**High-Level Design**

1

10

2

Parser

Visualizer

4

3

Core

Logger

67

9

Solver

Tests

8

7

5

Console

Generator

**Pipeline**

**ПО консоль**

1. User из командной строки запускает ПО Console с параметрами запуска: path1.txt path2.txt op path3.txt [path4.txt]

path1 – путь к первому множеству

path2 – путь ко второму множеству

op – тип операции (0-объединение, 1 – пересечение, 2 – разность)

path3 – выходной файл

path4 – путь к файлу с эталоном (необязательный параметр)

2. ПО Console создает объекты Polygons (Core)(9) для двух входных множеств и одного выходного.

3. ПО Console проверяет наличие и корректность path1, протоколирует выполнение (Logger), завершает работу в случае ошибки.

4. ПО Console вызывает функцию чтения множества (Parser)(7), передав название файла и объект Polygons.

4. Parser из файла извлекает в текстовом виде (1): количество многоугольников, количество точек многоугольника, координаты вершин многоугольника в порядке обхода по часовой стрелке, и записывает их в объект Polygons (Core)(3).

5. Parser вызывает функцию проверки корректности считанной структуры, протоколирует выполнение (Logger), в случае ошибки прерывает работу программы.

6. ПО Console проверяет наличие и корректность path2, протоколирует выполнение (Logger), завершает работу в случае ошибки.

7. ПО Console вызывает функцию чтения множества (Parser)(7), передав название файла и объект Polygons.

8. Parser из файла извлекает в текстовом виде (1): количество многоугольников, количество точек многоугольника, координаты вершин многоугольника в порядке обхода по часовой стрелке, и записывает их в объект Polygons (Core)(3).

9. Parser вызывает функцию проверки корректности считанной структуры, протоколирует выполнение (Logger), в случае ошибки прерывает работу программы.

10. ПО Console проверяет наличие и корректность op, протоколирует выполнение (Logger), завершает работу в случае ошибки.

11. ПО Console вызывает функцию применения выбранной операции (Solver)(8) к полученным множествам, передав 3 объекта Polygons

12. Solver применяет операцию и записывает результат в объект Polygons (Core)(6).

13. ПО Console проверяет наличие и корректность path4, в случае наличия пункты 13.1-13.4

13.1. ПО Console вызывает функцию чтения множества (Parser)(7), передав название файла и объект Polygons.

13.2. Parser из файла извлекает в текстовом виде (1): количество многоугольников, количество точек многоугольника, координаты вершин многоугольника в порядке обхода по часовой стрелке, и записывает их в объект Polygons (Core)(3).

13.3. Parser вызывает функцию проверки корректности считанной структуры (Core), протоколирует выполнение (Logger), в случае ошибки прерывает работу программы.

13.4 ПО консоль проверяет равенство результата и эталона, протоколирует выполнение (Logger), в случае ошибки завершает работу.

14. По Console вызывает функцию записи множества (Parser), передав путь к файлу и объект Polygons

15. Parser записывает в файл множество в текстовом виде (2), извлекая данные из класса (Core)(3): количество многоугольников, количество точек многоугольника, координаты вершин многоугольника в порядке обхода по часовой стрелке

**ПО Generator**

1. User из командной строки запускает ПО Generator с параметрами запуска: path.txt

2. ПО Generator создает и записывает в файл множество многоугольников в текстовом виде (5): количество многоугольников, количество точек многоугольника, координаты вершин многоугольника в порядке обхода по часовой стрелке

**ПО Visualizer**

1. User из командной строки запускает ПО Generator с параметрами запуска: path.txt

2. ПО Visualizer создает объект Polygons (Core)(4)

3. ПО Visualizer проверяет наличие и корректность path, протоколирует выполнение (Logger), завершает работу в случае ошибки.

4. ПО Visualizer вызывает функцию чтения множества (Parser)(10), передав название файла и объект Polygons.

5. Parser из файла извлекает в текстовом виде (1): количество многоугольников, количество точек многоугольника, координаты вершин многоугольника в порядке обхода по часовой стрелке, и записывает их в объект Polygons (Core)(3).

6. Parser вызывает функцию проверки корректности считанной структуры, протоколирует выполнение (Logger), в случае ошибки прерывает работу программы.

7. ПО Visualizer отображает множество в окне

Все компоненты реализуются на C++

модуль Tests использует framework Google test

модуль Visualizer использует API GDI

**Требования к компонентам**

Parser – считывание теста из файла, запись множества в файл

Core – хранение множеств многоугольников, функции изменения множеств, функции получения доступа к информации о множестве, функции проверки корректности

Solver – создание множества многоугольников, являющегося результатом булевой операции над двумя другими

Console – выполнение тестов

Visualizer – визуализация множества многоугольников

Generator – создание тестов

Tests – хранение тестов.

Logger – вывод сообщений о зарегистрированных ошибках

**Основные API**

Parser :

void read\_input(string &path, Polygons &res) – считывает множество из файла

void write\_output(string &path, Polygons &data) – записывает множество в файл

Solver :

void merge(Polygons &data1,Polygons &data2, Polygons &res) – возвращает результат объединения двух множеств

void intersect(Polygons &data1,Polygons &data2, Polygons &res) – возвращает результат пересечения двух множеств

void subtract(Polygons &data1,Polygons &data2, Polygons &res) – возвращает результат вычитания двух множеств

**Передаваемые данные**

1 – тест (.txt)

void read\_input(string &path, Polygons &res)

2 – Множество многоугольников (.txt)

void write\_output(string &path, Polygons &data)

3 – Функции изменения множества многоугольников, функции получения доступа к информации о множестве многоугольников.

4 – функции получения доступа к информации о множестве многоугольников.

5 – тест (.txt)

6 – Функции изменения множества многоугольников, функции получения доступа к информации о множестве многоугольников.

7 – вызов функции чтения теста и функции вывода множества

void read\_input(string &path, Polygons &res)

void write\_output(string &path, Polygons &data)

8 – функция применения булевой операции

void merge(Polygons &data1,Polygons &data2, Polygons &res)

void intersect(Polygons &data1,Polygons &data2, Polygons &res)

void subtract(Polygons &data1,Polygons &data2, Polygons &res)

Функции получения доступа к информации о множестве многоугольников:

int Polygons::size()const – возвращает количество многоугольников в множестве

const Polygon& Polygons::get(int i) const – возвращает i-ый многоугольник

int Polygon::size()const – возвращает количество точек в многоугольнике

const pair<int,int> & Polygon::get(int i)const – возвращает i-ую точку многоугольника

Функции изменения множества многоугольников:

void Polygon::add(pair<int,int> &p) - добавляет точку в многоугольник

void Polygons::add(Polygon &p) – добавляет многоугольник в множество