СОГЛАСОВАНО	УТВЕРЖДАЮ
Сторона ЗАКАЗЧИКА	Сторона ИСПОЛНИТЕЛЯ
Инженер-программист НИО 9740 филиала РФЯЦ-ВНИИЭФ «НИИИС им. Ю.Е. Седакова»	Профессор кафедры ИАНИ ННГУ, д.т.н.
_____ Ю.А. Живчикова «____»_____2021 г.	_____ Н.В. Старостин «____»_____2021 г.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ВЫПОЛНЕНИЯ ЛОГИЧЕСКИХ ОПЕРАЦИЙ НАД МНОЖЕСТВАМИ ОРТОГОНАЛЬНЫХ МНОГОУГОЛЬНИКОВ

Руководство программиста

Этап 2. Разработка программной документации

НИР «Разработка и реализация программного обеспечения для выполнения логических операций над множествами ортогональных многоугольников»

(Шифр ПО «SoR»)

Ответственный исполнитель

_____ В.А. Куликов

«____»_____2021 г.

2021 г.

АННОТАЦИЯ

Руководство программиста представляет собой информацию по содержанию, правилам работы и настройки программы «SoR». Программа выполняет булевы операции над множествами ортогональных многоугольников.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

1.1 Назначения программы

Программа выполняет булевы операции над множествами ортогональных многоугольников (ПО «SoR»).

1.2 Функции программы

ПО «SoR» выполняет следующие функции:

- чтение исходных данных из файла .txt;
- выполнение булевых операций над входными данными;
- запись результатов выполнения расчетов в файл .txt;
- логирование возникающих ошибок;
- запись логов в файл .txt;
- возможность визуализировать входные данные и результаты расчетов.

1.3 Системные требования

Для функционирования программы на компьютере он должен удовлетворять следующим требованиям: оперативная память не менее 16ГБ, доступная дисковая память не менее 1ГБ, процессор с PR-рейтингом не менее 2000, ОС: WindowsXP и выше, клавиатура, мышь.

2. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

ПО «SoR» является консольным приложением. Все возможные параметры для передачи в командную строку описаны в разделе «Обращение к программе». Основные характеристики ПО «SoR»:

1. Программа состоит из нескольких модулей:

- core (определяет структуру данных «многоугольник»);
- parser (позволяет считывать исходные данные и записывать результат работы программы в файл);
- logger (регистрирует сообщения о ходе работы программы);
- generator (генерирует необходимое количество данных для проверки работоспособности программы. Модуль не является обязательным);
- solver (решатель задачи);
- visualizer (визуализатор, позволяет наглядно посмотреть получившийся результат).

2. Одной из характеристик программы является одновременное и быстрое выполнение булевых операций над большим количеством (порядка 10^5) ортогональных многоугольников.

3. Время на выполнение любой булевой операции одинаковое. При 100000 многоугольников (не менее 800000 точек) время расчета составляет от 4 до 5 сек.

4. Вывод результатов осуществляется в текстовый файл. Модуль visualizer предоставляет возможность графически отобразить полученный результат.

5. Правильность полученных результатов гарантируется правильностью решенных тестовых базисов, в которых предусмотрены все утвержденные случаи.

6. Основные функции API

6.1. Парсинг - read_input

Сигнатура функции:

void read_input (string &path, Polygons &res)

Данная функция позволяет преобразовать данные из файла в класс.

Входные параметры:

- string path - путь к файлу с входными данными.

Возвращает: класс, представляющий входные данные.

6.2. Функция объединения множеств - merge

Сигнатура функции:

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

Осуществляет объединение множеств.

В случае возникновения ошибки, записывает в файл информацию, что программа делала в этот момент.

Входные параметры:

- Polygons data1 - первое множество;
- Polygons data2 - второе множество.

Возвращает: результирующий класс в переменную res.

6.3. Функция пересечения множеств - intersect

Сигнатура функции:

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

Осуществляет пересечение множеств.

В случае возникновения ошибки, записывает в файл информацию, что программа делала в этот момент.

Входные параметры:

- Polygons data1 - первое множество;
- Polygons data2 - второе множество.

Возвращает: результирующий класс в переменную res.

6.4. Функция вычитания множеств - subtract

Сигнатура функции:

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

Осуществляет вычитание множеств.

В случае возникновения ошибки, записывает в файл информацию, что программа делала в этот момент.

Входные параметры:

- Polygons data1 - первое множество;
- Polygons data2 - второе множество.

Возвращает: результирующий класс в переменную res.

6.5. Метод записи выходных данных - write_output

Сигнатура функции:

void write_output(string &path, Polygons &data)

Данная функция позволяет преобразовать результирующий класс в выходные данные и дальнейшую их запись в файл.

Входные параметры:

- Polygons data – результирующий класс.

Возвращает: количество многоугольников, количество точек в многоугольниках, координаты многоугольников в порядке их обхода по часовой стрелке.

3. ОБРАЩЕНИЕ К ПРОГРАММЕ

Для того чтобы запустить программу, необходимо выполнить команду консоли:

```
path/SOR/Console/bin/Release/Console.exe path1 path2 op path3
```

где path – путь к директории установки,

path1 – путь к файлу, содержащему первое множество,

path2 – путь к файлу, содержащему второе множество,

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

path3 - путь к файлу для записи результата.

Высокоуровневый дизайн программы приведен на рис. 1., а его подробное описание, не включая необязательные компоненты (generator, visualizer), в п. 3.1.

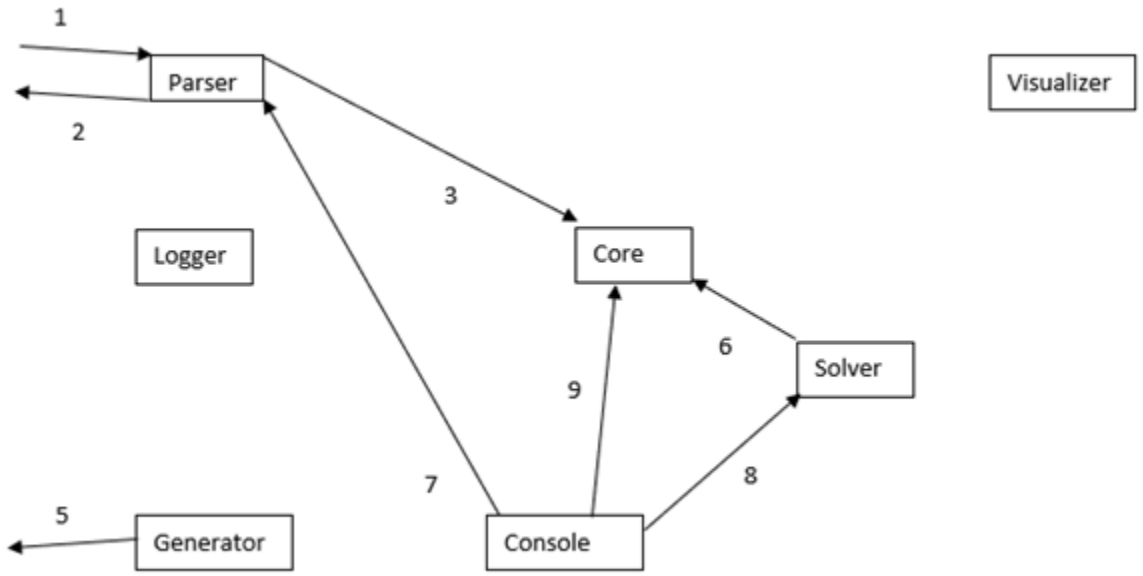

Рисунок 1 – Высокоуровневый дизайн ПО «SoR»

3.1 Описание работы программного модуля Console

Ниже приведен алгоритм работы программного модуля Console (ПМ Console):

1. Пользователь из командной строки запускает ПМ Console с параметрами запуска: `path1.txt path2.txt op path3.txt [path4.txt]`, где

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

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

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

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

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

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

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

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

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

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

12. ПМ Console вызывает функцию применения выбранной операции (ПМ Solver) (поток 8) к полученным множествам, передав 3 объекта Polygons

13. ПМ Solver применяет операцию и записывает результат в объект Polygons (ПМ Core) (поток 6).

14. ПМ Console проверяет наличие и корректность path4, в случае наличия выполняются пункты 14.1-14.4.

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

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

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

14.4 ПМ Console проверяет равенство результата и эталона, протоколирует выполнение (ПМ Logger), в случае ошибки завершает работу.

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

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

4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Входные и выходные данные содержатся в файле формата .txt. Файл имеет следующий формат:

```
Count n

Boundary m1 x1, y1, x2, y2, … xm1, ym1
```

…

```
Boundary mn x1, y1, x2, y2, … xmn, ymn
```

где n – количество многоугольников,

mi – количество вершин многоугольника в записи файла,

xi, yi – координаты вершины многоугольника.

Пример файла входных данных приведен на рис. 2. Формат выходных данных совпадает с форматом входных данных.

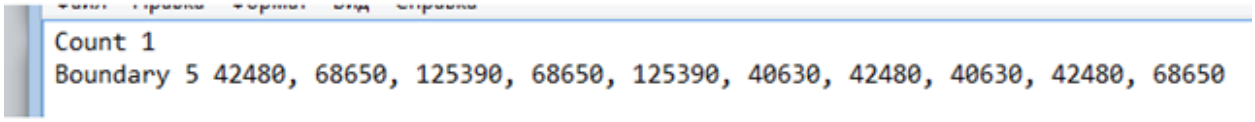

Рисунок 2 – Файл входных данных

Многоугольник должен описываться координатами его вершин в порядке обхода по часовой стрелке, последняя точка многоугольника должна совпадать с первой. Многоугольники одного множества не должны иметь пересечений ненулевой площади между собой. Многоугольники должны быть невырожденными.

К выходным данным также относится протокол работы системы, пример приведен на рис. 3.

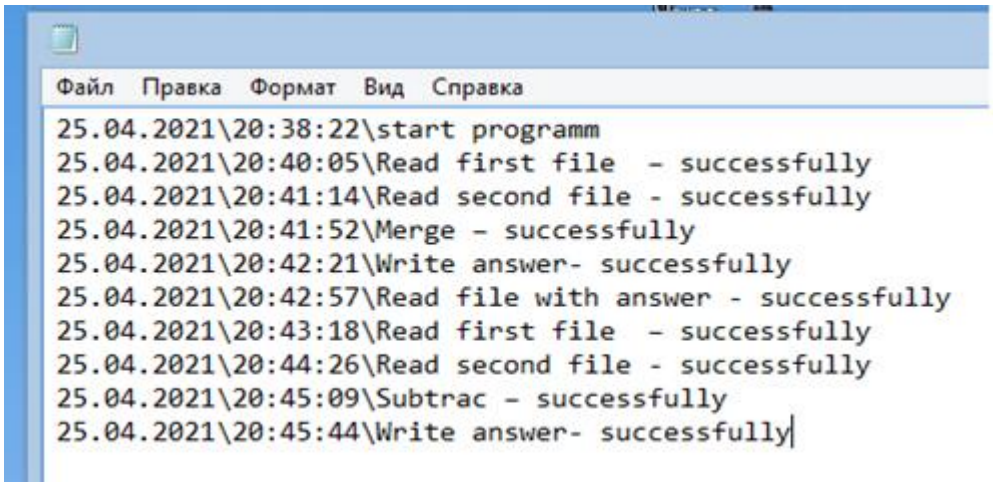

Рисунок 3 – Лог-файл

5. СООБЩЕНИЯ

При появлении любого сообщения из списка ниже, проверьте корректность входных данных.

- "Count" needed in file: path;

- The number of boundaries needed in file: path;

- "Boundary" needed in file: path;

- The number of points needed in i line in file: path;

- Wrong number of points in i line in file: path;

- Sudden EOF: path;

- Error in coordinates in i line in file: path;

- The first and last point of boundary in i line are not the same in file: path;

- Incorrect index (get);

- Incorrect index ([]).