Определение принадлежности точки невыпуклому многоугольнику	
Внутренняя спецификация	
Студент	Бебахани А. А
Преподаватель	Матюшечкин Д.С.
Сдано	

Содержание

1 Общие сведения

Название программы: PolygonInPoint.

Для функционирования программы необходима операционная система Windows 8 или выше.

Программа написана на языке C++ с использованием Visual Studio 2019.

2 Функциональное назначение

Данная программа позволяет проверить находится ли двумерная точка внутри двумерного невыпуклого многоугольника.

3 Описание логической структуры

Функции, используемые в задаче описаны в приложении А.

Используемые структуры данных описаны в приложении Б.

Дерево вызова функций представлено в приложении В.

Диаграмма потоков данных представлена в приложении Г.

Метод решения задачи:

1. Считать входные данные.
2. Проверить корректность входных данных.
3. Произвести проверку находится ли данная точка в многоугольнике методом учёта числа пересечений.
4. Вывести результат на консоль.

4 Используемые технические средства

Описание требуемых технических средств содержится в Техническом задании в пункте 3.3.

5 Вызов и загрузка

Программа запускается из командной строки. Программа принимает путь к файлу из стандартного потока ввода.

Пример запуска из командной строки:

```
PolygonInPoint.exe
input.txt
```

6 Входные и выходные данные

Формат входных и выходных данных описан в пункте 3.4 технического задания.

Приложение А

Описание функций

Функция: int main()

Обеспечивает считывание из файла, вывод в консоль ошибок, если они есть, вывод ответа в консоль, вызов главной функции, решающей задачу.

Алгоритм работы функции:

1. Считать путь к файлу и открыть его. Если открыть файл невозможно – выдать ошибку.
2. Считать данные из файла.
3. Запустить главную функцию программы.
4. Вывести ответ на задачу на консоль или ошибку, если она есть.

Функция bool is_point_inside_polygon(std::vector<point2d> polygon_points, const point2d point)

Проверяет находится ли точка внутри многоугольника, заданного набором точек.

Алгоритм работы функции:

1. Проверяем корректные ли данные. Если нет – бросаем ошибку.
2. Проверяем принадлежит ли точка грани многоугольника. Если да – возвращаем true.
3. Вычисляем углы для каждой вершины многоугольника.
4. Сортируем углы в порядке убывания их модуля.
5. Вычисляем луч для проверки принадлежности точки.
6. Считаем количество пересечений луча с многоугольником. Если количество пересечений чётное – возвращаем false, иначе – true.

Функция data_check_result check_data(const polygon2d& polygon, const point2d point)

Проверяет являются ли полученные данные корректными с точки зрения программы.

Алгоритм работы функции:

1. Если количество точек не находится в допустимом диапазоне, вернуть ошибку.
2. Если координаты искомой точки не находятся в допустимом диапазоне, вернуть ошибку.
3. Для каждой точки многоугольника проверяем находятся ли её координаты допустимом диапазоне.
4. Если найдена точка многоугольника, для которой координаты не находятся в допустимом диапазоне, вернуть ошибку.
5. Проверить есть ли совпадающие точки. Если да, вернуть ошибку.
6. Проверить, пересекаются ли стороны многоугольника. Если да, вернуть ошибку.
7. Вернуть корректность данных.

Функция bool is_point_valid(const point2d point)

Проверяет корректность координат точки.

Алгоритм работы функции:

1. Если значение одной из координат выходит за границы допустимого диапазона значений, вернуть false.
2. Иначе вернуть true.

Функция bool any_points_match(const polygon2d& polygon)

Проверяет, совпадают ли точки многоугольника.

Алгоритм работы функции:

1. Создаём множество точек.
2. Для каждой точки в многоугольнике:

2.1. Если точка уже есть в множестве, возвращаем true.

2.2. Иначе – добавляем точку в множество.

3. Возвращаем false.

Функция bool any_polygon_sides_intersect(const polygon2d& polygon)

Проверяет, пересекаются ли стороны многоугольника.

Алгоритм работы функции:

1. Для каждой пары сторон многоугольника:

1.1. Проверим что отрезки сторон не пересекаются.

1.2. Если найдена пара сторон многоугольника, которые пересекаются между собой – вернуть true.

2. Вернуть false.

Функция bool two_segments_intersect(const point2d first1, const point2d last1, const point2d first2, const point2d last2)

Проверяет, пересекаются ли два отрезка.

Алгоритм работы функции:

1. Посчитать коэффициенты прямой для двух отрезков.

2. Посчитать вспомогательные коэффициенты.

3. Провести проверки по вспомогательным коэфициентам.

4. Если одна из проверок прошла – вернуть true.

5. Иначе если все вспомогательные коэффициенты равны нулю, провести проверки по координатам точек.

6. Если проверки прошли – вернуть true.

7. Вернуть false.

Функция bool is_point_on_edge_of_polygon(const polygon2d& polygon, const point2d point)

Проверяет, находится ли точка на стороне многоугольника.

Алгоритм работы функции:

1. Для каждого ребра многоугольника:

1.1. Если данная точка принадлежит многоугольнику, вернуть true.

2. Вернуть false.

Функция bool is_point_on_segment(const point2d first, const point2d last, const point2d p)

Проверяет, находится ли точка на данном отрезке.

Алгоритм работы функции:

1. Проверить, что точка находится на одной прямой с отрезком.

2. Проверить, что значение координаты точки находятся в диапазоне значений координат точек отрезка.

3. Вернуть результат проверок.

Функция void calculate_angles_for_polygon(polygon2d& polygon, const point2d point)

Вычисляет углы между Ox и отрезками, соединяющими искомую точку и вершины многоугольника.

Алгоритм работы функции:

1. Для каждой точки в многоугольнике:

1.1. Если угол не равен нулю:

1.1.1. Вычисляем угол, «достраивая» до прямоугольного треугольника.

1.1.2. Если точка находится в 1 или 4 четверти, изменяем угол по формуле приведения

1.1.3. Если точка находится в 3 или 4 четверти, домножаем угол на -1.

1.1.4. Добавляем получившийся угол в массив углов многоугольника.

Функция void calculate_angle_of_rotation(ray2d& ray, const polygon2d& polygon)

Вычисляет угол поворота луча относительно оси Ox.

Алгоритм работы функции:

1. Если первый угол в многоугольнике ненулевой, тогда используем угол 0.

2. Иначе, находим первый ненулевой угол и используем половину от его значения.

3. Вычисляем тригонометрические характеристики (синус, косинус, тангенс) найденного угла и записываем их лучу.

Функция int count_intersects(const polygon2d& polygon, const ray2d ray)

Считает количество пересечений многоугольника лучём.

Алгоритм работы функции:

1. Для каждого ребра многоугольника:

1.1. Если луч пересекает ребро, увеличить счётчик пересечений на 1.

2. Вернуть количество пересечений.

Функция bool ray_intersects_segment(const point2d first, const point2d last, const ray2d ray)

Определяет, пересекает ли луч отрезок.

Алгоритм работы функции:

1. Вычисляем коэффициенты параметрических уравнений.

2. Проверяем полученные коэффициенты на соответствие условию пересечения.

3. Если проверка прошла успешно, вернуть true, иначе – false.

Приложение Б

Описание структур данных

Структура point2d содержит поля:

int x – значение координаты по оси абсцисс.

int y – значение координаты по оси ординат.

Класс polygon2d содержит поля:

std::vector<point2d> points – набор точек, задающих многоугольник.

std::vector<double> angles – углы между вершинами многоугольника и искомой точкой.

Класс ray2d содержит поля:

int x – значение координаты точки начала луча по оси Ox

int y – значение координаты точки начала луча по оси Oy

double angle – значение угла луча

double sin – значение синуса угла луча

double cos – значение косинуса угла луча

double tg – значение тангенса угла луча

Структура data_check_result содержит поля:

bool is_correct – признак корректные ли данные

std::string reason – причина некорректности данных

Приложение В

Дерево вызова функций

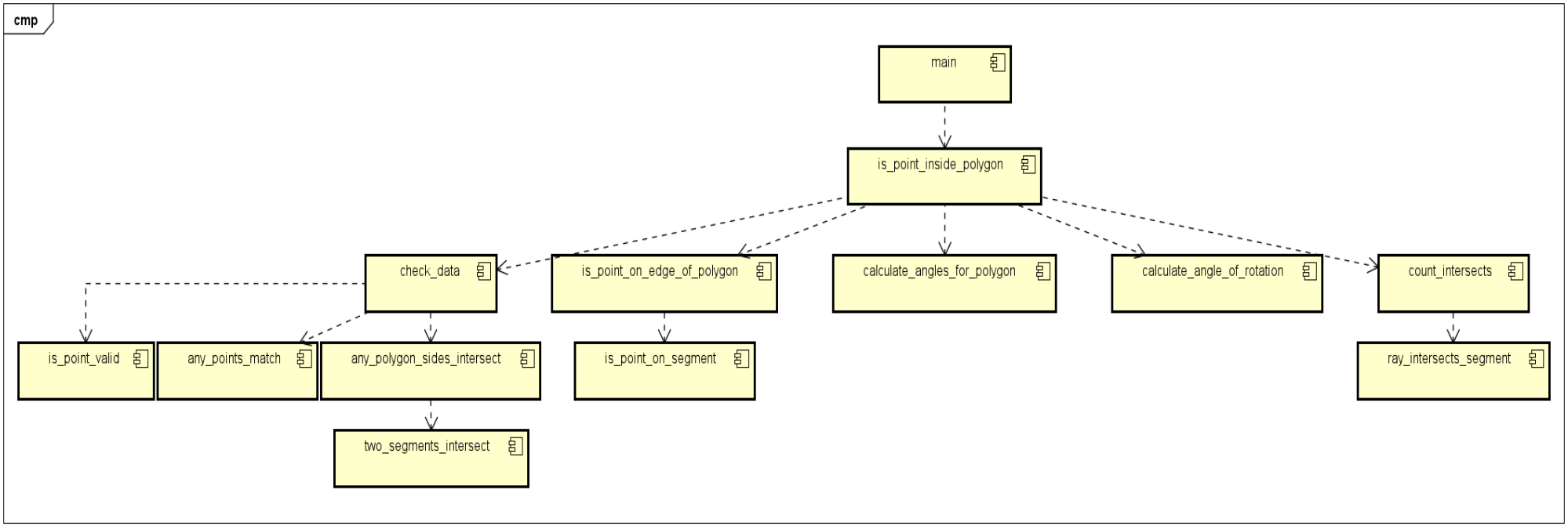

Рис. 1. Дерево вызова функций.

Приложение Г

Диаграмма потоков данных

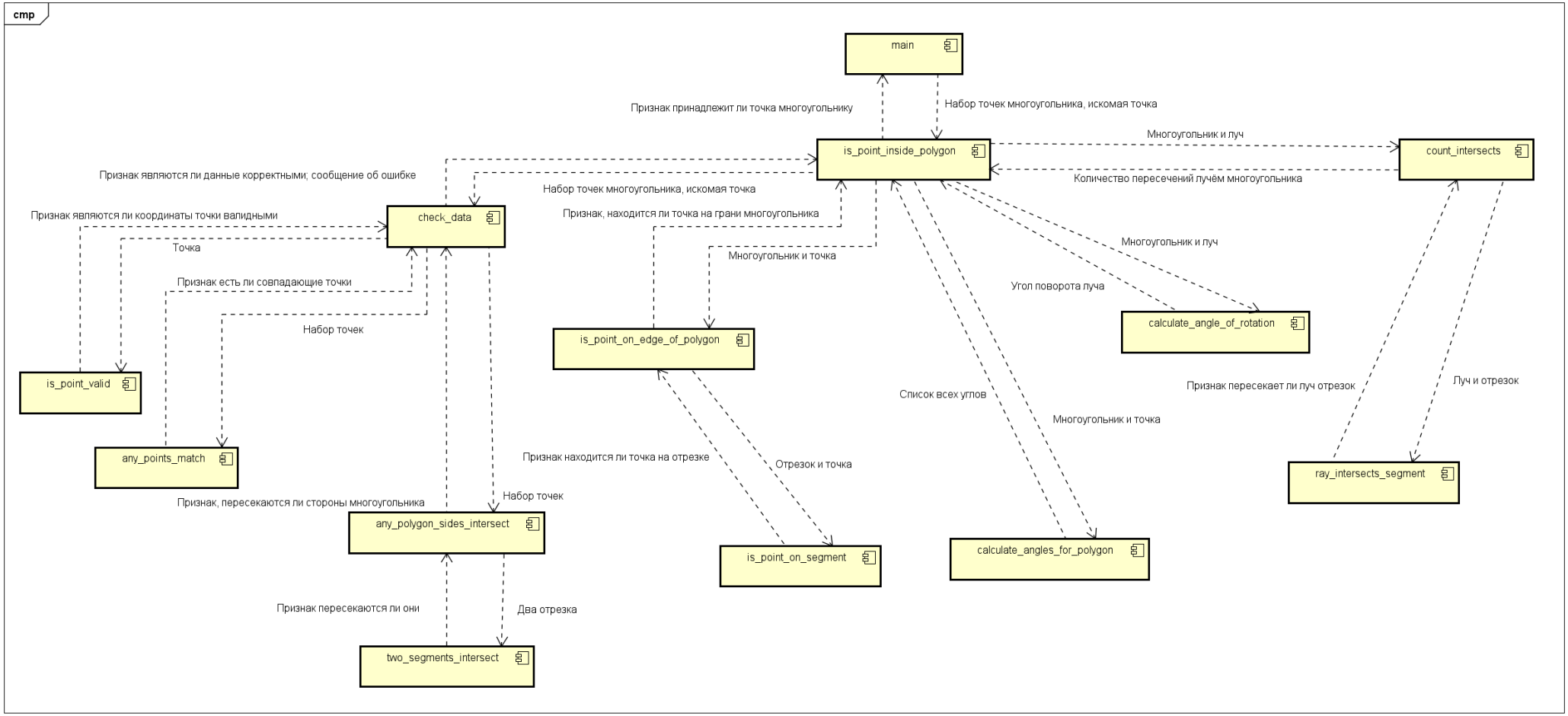

Рис. 2. Диаграмма потоков данных