Определение принадлежности точки невыпуклому многоугольнику	
Программа и методика испытаний	
Студент	Бебахани А. А
Преподаватель	Матюшечкин Д.С.
Сдано	

Содержание

1 Объект испытаний

Программа будет именоваться: программа для определения принадлежности точки невыпуклому многоугольнику (далее – программа). Латинское название программы: PolygonInPoint.

2 Цель испытаний

Проверить соответствие программы требованиям к функциональным характеристикам.

3 Требования к программе

Программа должна определить находится ли данная точка в границах невыпуклого многоугольника.

Многоугольник задаётся набором точек, которые являются его вершинами. Количество точек многоугольника не должно превышать 50 и не должно быть меньше 3. Точка, для которой производится проверка, задаётся последней. Все точки задаются двумя координатами: по оси абсцисс и по оси ординат. Координаты являются целыми числами и должны иметь значение в промежутке от -1000 до 1000.

Если точка принадлежит многоугольнику – выводится 1, иначе – 0.

Требования к надёжности программы, к составу и параметрам технических средств, к информационной и программной совместимости описаны в п. 3.2-3.4 документа «Техническое задание».

4 Требования к программной документации

В бумажной форме должно быть представлено техническое задание, технический проект в виде описания программы, программа и методика испытаний и руководство программиста. В электронной форме должны быть представлены копии всех документов бумажной формы, рабочая документация и текст программы. Всядокументациядолжнабытьпредставлена в соответствии с ГОСТ 19.

5 Средства и порядок испытаний

Для запуска тестов необходимы следующие технические средства: процессор Intel Core i3 и 4 Гб оперативной памяти. Программные средства: ОС Windows 10, Microsoft Visual Studio 2019.

6 Методы испытаний

Юнит-тесты для функций is_point_inside_polygon, count_intersects, check_data, calculate_angles_for_polygon, calculate_angle_of_rotation, is_point_on_edge_of_polygon, is_point_on_segment описаны в приложениях 1-8.

Приложение 1.

Unit-тесты для функции is_point_inside_polygon

Имя функции: is_point_inside_polygon

Функция, решающая главную задачу. Определяет находится ли точка внутри многоугольника.

Функция вызывает:

1) функцию, проверяющую данные (check_data);

2) функцию, проверяющую находится ли точка на грани многоугольника (is_point_on_edge_of_polygon);

3) функцию, считающую углы между искомой точкой и вершинами многоугольника (calculate_angles_for_polygon);

4) функцию, считающую угол поворота проверочного луча (calculate_angle_of_rotation);

5) функцию, считающую количество пересечений лучом граней многоугольника (count_intersects).

Заголовок:

```
bool is_point_inside_polygon(std::vector<point2d> points, const point2d point)
```

Таблица 1. Тесты для функции is_point_inside_polygon

№	Ситуация	Входные данные	Выходные данные
1	Искомая точка совпадает с вершиной многоугольника	Точки многоугольника: {1, 1}, {4, 4}, {6, 1}, {4, 2} Искомая точка: {4, 4}	true
2	Искомая точка находится на грани многоугольника	Точки многоугольника: {1, 1}, {4, 4}, {6, 1}, {4, 2} Искомая точка: {3, 3}	true
3	Искомая точка находится сверху многоугольника	Точки многоугольника: {1, 1}, {4, 4}, {6, 1}, {4, 2} Искомая точка: {1, 5}	false
4	Искомая точка находится под многоугольником	Точки многоугольника: {1, 1}, {4, 4}, {6, 1}, {4, 2} Искомая точка: {0, 0}	false

Продолжение таблицы 1

№	Ситуация	Входные данные	Выходные данные
5	Искомая точка находится справа от многоугольника	Точки многоугольника: {1, 1}, {4, 4}, {6, 1}, {4, 2} Искомая точка: {6, 5}	false
6	Искомая точка находится слева от многоугольника	Точки многоугольника: {1, 1}, {4, 4}, {6, 1}, {4, 2} Искомая точка: {1, 2}	false
7	Сложный многоугольник и точка внутри него	Точки многоугольника: {2, 2}, {2, 5}, {4, 4}, {6, 5}, {6, 2}, {4, 1} Искомая точка: {4, 3}	true

Приложение 2.

Unit-тесты для функции count_intersects

Имя функции: count_intersects.

Проверяет корректность данных, предоставленных пользователем.

Функция вызывает:

1) функцию, проверяющую пересекает ли луч отрезок (ray_intersects_segment).

Заголовок:

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

Таблица 2. Тесты для функции count_intersects

№	Ситуация	Входные данные	Выходные данные
1	Нет пересечений	Многоугольник: {1, 1}, {4, 5}, {6, 1}, {4, 3} Луч: точка = {0, 0}, тангенс угла = 0	0
2	Одно пересечение	Многоугольник: {1, 1}, {4, 5}, {6, 1}, {4, 3} Луч: точка = {4, 4}, тангенс угла = 0	1
3	Два пересечения	Многоугольник: {1, 1}, {4, 5}, {6, 1}, {4, 3} Луч: точка = {1, 4}, тангенс угла = 0	2
4	Четное количество пересечений	Многоугольник: {1, 1}, {4, 5}, {6, 1}, {4, 3} Луч: точка = {0, 2}, тангенс угла = 0	4
5	Нечетное количество пересечений	Многоугольник: {1, 1}, {4, 5}, {6, 1}, {4, 3} Луч: точка = {0, 2}, тангенс угла = 0	3

Приложение 3.

Unit-тесты для функции check_data

Имя функции: check_data.

Считает количество пересечений лучом граней многоугольника.

Функция вызывает:

1) функцию, проверяющую являются ли координаты точки корректные (is_point_valid);

2) функцию, проверяющую совпадают ли вершины многоугольника (any_points_match);

3) функцию, проверяющую пересекаются ли стороны многоугольника (any_polygon_sides_intersect).

Заголовок:

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

Таблица 3. Тесты для функции check_data

№	Ситуация	Входные данные	Выходные данные
1	Все данные корректные	Многоугольник: {2, 2}, {4, 5}, {6, 2}, {4, 3} Искомая точка: {1, 1}	{ true, "" }
2	Слишком мало точек многоугольника	Многоугольник: {2, 2}, {4, 5} Искомая точка: {1, 1}	{ false, "The number of points must be in the range [3; 50]" }
3	Слишком много точек многоугольника	Многоугольник: {2, 2}, {4, 5} Искомая точка: {1, 1}	{ false, "The number of points must be in the range [3; 50]" }

Продолжение таблицы 3

№	Ситуация	Входные данные	Выходные данные
4	Вершины многоугольника совпадают	Многоугольник: {2, 2}, {2, 2}, {6, 2}, {4, 3} Искомая точка: {1, 1}	{ false, "The points of the polygon must not match" }
5	Стороны многоугольника пересекаются	Многоугольник: {2, 2}, {6, 5}, {2, 5}, {6, 2} Искомая точка: {1, 1}	{ false, "The polygon sides must not intersect" }
6	Искомая точка имеет координату с невалидным значением	Многоугольник: {2, 2}, {4, 5}, {6, 2}, {4, 3} Искомая точка: {1, 1500}	{ false, "The coordinates of the point must not exceed the allowed range [-1000; 1000]" }
7	Одна из точек многоугольника имеет координату с невалидным значением	Многоугольник: {2, 2}, {4, 56460561}, {6, 2}, {4, 3} Искомая точка: {1, 1}	{ false, "The coordinates of the polygon's point 2 must not exceed the allowed range [-1000; 1000]" }

Приложение 4.

Unit-тесты для функции calculate_angles_for_polygon

Имя функции: calculate_angles_for_polygon.

Считает углы между искомой точкой и вершинами многоугольника.

Заголовок:

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

Таблица 4. Тесты для функции calculate_angles_for_polygon

№	Ситуация	Входные данные	Выходные данные
1	Один из углов равен нулю	Многоугольник: {3, 8}, {6, 1}, {3, 3}, {4, 4} Точка: {3, 4}	Углы многоугольника: {90, -45, -90, 0}
2	Два из углов равен нулю	Многоугольник: {4, 2}, {5, 5}, {7, 2}, {2, 1} Точка: {2, 2}	Углы многоугольника: {0, 45, 0, -90}
3	Несколько углов равны нулю	Многоугольник: {2, 6}, {6, 4}, {8, 4}, {2, 2}, {4, 4} Точка: {2, 4}	Углы многоугольника: {90, 0, 0, -90, 0}
4	Один из углов равен 90	Многоугольник: {2, 6}, {6, 4}, {2, 2}, {4, 4} Точка: {2, 4}	Углы многоугольника: {90, 0, -90, 0}
5	Один из углов равен 180	Многоугольник: {4, 4}, {4, 6}, {2, 4}, {4, 2}, {6, 3} Точка: {6, 4}	Углы многоугольника: {180, 135, 180, -135, -90}
6	Один из углов равен 270	Многоугольник: {2, 6}, {6, 4}, {2, 2}, {4, 4} Точка: {2, 4}	Углы многоугольника: {90, 0, -90, 0}
7	Углы расположены в первой четверти	Многоугольник: {3, 3}, {6, 6}, {8, 3}, {6, 4} Точка: {2, 2}	Углы многоугольника: {45, 45, 9.464, 26.565}
8	Углы расположены во второй четверти	Многоугольник: {2, 7}, {6, 9}, {5, 7}, {5, 4} Точка: {7, 2}	Углы многоугольника: {135, 98.132, 111.801, 135}
9	Углы расположены в третьей четверти	Многоугольник: {1, 7}, {4, 6}, {7, 7}, {3, 3} Точка: {8, 8}	Углы многоугольника: {-171.87, -153.435, -135, -135}
10	Углы расположены в четвертой четверти	Многоугольник: {3, 7}, {7, 3}, {3, 2}, {5, 4} Точка: {2, 8}	Углы многоугольника: {-45, -45, -80.538, -53.13}
11	Углы расположены в всех четвертях	Многоугольник: {7, 6}, {6, 4}, {7, 2}, {5, 3}, {3, 2}, {4, 4}, {3, 6}, {5, 5} Точка: {5, 4}	Углы многоугольника: {45, 0, -45, -90, -135, 180, 135, 90}

Приложение 5.

Unit-тесты для функции calculate_angle_of_rotation

Имя функции: calculate_angle_of_rotation.

Считает угол поворота проверочного луча.

Заголовок:

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

Таблица 5. Тесты для функции calculate_angle_of_rotation

№	Ситуация	Входные данные	Выходные данные
1	Один из углов многоугольника равен 0	Углы многоугольника: {0, 90, 90, 180} Луч: точка = {0, 0}	Угол луча: 45
2	Два из углов многоугольника равны 0	Углы многоугольника: {0, 0, 20, 60} Луч: точка = {0, 0}	Угол луча: 10
2	Несколько из углов многоугольника равны 0	Углы многоугольника: {0, 0, 0, 0, 15, -20, 30, -30, -45} Луч: точка = {0, 0}	Угол луча: 7,5
3	Углы многоугольника в первой четверти	Углы многоугольника: {15, 20, -60, 70} Луч: точка = {0, 0}	Угол луча: 0
4	Углы многоугольника во второй четверти	Углы многоугольника: {110, -130, 150, 150, 160, 180} Луч: точка = {0, 0}	Угол луча: 0
5	Углы многоугольника в третьей четверти	Углы многоугольника: {-100, 150, -150, -170, 180} Луч: точка = {0, 0}	Угол луча: 0
6	Углы многоугольника в четвертой четверти	Углы многоугольника: {-20, 90, -90, -135} Луч: точка = {0, 0}	Угол луча: 0

Приложение 6.

Unit-тесты для функции is_point_on_edge_of_polygon

Имя функции: is_point_on_edge_of_polygon.

Проверяет находится ли точка на грани многоугольника.

Функция вызывает:

1) функцию, проверяющую находится ли точка на отрезке (is_point_on_segment).

Заголовок:

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

Таблица 6. Тесты для функции is_point_on_edge_of_polygon

№	Ситуация	Входные данные	Выходные данные
1	Точка на одной из граней	Многоугольник: {2, 5}, {5, 5}, {4, 4}, {5, 2}, {2, 2} Точка: {2, 4}	true
2	Точка является вершиной	Многоугольник: {2, 5}, {5, 5}, {4, 4}, {5, 2}, {2, 2} Точка: {2, 5}	true
3	Точка внутри многоугольника	Многоугольник: {2, 5}, {5, 5}, {4, 4}, {5, 2}, {2, 2} Точка: {3, 4}	false
4	Точка снаружи многоугольника	Многоугольник: {2, 5}, {5, 5}, {4, 4}, {5, 2}, {2, 2} Точка: {2, 1}	false

Приложение 7.

Unit-тесты для функции is_point_on_segment

Имя функции: is_point_on_segment.

Проверяет является ли точка на отрезке.

Заголовок:

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

Таблица 7. Тесты для функции is_point_on_segment

№	Ситуация	Входные данные	Выходные данные
1	Точка является вершиной отрезка	first = {1, 1} last = {4, 2} point ={1, 1}	true
2	Точка на вертикальном отрезке	first = {1, 1} last = {2, 5} point ={2, 4}	true
3	Точка на горизонтальном отрезке	first = {1, 1} last = {4, 1} point ={2, 1}	true
4	Точка на отрезке	first = {3, 2} last = {6, 5} point ={4, 3}	true
5	Точка ниже горизонтальной стороны отрезка	first = {1, 1} last = {4, 2} point ={2, 0}	false
4	Точка справа от горизонтальной стороны отрезка	first = {1, 1} last = {4, 2} point ={6, 1}	false
5	Точка выше вертикальной стороны отрезка	first = {1, 1} last = {2, 5} point ={2, 6}	false
6	Точка слева от вертикальной стороны отрезка	first = {2, 2} last = {2, 5} point ={1, 3}	false
7	Точка находится вне прямоугольника, который образует отрезок	first = {3, 2} last = {6, 5} point ={9, 0}	false

Приложение 8.

Unit-тесты для функции ray_intersects_segment

Имя функции: ray_intersects_segment.

Проверяет пересекает ли луч отрезок.

Заголовок:

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

Таблица 8. Тесты для функции ray_intersects_segment

№	Ситуация	Входные данные	Выходные данные
1	Луч и отрезок параллельны	Отрезок: {2, 2}, {4, 2} Луч: {1, 1}, тангенс = 0	false
2	Луч и отрезок находятся на одной прямой, но не пересекаются	Отрезок: {1, 1}, {3, 1} Луч: {4, 1}, тангенс = 0	false
3	Луч и отрезок находятся на одной прямой и пересекаются	Отрезок: {3, 3}, {4, 4} Луч: {2, 2}, тангенс = 1	true
4	Луч и отрезок пересекаются крест-накрест	Отрезок: {1, 2}, {3, 1} Луч: {2, 1}, тангенс = 1	true
5	Луч и отрезок перпендикулярны	Отрезок: {2, 2}, {3, 1} Луч: {2, 1}, тангенс = 1	true
6	Луч пересекает отрезок	Отрезок: {2, 2}, {5, 1} Луч: {2, 2}, тангенс = 1	true
7	Луч пересекает отрезок	Отрезок: {2, 2}, {4, 4} Луч: {3, 3}, тангенс = 0	true
8	Луч начинается в вершине отрезка	Отрезок: {2, 2}, {5, 2} Луч: {2, 2}, тангенс = 1	true

Приложение 9.

Unit-тесты для функции two_segments_intersect

Имя функции: two_segments_intersect.

Проверяет пересекаются ли два отрезка.

Заголовок:

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

Таблица 9. Тесты для функции two_segments_intersect

№	Ситуация	Входные данные	Выходные данные
	Отрезки параллельны	Отрезок 1: {1, 1}, {1, 3} Отрезок 2: {2, 1}, {2, 3}	false
	Отрезки лежат на одной прямой, но не пересекаются	Отрезок 1: {1, 1}, {1, 3} Отрезок 2: {1, 4}, {1, 6}	false
	Отрезки лежат на одной прямой и пересекаются	Отрезок 1: {2, 1}, {2, 3} Отрезок 2: {2, 2}, {2, 4}	true
	Отрезки перпендикулярны	Отрезок 1: {1, 2}, {3, 2} Отрезок 2: {2, 1}, {2, 3}	true
	Отрезки пересекаются крест-накрест	Отрезок 1: {1, 3}, {3, 2} Отрезок 2: {1, 2}, {3, 3}	true
	Отрезки одинаковы	Отрезок 1: {1, 3}, {3, 2} Отрезок 2: {1, 3}, {3, 2}	true
	Отрезки имеют одинаковую вершину	Отрезок 1: {2, 1}, {2, 3} Отрезок 2: {1, 2}, {3, 1}	true