Лабораторная работа 4.

# Задание 1.

Требуется выполнить этап проектирования для реализации следующей задачи: построить Выпуклую Оболочку Convex последовательно поступающих точек плоскости, на каждом шаге вычислять ее периметр и площадь.

## **Инициализация Выпуклой Оболочки.**

1. **Convex**.

Название: Convex – “Инициализировать Выпуклую Оболочку, объект типа Convex”.

Входные аргументы: нет.

Что делает: пользовательская функция Convex инициализирует объект типа Convex в виде Convex[{список координат крайних точек}, {список координат внутренних точек}, периметр, площадь].

Выход, структура результата: объект типа Convex в виде Convex[{}, {}, 0, 0].

Какие функции вызывает: нет.

## **Отображение Выпуклой Оболочки.**

1. **Format**.

Название: Format – “Отобразить Выпуклую Оболочку в печатном виде”.

Входные аргументы: объект CH типа Convex.

Что делает: встроенная функция Format печатает части объекта типа Convex. Ассоциируется с головой-символом выражения Convex, заносится в список FormatValues его значений.

Выход, структура результата: печатный образ подаваемого на вход объекта типа Convex в виде Сonvex[{<<число крайних точек>>}, {<<число внутренних точек>>}, периметр, площадь].

Какие функции вызывает: MapAt, Shallow – встроенные функции.

1. **Graphics**.

Название: Graphics – “Отобразить Выпуклую Оболочку в графическом формате”.

Входные аргументы: объект CH типа Convex, дополнительные настройки свойств графического отображения options типа List.

Что делает: встроенная функция Graphics отображает внутренние точки Выпуклой Оболочки, крайние точки Выпуклой Оболочки и линии, соединяющие крайние точки Выпуклой Оболочки.

Выход, структура результата: графический образ подаваемого на вход объекта типа Convex.

Какие функции вызывает: If, Hue, Append, Line, PointSize, Point, PlotRange – встроенные функции.

## **Добавление точки к Выпуклой Оболочке.**

### Обработка вырожденных случаев.

* Пустая Выпуклая оболочка – нет точек.

1. **Add.**

Название: Add – “Добавить точку к Выпуклой Оболочке”.

Входные аргументы: объект CH типа Convex вида Convex[{}, {}, 0, 0], объект Point типа kmPoint.

Что делает: пользовательская функция Add добавляет точку Point в список крайних точек Выпуклой Оболочки CH.

Выход, структура результата: “одноугольник” – Выпуклая Оболочка CH с одной крайней точкой Point в виде Convex[{Point}, {}, 0, 0].

Какие функции вызывает: нет.

* Нульмерная Выпуклая оболочка – одна точка.

1. **Add.**

Название: Add – “Добавить точку к Выпуклой Оболочке”.

Входные аргументы: объект CH типа Convex вида Convex[{A}, {}, 0, 0], объект Point типа kmPoint.

Что делает:

Если точка Point совпадает с уже имеющейся точкой A из списка крайних точек Выпуклой Оболочки, то Выпуклая Оболочка не меняется;

Иначе пользовательская функция Add добавляет точку Point в список крайних точек Выпуклой Оболочки CH и с помощью функции Distance вычисляется длина нового отрезка, образуемого крайними точками, полученное значение вставляется на место нулевого периметра Выпуклой Оболочки CH.

Выход, структура результата:

Если точка Point совпадает с уже имеющейся точкой A из списка крайних точек Выпуклой Оболочки, то Выпуклая Оболочка не меняется, на выходе будет “одноугольник” – Выпуклая Оболочка CH с одной крайней точкой A в виде Convex[{Point}, {}, 0, 0].

Иначе на выходе будет “двухугольник” – Выпуклая Оболочка CH с двумя крайними точками A и Point в виде Convex[{A, Point}, {}, длина APoint, 0].

Какие функции вызывает: Distance – пользовательская функция.

1. **Distance**.

Название: Distance – “Посчитать расстояние между двумя точками”.

Входные аргументы: объект Point1 типа kmPoint, объект Point2 типа kmPoint.

Что делает: пользовательская функция Distance считает расстояние между двумя точками Point1 и Point2.

Выход, структура результата: действительное число.

Какие функции вызывает: Sqrt, Dot, Function: & – встроенные функции.

* Одномерная Выпуклая оболочка – отрезок.

1. **Add**.

Название: Add – “Добавить точку к Выпуклой Оболочке”.

Входные аргументы: объект CH типа Convex вида Convex[{A, B}, {... — одна внутренняя точка, или несколько внутренних точек или ни одной внутренней точки отрезка AB}, Длина AB, 0], объект Point типа kmPoint.

Что делает:

Если очередная точка Point совпадает с одной из точек A или B списка крайних точек Выпуклой оболочки CH, то Выпуклая Оболочка не меняется;

Иначе, если Point лежит внутри отрезка AB, то Point добавляется к списку внутренних точек Выпуклой Оболочки CH;

Иначе, если Point лежит на прямой AB таким образом, что длина AB меньше длины длины APoint, то точка B удаляется из списка крайних точек Выпуклой Оболочки CH и добавляется в список внутренних точек Выпуклой Оболочки CH, Point добавляется в конец списка крайних точек Выпуклой Оболочки CH, с помощью функции Distance вычисляется длина нового отрезка, образуемого крайними точками, полученное значение обновляет текущее значение периметра Выпуклой Оболочки CH;

Иначе, если Point лежит на прямой AB таким образом, что длина AB меньше длины длины BPoint, то точка A удаляется из списка крайних точек Выпуклой Оболочки CH и добавляется в список внутренних точек Выпуклой Оболочки CH, Point добавляется в начало списка крайних точек Выпуклой Оболочки CH, с помощью функции Distance вычисляется длина нового отрезка, образуемого крайними точками, полученное значение обновляет текущее значение периметра Выпуклой Оболочки CH;

Иначе Point добавляется к списку крайних точек Выпуклой Оболочки CH, список сортируется с помощью функции алгебраической площади getTriangleAlgebraicSquare точек A, Point, B: если getTriangleAlgebraicSquare точек A, Point, B больше нуля, то точка Point лежит справа от направленного отрезка и список конечных точек Выпуклой Оболочки CH будет иметь вид: {A, Point, B}, иначе {A, B, Point}. С помощью функции getPolygonSquare вычисляется площади, образуемой крайними точками, полученное значение вставляется на место нулевой площади Выпуклой Оболочки CH.

Выход, структура результата:

Если очередная точка Point совпадает с одной из точек A или B списка крайних точек Выпуклой оболочки CH, то Выпуклая Оболочка не меняется, на выходе будет “двухугольник” – Выпуклая Оболочка CH с двумя крайними точками A и B и некоторыми внутренними точками в виде Convex[{A, B}, {...}, Длина AB, 0].

Иначе, если Point лежит внутри отрезка AB, на выходе будет “двухугольник” – Выпуклая Оболочка CH с двумя крайними точками A и B плюс одной внутренней точкой Point в виде Convex[{A, B}, {..., Point}, Длина AB, 0];

Иначе, если Point лежит на прямой AB таким образом, что длина AB меньше длины длины APoint, на выходе будет “двухугольник” – Выпуклая Оболочка CH с двумя крайними точками A и Point и плюс одной внутренней точкой B в виде Convex[{A, Point}, {..., B}, Длина APoint, 0];

Иначе, если Point лежит на прямой AB таким образом, что длина AB меньше длины длины BPoint, на выходе будет “двухугольник” – Выпуклая Оболочка CH с двумя крайними точками Point и B и плюс одной внутренней точкой A в виде Convex[{Point, B}, {..., A}, Длина AB, 0];

Иначе на выходе будет “треугольник” – Выпуклая Оболочка CH с тремя крайними точками Point и B и некоторыми внутренними точками в виде Convex[{A, B, Point} или {A, Point, B}, {...}, Длина AB, 0]

Какие функции вызывает: Distance, getPolygonSquare – пользовательские функции.

1. **getPolygonSquare**.

Название: getPolygonSquare – “Найти площадь полигона, образуемого крайними точками Векторной Оболочки CH”.

Входные аргументы: объект CH типа Convex вида Convex[{список конечных точек}, {... — одна внутренняя точка, или несколько внутренних точек или ни одной внутренней точки отрезка AB}, Длина Выпуклой Оболочки или 0, Площадь выпуклой оболочки].

Что делает: пользовательская функция getPolygonSquare разбивает список крайних точек Выпуклой Оболочки CH на множество списков из трех точек – всевозможные треугольники, полученные пересечением крайних точек между собой. Затем считается модуль суммы алгебраической площади каждого из полученных getTriangleAlgebraicSquare треугольников. Это и есть искомая площадь.

Выход, структура результата: действительное число.

Какие функции вызывает: Abs, Plus, @@, &, Map – встроенные функции, getTriangleAlgebraicSquare – пользовательская функция.

1. **getTriangleAlgebraicSquare**.

Название: getTriangleAlgebraicSquare – “Найти найти алгебраическую площадь треугольника”.

Входные аргументы: объект Point1 типа kmPoint, объект Point2 типа kmPoint, объект Point3 типа kmPoint.

Что делает: считает половину определителя выражения {Point2 - Point1, Point3 - Point1} с помощью встроенной функции Det.

Выход, структура результата: действительное число.

Какие функции вызывает: Det – встроенная функция.

### Обработка регулярного случая.

* Многоугольник.

1. **Add**.

Название: Add – “Добавить точку к Выпуклой Оболочке”.

Входные аргументы: объект CH типа Convex вида Convex[{... — одна крайняя точка, или несколько крайних точек или ни одной крайней точки Выпуклой Оболочки}, {... — одна внутренняя точка, или несколько внутренних точек или ни одной внутренней точки отрезка AB}, Периметр Выпуклой Оболочки, Площадь Выпуклой оболочки], объект Point типа kmPoint.

Что делает: добавляет или не добавляет точку Point в список крайних точек Выпуклой Оболочки CH или список внутренних точек Выпуклой Оболочки CH; после добавления точки Point в список крайних точек Выпуклой Оболочки CH или список внутренних точек Выпуклой Оболочки CH обновляет значения периметра и площади Выпуклой Оболочки CH.

Если вызываемая функция FoundVisible не находит ни одного “видимого” из точки Point ребра, образуемого крайними точками Выпуклой Оболочки CH, то Выпуклая Оболочка CH не меняется.

Инициализируется список positionsOfPointsToRemove.

Далее, пока функция FoundVisible будет находить очередное “видимое” из точки Point ребро, это ребро будет удаляться из Выпуклой Оболочки CH, с помощью вызова функции RemoveEdge, обновляя периметр и площадь Выпуклой Оболочки CH. Положение правой крайней точки в найденном “видимом” из точки Point ребре в списке крайних точек Выпуклой Оболочки CH запоминается и добавляется в список positionsOfPointsToRemove.

Из списка positionsOfPointsToRemove достается последняя точка: сохраняется в переменную PointToInsertBefore и удаляется из списка positionsOfPointsToRemove, так как она не должна удаляться из списка крайних точек Выпуклой Оболочки CH .

С помощью встроенной функции Delete, из списка крайних точек Выпуклой Оболочки CH удаляются позиции, содержащиеся в списке positionsOfPointsToRemove.

Точка Point вставляется в список крайних точек Выпуклой Оболочки CH на позицию, находящуюся до позиции точки PointToInsertBefore в обновленном списке крайних точек Выпуклой Оболочки CH.

Важно: почему я не ищу ребра до очередного найденного “видимого” из точки Point ребра? Функция FoundVisible работает таким образом, что находит “видимые ребра” последовательно, по порядку обхода ребер против часовой стрелки. А все точки из списка крайних точек выпуклой оболочки в описанном алгоритме и в алгоритме добавления точки отрезку добавляются таким образом, что обход точек против часовой стрелки не нарушается.

Выход, структура результата: “многоугольник” – Выпуклая Оболочка CH с некоторым количеством крайних и внутренних точек, периметром и площадью в виде Convex[{...}, {...}, Периметр Выпуклой Оболочки, Площадь Выпуклой Оболочки].

Какие функции вызывает: нет.

1. **FoundVisible**.

Название: FoundVisible – “Найти одно видимое ребро”.

Входные аргументы: объект CH типа Convex вида Convex[{... — одна крайняя точка, или несколько крайних точек или ни одной крайней точки Выпуклой Оболочки}, {... — одна внутренняя точка, или несколько внутренних точек или ни одной внутренней точки отрезка AB}, Периметр Выпуклой Оболочки, Площадь Выпуклой оболочки], объект Point типа kmPoint.

Что делает: пользовательская функция FoundVisible находит “видимое” ребро среди крайних точек Выпуклой Оболочки CH.

Ребро, образуемое первой точкой из списка крайних точек Выпуклой Оболочки CH и последней точкой из списка крайних точек Выпуклой Оболочки CH должно проверяться отдельно на “видимость” из точки Point, до начала проверки всех остальных точек.

Затем, пробегая по каждому ребру, образуемому текущей точкой и следующей точкой из списка крайних точек Выпуклой Оболочки CH, производится проверка на “видимость” ребра из точки Point. Если такое ребро найдено, функция завершает работу и отправляет результат.

Выход, структура результата: если было найдено “видимое” из точки Point ребро, то на выходе будет список из двух точек, образующих это ребро; если не было найдено ни одного “видимого” ребра, на выходе будет значение в виде литерала False.

Какие функции вызывает: Catch, Throw, Do, If – встроенные функции, VisibleQ – пользовательская функция.

1. **VisibleQ**.

Название: VisibleQ – “Является ли ребро видимым”.

Входные аргументы: объект A типа kmPoint, объект B типа kmPoint, объект Point типа kmPoint.

Что делает: пользовательская функция VisibleQ определяет, является ли отрезок “видимым” из точки Point.

Если алгебраическая площадь getTriangleAlgebraicSquare точек A, Point, B меньше нуля, что означает, точка Point находится справа от направленного отрезка AB, или, если алгебраическая площадь getTriangleAlgebraicSquare точек A, Point, B равна нулю и точка Point не принадлежит отрезку AB, то отрезок AB является “видимым” из точки Point, иначе отрезок AB является “не видимым” из точки Point.

Выход, структура результата: литерал True – отрезок AB является “видимым” из точки Point, False – отрезок AB является “не видимым” из точки Point.

Какие функции вызывает: If, Or, Block, Less – встроенные функция, getTriangleAlgebraicSquare. – пользовательская функция.

1. **RemoveEdge**.

Название: RemoveEdge – “Обработать видимое ребро”.

Входные аргументы: объект CH типа Convex вида Convex[{... — одна крайняя точка, или несколько крайних точек или ни одной крайней точки Выпуклой Оболочки}, {... — одна внутренняя точка, или несколько внутренних точек или ни одной внутренней точки отрезка AB}, Периметр Выпуклой Оболочки, Площадь Выпуклой оболочки], объект AB типа kmSegment, объект Point типа kmPoint.

Что делает: пользовательская функция RemoveEdge подготавливает к удалению крайние точки A и B из списка крайних точке Выпуклой Оболочки CH, а именно обновляет периметр и площадь Выпуклой Оболочки CH без ребра AB.

Сначала периметр Выпуклой Оболочки уменьшается длину AB.

Затем прибавляю модуль алгебраической площади getTriangleAlgebraicSquare точек A, Point, B. к площади Выпуклой Оболочки CH.

Выход, структура результата: нет.

Какие функции вызывает: Distance, getTriangleAlgebraicSquare – пользовательские функция.

# Задание 2.

Требуется выполнить этап проектирования для следующей задачи: найти все пары пересекающихся отрезков заданного множества отрезков прямой на плоскости.

1. **getPairsOfIntersectedSegments**.

Название: getPairsOfIntersectedSegments– “Найти пары пересекающихся отрезков”.

Входные аргументы: список segments типа List состоящий из объектов типа kmSegment.

Что делает: пользовательская функция getPairsOfIntersectedSegments определяет пары пересекающихся отрезков методом заметающей прямой.

Инициализируется список T, содержащий отрезки, характеризующие состояние относительно заметающей прямой.

Инициализируется список result, хранящий результирующий список списков пар пересекающихся отрезков списка segment.

Конечные точки отрезков списка segments сортируются по возрастанию координатой x. Если у каких-то точек координаты x совпадают, эти точки сортируются по возрастанию координатой y.

Пробегая по каждой точке отсортированного списка очередная точка проверяется:

Если это левая крайняя точка какого-то отрезка s из списка segments, отрезок s добавляется в конец списка T.

Если в списке T найдется отрезок, находящийся выше отрезка s по координате y, или в списке T найдется отрезок, находящийся ниже отрезка s по координате y, то проверяется пересечение найденного отрезка с отрезком s. Если они пересекаются, оба отрезка добавляются в список result в виде списка из двух отрезков {s, найденный отрезок}.

Иначе это правая точка какого-то отрезка s из списка segments. Если для отрезка s в списке T найдутся и верхний и нижний отрезки, причем верхний отрезок пересекается с нижним, оба отрезка добавляются в список result в виде списка из двух отрезков {нижний отрезок, верхний отрезок}. Отрезок s удаляется из списка T.

В конце возвращается список result

Выход, структура результата: возвращает список из списков пар пересекающихся отрезков.

Какие функции вызывает: Module, Position, ContainsAny, DeleteCases, Append, Or, And – пользовательские функция.