройден.  Окружность нарисована правильно |
| **Тест 3**  Координаты центра: 150, 300  Радиус: 100 | Тест пройден.  Окружность нарисована правильно |
| **Тест 4**  Координаты центра: 400, 400  Радиус 100 | Тест пройден  Окружность не отрисована, обработано исключение (Рисунок 7) |
| **Тест 5**  Координаты центра: 400, -20  Радиус 100 | Тест пройден  Окружность не отрисована, обработано исключение (Рисунок 7) |
| **Скрыть**  **окружность** | **Тест1**  Координаты центра: 100, 200  Радиус: 50 | Тест пройден  Окружность скрыта |
| **Тест2**  Координаты центра: 300, 300  Радиус: 110 | Тест пройден  Окружность скрыта |
| **Тест 3**  Координаты центра: 100, 200  Радиус: 90 | Тест пройден  Окружность скрыта |
| **Переместить**  **окружность** | **Тест1**  Новые координаты центра: 300, 200 | Тест пройден.  Окружность перемещена в новые координаты центра |
| **Тест2**  Новые координаты центра: 300, 350 | Тест пройден.  Окружность перемещена в новые координаты центра |
| **Тест3**  Новые координаты центра: 250, 150 | Тест пройден.  Окружность перемещена в новые координаты центра |
| **Тест4**  Новые координаты центра: 700, 750 | Тест пройден.  Окружность не перемещена, обработано исключение (Рисунок 7) |
| **Тест5**  Новые координаты центра: 250, -50 | Тест пройден.  Окружность не перемещена, обработано исключение (Рисунок 7) |

Продолжение таблицы 5

|  |  |  |
| --- | --- | --- |
| **Создать**  **треугольник** | **Тест1**  Координаты центра: 250, 250  Радиус: 25  Основание: 60 | Тест пройден.  Объект создан |
| **Тест2**  Координаты центра: 300, 300  Радиус: 40  Основание: 90 | Тест пройден.  Объект создан |
| **Тест 3**  Координаты центра: 200, 250  Радиус: 70  Основание: 90 | Тест пройден.  Объект не создан.  Вызвано исключение  (рисунок 6) |
| **Тест 4**  Координаты центра: 400, 400  Радиус: 25  Основание: 60 | Тест пройден  Объект создан |
| **Тест 5**  Координаты центра: 400, -20  Радиус: 55  Основание: 75 | Тест пройден.  Объект не создан.  Вызвано исключение  (рисунок 6) |
| **Показать**  **треугольник** | **Тест 1**  Координаты центра: 250, 250  Радиус: 25  Основание: 60 | Тест пройден.  Треугольник нарисован правильно |
| **Тест2**  Координаты центра: 300, 300  Радиус: 40  Основание: 90 | Тест пройден.  Треугольник нарисован правильно |
| **Тест 3**  Координаты центра: 200, 250  Радиус: 40  Основание: 90 | Тест пройден.  Треугольник нарисован правильно |
| **Тест 4**  Координаты центра: 400, 400  Радиус: 25  Основание: 60 | Тест пройден.  Треугольник не отрисован, обработано исключение (Рисунок 7) |
| **Тест 5**  Координаты центра: 400, -20  Радиус: 30  Основание: 75 | Тест пройден.  Треугольник не отрисован, обработано исключение (Рисунок 7) |
| **Скрыть**  **треугольник** | **Тест 1**  Координаты центра: 250, 250  Радиус: 25  Основание: 60 | Тест пройден.  Треугольник скрыт |
| **Тест2**  Координаты центра: 300, 300  Радиус: 40  Основание: 90 | Тест пройден.  Треугольник скрыт |
| **Тест 3**  Координаты центра: 250, 250  Радиус: 50  Основание: 60 | Тест пройден.  Треугольник скрыт |
| **Тест 4**  Координаты центра: 250, 100  Радиус: 95  Основание: 200 | Тест пройден.  Треугольник скрыт |
| **Тест 5**  Координаты центра: 115, 115  Радиус: 30  Основание: 60 | Тест пройден.  Треугольник скрыт |
| **Переместить**  **треугольник** | **Тест1**  Новые координаты центра: 300, 300 | Тест пройден.  Треугольник перемещен в новые координаты |
| **Тест2**  Новые координаты центра: 400, 400 | Тест пройден.  Треугольник перемещен в новые координаты |
| **Тест3**  Новые координаты центра: 350, 200 | Тест пройден.  Треугольник перемещен в новые координаты |
| **Тест4**  Новые координаты центра: 400, 400 | Тест пройден.  Треугольник не перемещен в новые координаты. Обработано исключение (Рисунок 7) |
| **Тест5**  Новые координаты центра: 350, 200 | Тест пройден.  Треугольник не перемещен в новые координаты. Обработано исключение (Рисунок 7) |

Продолжение таблицы 5

|  |  |  |
| --- | --- | --- |
| **Создать**  **сложную фигуру** | **Тест1**  Координаты центра: 200, 400  Радиус: 60  Основание: 200 | Тест пройден.  Сложная фигура нарисована правильно |
| **Тест2**  Координаты центра: 100, 100  Радиус: 60  Основание: 200 | Тест пройден.  Сложная фигура нарисована правильно  (рисунок 7) |
| **Тест 3**  Координаты центра: 300, 200  Радиус: 20  Основание: 60 | Тест пройден.  Сложная фигура нарисована правильно |
| **Показать**  **сложную фигуру** | **Тест1**  Координаты центра: 200, 400  Радиус: 60  Основание: 200 | Тест пройден.  Сложная фигура нарисована правильно |
| **Тест2**  Координаты центра: 100, 100  Радиус: 60  Основание: 200 | Тест пройден.  Сложная фигура нарисована правильно  (рисунок 7) |
| **Тест 3**  Координаты центра: 300, 200  Радиус: 20  Основание: 60 | Тест пройден.  Сложная фигура нарисована правильно |
| **Скрыть**  **сложную фигуру** | **Тест1**  Координаты центра: 200, 400  Радиус: 60  Основание: 200 | Тест пройден  Сложная фигура скрыта |
| **Тест2**  Координаты центра: 100, 100  Радиус: 60  Основание: 200 | Тест пройден  Сложная фигура скрыта  (рисунок 8) |
| **Тест 3**  Координаты центра: 300, 200  Радиус: 20  Основание: 60 | Тест пройден  Сложная фигура скрыта |
| **Переместить**  **сложную фигуру** | **Тест1**  Новые координаты центра: 400, 400 | Тест пройден.  Сложная фигура перемещена в новые координаты центра  (рисунок 9) |
| **Тест2**  Новые координаты центра: 600, 300 | Тест пройден.  Сложная фигура перемещена в новые координаты центра |
| **Тест3**  Новые координаты центра: 100, 300 | Тест пройден.  Сложная фигура перемещена в новые координаты центра |
| **Тест4**  Новые координаты центра: 100, 600 | Тест пройден.  Сложная фигура не перемещена в новые координаты центра, обработано исключение (Рисунок 7) |
| **Тест5**  Новые координаты центра: 400, 400 | Тест пройден.  Сложная фигура не перемещена в новые координаты центра, обработано исключение (Рисунок 7) |
| **Добавить**  **сложную фигуру**  **в контейнер** | **Тест1**  Добавление окружности с координатами центра: 200, 400  Радиус: 60  Основание: 200 | Тест пройден.  Сложная фигура добавлена в cтек. |
| **Добавить**  **треугольник в контейнер** | **Тест2**  Добавление сложной фигуры с координатами центра:250, 250  Радиус: 25  Основание: 60 | Тест пройден.  Треугольник добавлен  в cтек. |
| **Добавить окружность в**  **контейнер** | **Тест 3**  Координаты центра: 100, 200  Радиус: 50 | Тест пройден.  Окружность добавлена  в стек |
| **Показать все**  **фигуры из контейнера** | **Тест 1**  **Окружность:**  Координаты центра:100,200  Радиус: 50  **Треугольник:**  Координаты центра: 250, 250  Радиус: 25  Основание: 60  **Сложная фигура:**  Координаты центра: 200, 350  Радиус: 60  Основание: 200 | Тест пройден.  Все фигуры были нарисованы правильно.  (рисунок 10) |
| **Тест 2**  **Окружность:**  Координаты центра:210,210  Радиус: 50  **Треугольник:**  Координаты центра: 150, 250  Радиус: 25  Основание: 60  **Сложная фигура:**  Координаты центра: 170, 250  Радиус: 60  Основание: 200 | Тест пройден.  Все фигуры были нарисованы правильно.  (рисунок 10) |
| **Тест 3**  **Окружность:**  Координаты центра:100,260  Радиус: 50  **Треугольник:**  Координаты центра: 250, 250  Радиус: 25  Основание: 60  **Сложная фигура:**  Координаты центра: 230, 400  Радиус: 60  Основание: 200 | Тест пройден.  Треугольник и Треугольник отрисованы, сложная фигура не была отрисована, обработано исключение  (рисунок 7) |

В таблице приведен не полный план тестирований. Но данного плана вполне хватает так, как мы проверяем один вариант из множества схожих, которые несущественно отличаются по реализации.

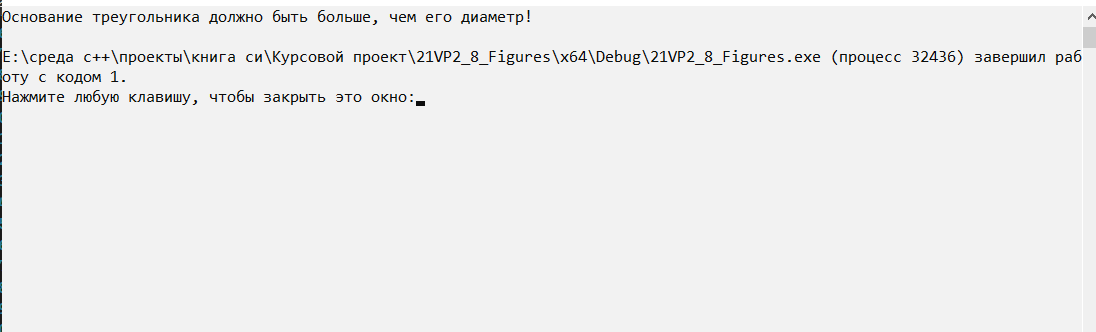


Рисунок 6 – Неправильно заданное значение основания треугольника

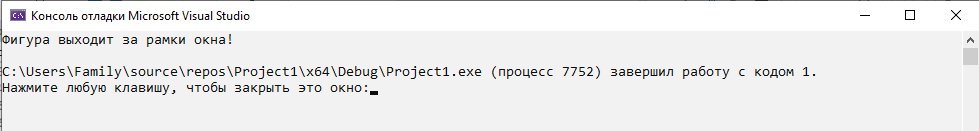


Рисунок 7 – Неправильно заданные значения координаты фигуры

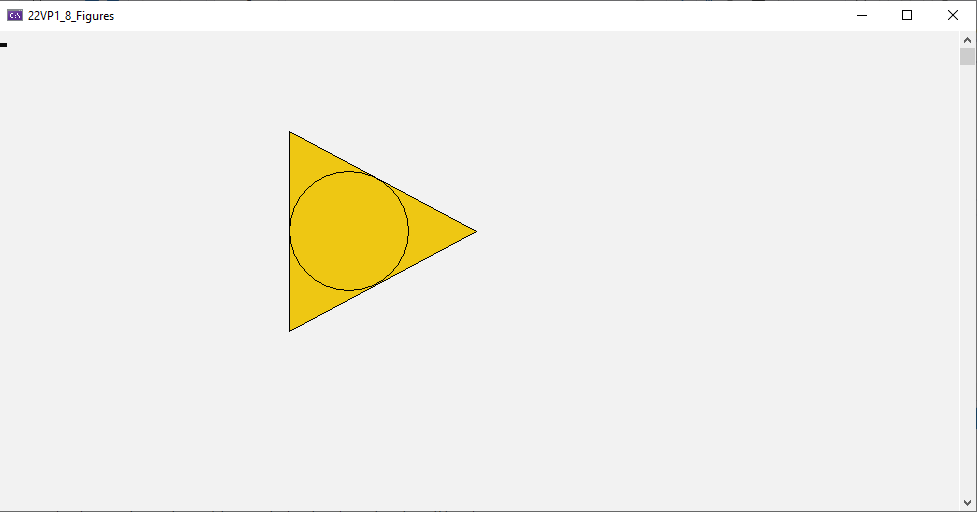


Рисунок 8 – Показ сложной фигуры

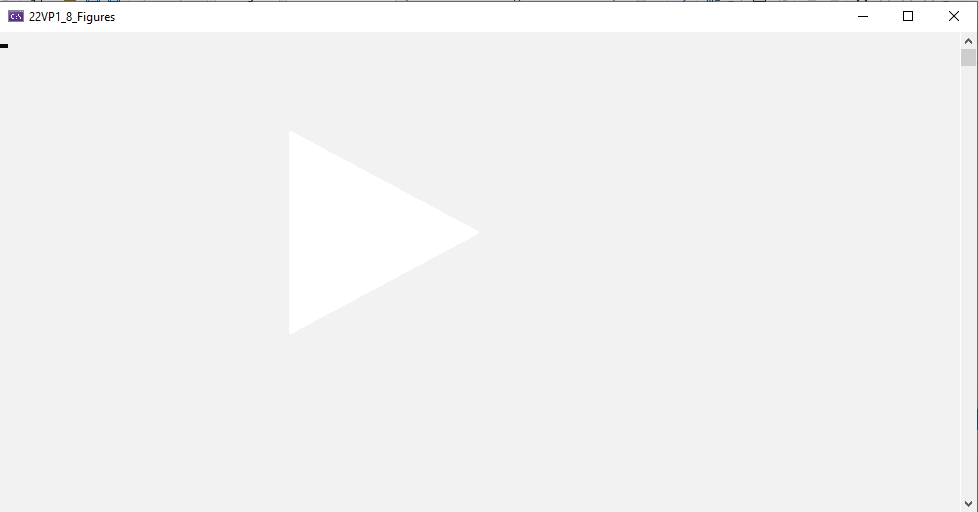


Рисунок 9 – Скрытие сложной фигуры

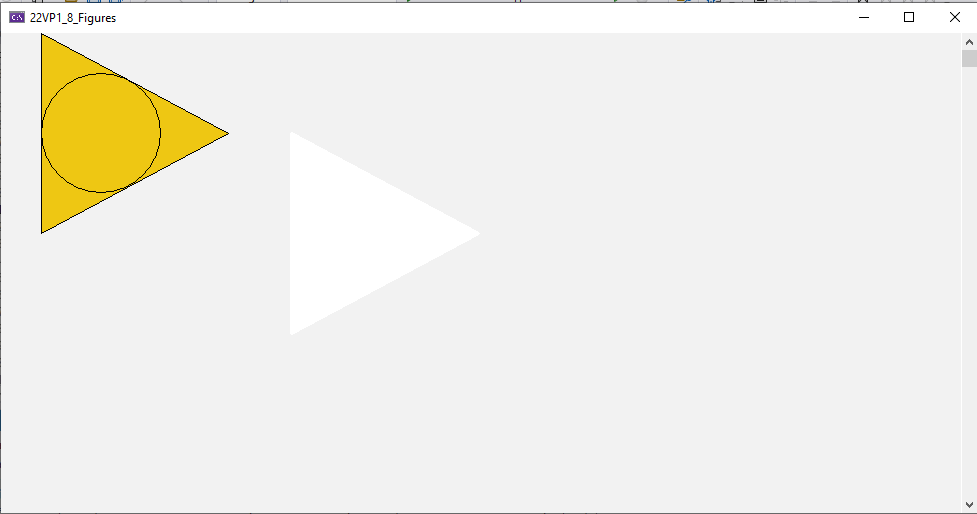


Рисунок 10 – Перемещение сложной фигуры

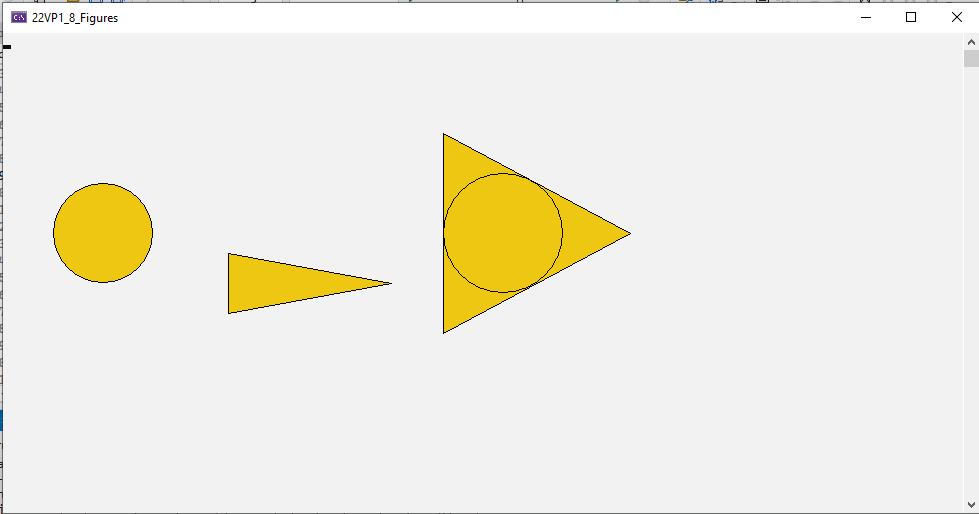


Рисунок 11 – Показ всех фигур входящих в стек

При введении пользователем некорректных значений параметров геометрических фигур, которые приводят к неправильному графическому отображению в консоли – в коде приложения присутствует класс исключений и операторы вывода с описанием исключительной ситуации. Так, на рисунке 6 приложение вышло на исключение при попытке пользователя создать сложную фигуру с координатами центра (200; 250), радиуса (70), основания (90).

# Заключение

В процессе анализа предметной области курсового проекта была построена модель будущего приложения, которая описала реализуемые геометрические фигуры (ComplexFigure – сложная фигура, Circle - окружность, Triangle - треугольник). Сущности ComplexFigure, Circle и Triangle принимают в качестве атрибутов координаты центра фигур (x, y), радиус окружности (r) и основание треугольника (а). Сущность ComplexFigure состоит из сущностей Circle и Triangle и принимает каждую из них только один раз.

С помощью анализа функциональных требований были выявлены все функции вышеперечисленных сущностей (создать фигуру, показать/скрыть фигуру, переместить фигуру, добавить фигуру в стек, выбрать фигуру из стека, удалить стек, стек пуст, стек заполнен). Благодаря построенной диаграмме вариантов использования и описанным сценариям некоторых вариантов был осуществлен следующий этап – проектирование.

В процессе выполнения курсового проекта были спроектированы и реализованы следующие классы: базовый класс Figure и наследуемые от него классы Circle, Triangle и ComplexFigure. Все классы наследуют от класса Figure координаты центра (x, y), радиус (r) и функции показа(draw), скрытия (hide) и перемещения (move) фигур. Все эти функции переопределяются в соответствующих классах (Circle, Triangle, ComplexFigure). Класс ComplexFigure обрабатывает два объекта классов Circle и Triangle и представляет собой фигуру, состоящую из этих объектов.

С помощью тестирования были выявлены и исправлены ошибки, рассмотрены многочисленные варианты расположения фигур. Также была добавлена обработка исключительной операции неправильно заданных значений, что позволяет избежать создания фигур по некорректно введенным данным и вывести соответствующее сообщение для пользователя.

В результате разработано приложение, позволяющее создавать геометрические фигуры (равнобедренный треугольник, окружность, вписанная в равнобедренный треугольник), показывать и скрывать их, перемещать на новые координаты, сохранять их в стек и удалять из стека.

# 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. Т. А. Павловская. C/C++. Программирование на языке высокого уровня. – СПб.: Питер, 2003. – 461 с
3. Мартин Р. Чистый код: создание, анализ и рефакторинг. - 3-е изд. - Санкт-Петербург: Издательство "Питер", 2022. - 464 с.
4. Джим Арлоу. UML2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование/Джим Арлоу, Айла Нейштадт. – Санкт-Петербург, Издательство Символ-Плюс, 2007. – 624с
5. Гради Буч. Введение в UML от создателей языка/Гради Буч, Джеймс Рамбо, Ивар Якобсон. – Москва, Издательство ДМК-Пресс, 2015. – 496с

# 

# Приложение А

КОД ПРОГРАММЫ

(обязательное)

**Главная программа:**

#include "Triangle.h"

#include <iostream>

#include <stdio.h>

#include "ComplexFigure.h"

#include "Queue.h"

#include "InvalidDataException.h"

using namespace std;

const int not\_used = system("color F0");

int main()

{

try {

SetConsoleTitle((LPCWSTR)L"22VP1\_8\_Figures");

setlocale(LC\_ALL, "Ru");

Circle circle(100, 200, 50, RGB(238, 199, 19));

Triangle triangle(250, 250, 25, 60, RGB(238, 199, 19));

ComplexFigure figure(500, 200, 60, 200, RGB(238, 199, 19));

Queue container;

container.enqueue(&circle);

container.enqueue(&triangle);

container.enqueue(&figure);

container.showAll();

Sleep(2000);

Sleep(2000);

figure.move(100,100);

Sleep(2000);

}

catch (Triangle::InvalidLengthOfBase& ex)

{

cerr << ex.what()<<endl;

return 1;

}

catch (const InvalidDataException& ex)

{

cerr << ex.what()<<endl;

return 1;

}

}

**Figure.h**

#pragma once

#include<iostream>

#include <windows.h>

#include <windowsx.h>

class Figure

{

protected:

int x, y, r;

int w, h;

COLORREF color;

HWND hwnd;

HDC hdc;

public:

Figure(int x\_ = 0, int y\_ = 0, int r\_ = 0 , COLORREF \_color=0);

virtual void draw() = 0;

virtual void hide() = 0;

virtual void move(int, int);

~Figure();

};

**Figure.cpp**

#include "Figure.h"

Figure::Figure(int x\_, int y\_, int r\_, COLORREF \_color)

{

x = x\_;

y = y\_;

r = r\_;

color = \_color;

hwnd = GetConsoleWindow();

RECT r;

GetClientRect(hwnd, &r);

hdc = GetDC(hwnd);

w = r.right;

h = r.bottom;

}

void Figure::move(int new\_x, int new\_y)

{

hide();

x = new\_x;

y = new\_y;

draw();

}

Figure::~Figure()

{

ReleaseDC(hwnd, hdc);

}

**Circle.h**

#pragma once

#include "Figure.h"

class Circle : public Figure

{

public:

Circle(int x\_, int y\_, int r\_, COLORREF color) : Figure(x\_, y\_, r\_, color) {}

void draw() override;

void hide() override;

};

**Circle.cpp**

#include "Circle.h"

#include <iostream>

#include "InvalidDataException.h"

void Circle::draw()

{

HPEN pen;

pen = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

SelectObject(hdc, pen);

HBRUSH brush;

brush = CreateSolidBrush(color);

SelectObject(hdc, brush);

int x1 = x - r;

int y1 = y - r;

int x2 = x + r;

int y2 = y + r;

if (x1<0||y1<0||x2 > w || y2 > h) {

throw InvalidDataException("Фигура выходит за рамки окна!");

}

else

{

Ellipse(hdc, x1, y1, x2, y2);

}

}

void Circle::hide()

{

HPEN pen;

pen = CreatePen(PS\_SOLID, 3, RGB(255, 255, 255));

SelectObject(hdc, pen);

HBRUSH brush;

brush = CreateSolidBrush(RGB(255, 255, 255));

SelectObject(hdc, brush);

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

}

**Triangle.h**

#pragma once

#include <math.h>

#include "Figure.h"

class Triangle : public Figure

{

private:

int a;

public:

Triangle(int x, int y, int r, int a\_, COLORREF color) :Figure(x, y, r, color), a(a\_)

{

if (a < 2 \* r) throw InvalidLengthOfBase("Основание треугольника должно быть больше его диаметра");

}

void draw() override;

void hide() override;

class InvalidLengthOfBase : public std::exception {

private:

std::string message;

public:

InvalidLengthOfBase(const std::string& msg) : message(msg) {}

const char\* what() const noexcept override {

return message.c\_str();

}

};

};

**Triangle.cpp**

#include "Triangle.h"

#include <iostream>

#include "InvalidDataException.h"

using namespace std;

void Triangle::draw()

{

int l = (4 \* r \* r \* r + a \* a \* r) / (a \* a - 4 \* r \* r);

int x1 = x - r;

int y1 = y - a / 2;

int x2 = x - r;

int y2 = y + a / 2;

int x3 = x + l;

int y3 = y;

POINT points[] = { {x1, y1}, {x2, y2}, {x3, y3} };

HPEN pen;

pen = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

SelectObject(hdc, pen);

HBRUSH brush;

brush = CreateSolidBrush(color);

SelectObject(hdc, brush);

if (x1<0 || y1<0 || x2 <0 || y2 > h ||x3>w) {

throw InvalidDataException("Фигура выходит за рамки окна!");

}

else

{

Polygon(hdc, points, 3);

}

}

void Triangle::hide()

{

int l = (4 \* r \* r \* r + a \* a \* r) / (a \* a - 4 \* r \* r);

POINT points[] = { {x - r, y - a / 2}, {x - r, y + a / 2}, {x + l, y} };

HPEN pen;

pen = CreatePen(PS\_SOLID, 3, RGB(255, 255, 255));

SelectObject(hdc, pen);

HBRUSH brush;

brush = CreateSolidBrush(RGB(255, 255, 255));

SelectObject(hdc, brush);

Polygon(hdc, points, 3);

}

**ComplexFigure.h**

#pragma once

#include "Figure.h"

#include "Triangle.h"

#include "Circle.h"

class ComplexFigure : public Figure

{

private:

Circle\* circle;

Triangle\* triangle;

public:

ComplexFigure(int x, int y, int r, int a, COLORREF color)

:Figure(x, y, r,color), circle(new Circle(x, y, r, color)),

triangle(new Triangle(x, y, r, a, color)) {}

void draw() override;

void hide() override;

void move(int, int) override;

~ComplexFigure();

};

**ComplexFigure.cpp**

#include "ComplexFigure.h"

#include "InvalidDataException.h"

void ComplexFigure::draw()

{

triangle->draw();

circle->draw();

}

void ComplexFigure::hide()

{

circle->hide();

triangle->hide();

}

ComplexFigure::~ComplexFigure()

{

delete circle;

delete triangle;

}

void ComplexFigure::move(int x, int y)

{

triangle->move(x, y);

circle->move(x, y);

}

**Queue.h**

#pragma once

#include <deque>

#include "Figure.h"

class Queue {

private:

std::deque<Figure\*> data; // двусторонняя очередь

public:

// Проверка, пуста ли очередь

bool isEmpty() const;

// Добавление элемента в конец очереди

void enqueue(Figure\*);

// Удаление и возврат элемента из начала очереди

Figure\* dequeue();

// Получение элемента из начала очереди без его удаления

Figure\* front() const;

//Отрисовка всех фигур

void showAll() const;

//Функция прячет все фигуры

void hideAll() const;

void clear();

};

**Queue.cpp**

#include <deque>

#include "Figure.h"

#include "Queue.h"

bool Queue::isEmpty() const {

return data.empty();

}

void Queue::enqueue(Figure\* element) {

data.push\_back(element);

}

Figure\* Queue::dequeue() {

Figure\* frontElement = data.front();

data.pop\_front();

return frontElement;

}

Figure\* Queue::front() const {

return data.front();

}

void Queue::showAll() const{

for (auto figure : data) {

figure->draw();

}

}

void Queue::hideAll() const {

for (auto figure : data) {

figure->hide();

}

}

void Queue::clear() {

while (!isEmpty()) {

data.pop\_front();

}

# }

# Приложение Б

ГРАФИЧЕСКАЯ ЧАСТЬ

(обязательное)

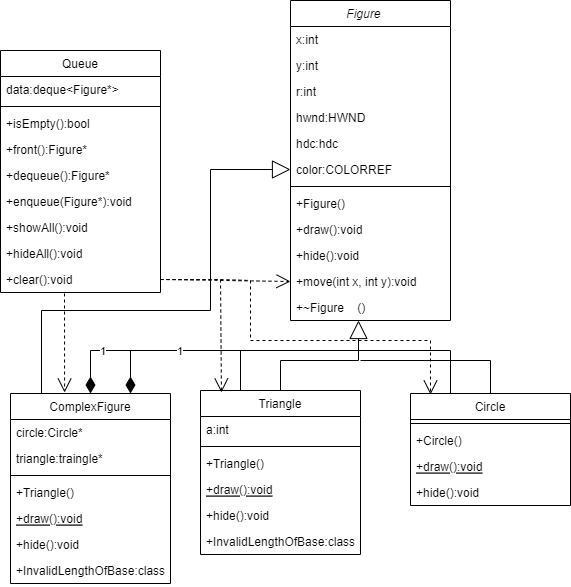


Рисунок Б1 – Диаграмма классов для предметной области

«Окружность, вписанная в равнобедренный треугольник»

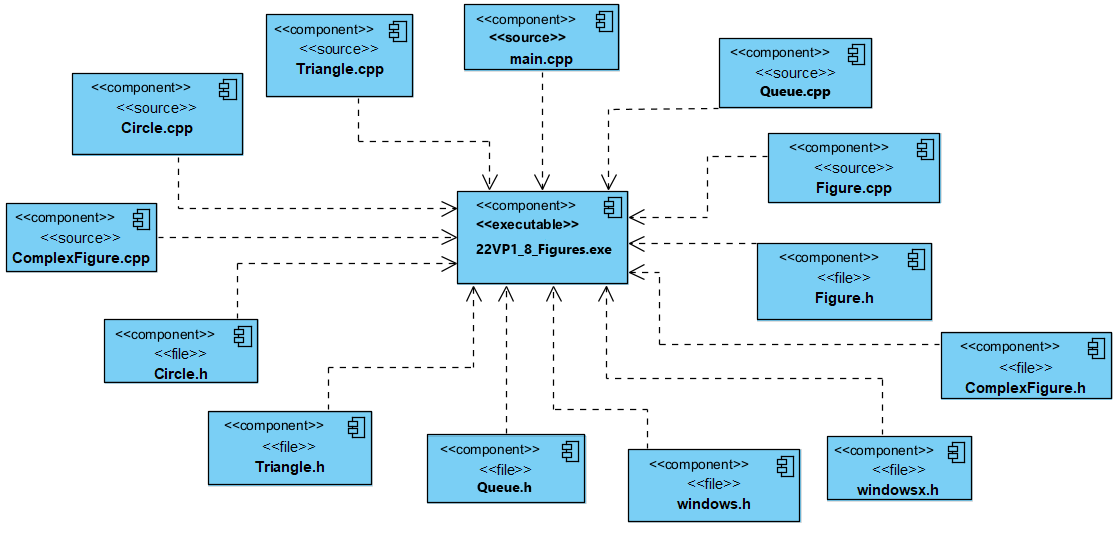


Рисунок Б2 - Диаграмма компонентов для предметной области «Окружность, вписанная в равнобедренный треугольник»