**Полигоны. Теория**

Имеем набор точек (полигон), передаваемый на обработку в функцию в качестве линейного однонаправленного списка. Задача: составить из точек выпуклый (если нет, то вывести сообщение об ошибке) многоугольник.

Определим содержание метода и выберем математическую модель, наиболее подходящую для его реализации. Критерием того, что многоугольник является выпуклым, будет выполнение условия одинакового поворота для всех точек, выполняя обход против/по часовой стрелке. Рассмотрим три варианта решения.

**Последовательный обход набора точек**

Не прибегая к различного рода усложнениям, составим многоугольник, последовательно обойдя все точки из исходного списка (не забываем про связь последней и нулевой). Но сначала нужно установить, как осуществлять обход: если для первых трех точек условие левого поворота является истинным, то последующую проверку остальных делаем относительно также левого поворота, иначе — правого. Но если первые три точки образуют два коллинеарных сонаправленных вектора, то тогда проверяем следующие три точки, считая от той, что была второй в первой тройке.

*Левый/правый*поворот — это поворот по наименьшему углу от одного вектора к другому *против* часовой стрелки или *по* ней. В решении будем использовать свойство векторного произведения двух векторов, что если z-координата результирующего вектора положительна, то поворот был против часовой стрелки, если отрицательна, то — по часовой, если же z-координата равна нулю, то векторы коллинеарны и сонаправлены. Например: 1) даны три точки в пространстве (Рисунок 1): A(3; 2; 0), B(7; 5; 0) и C(4; 7; 0); соединив их последовательно получим два вектора — *AB*{4; 3; 0} и *BC*{-3; 2; 0}. Их векторное произведение будет равно вектору *a* с координатами {0; 0; 17}, z-координата положительна, следовательно, это левый поворот. 2) Если поменять местами точки B и С (Рисунок 2), то поворот будет уже правым: *AB* {1; 5; 0} и *BC* {3; -2; 0}, тогда — *a*{0; 0; -17}, как видим, z-координата отрицательна.

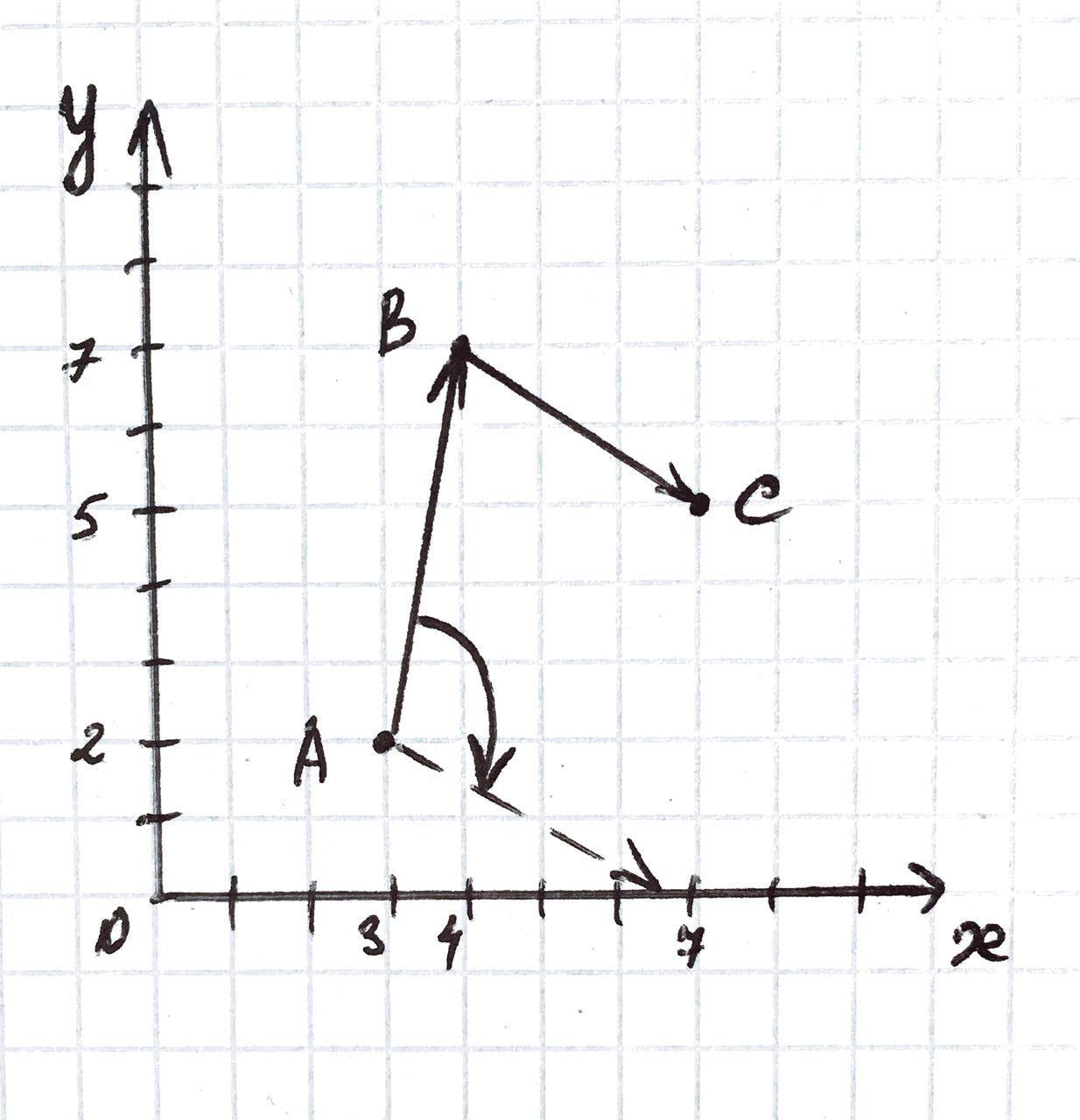
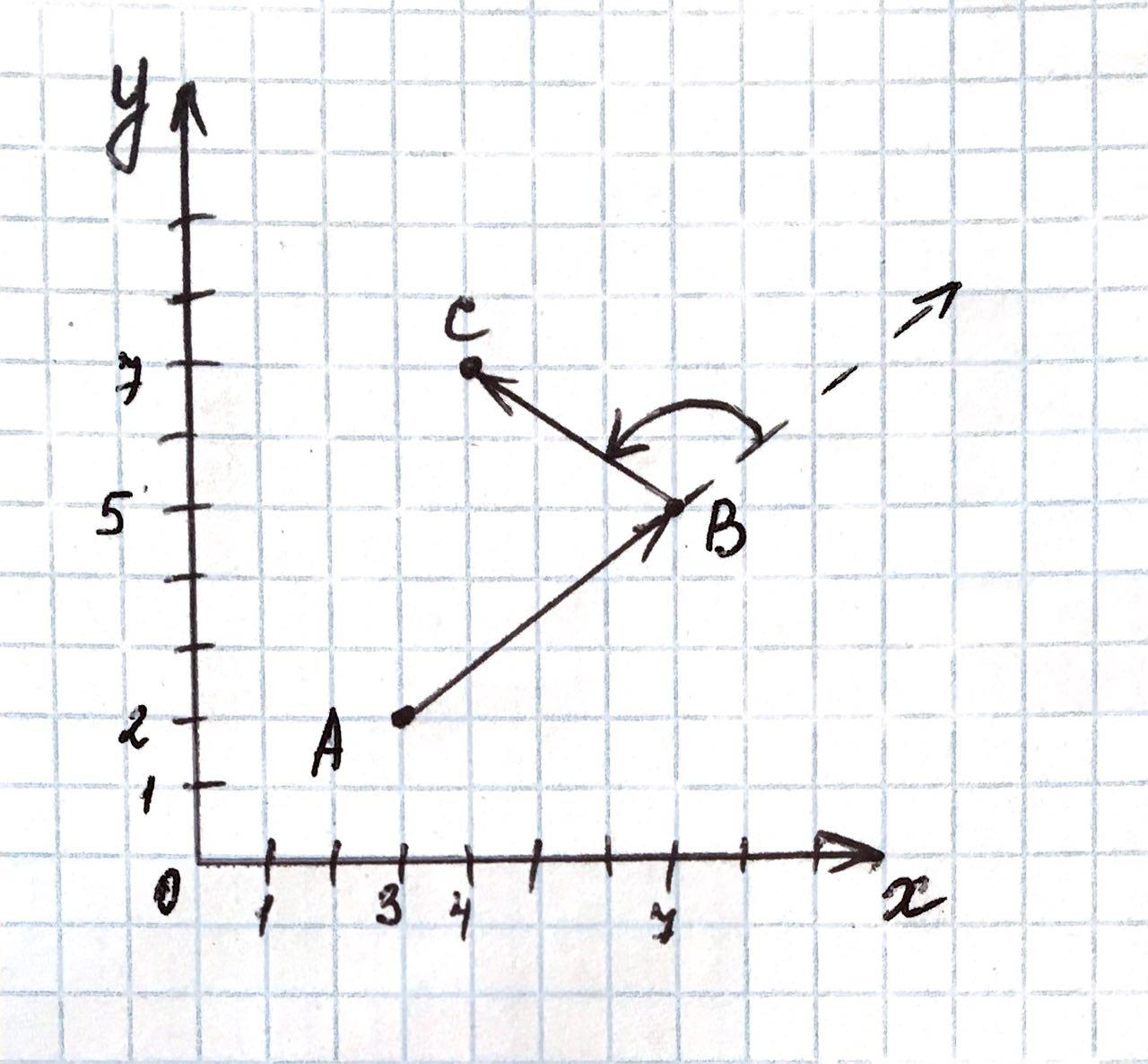


Рисунок 1 Рисунок 2

Таким образом, если при проверке на левый поворот выполняется условие правого (или наоборот), то искомый многоугольник является невыпуклым, значит, выводим соответствующее сообщение об ошибке (еще раз не забываем про связь последней и нулевой точки); если же два вектора оказались коллинеарными и сонаправленными, то они образуют одну сторону искомого многоугольника.

|  |
| --- |
| *Замечание*. Проверка единства поворота не отлавливает петли, поэтому для корректной работы с примерами, содержащие их, необходимо дополнительно проверять не пересекает ли следующий вектор (отрезок) уже проверенные векторы (отрезки). |

**Последовательный обход отсортированного набора точек**

Велика вероятность, что из имеющегося набора точек все же можно получить выпуклый многоугольник, но соединять точки нужно не по порядку, а в иной последовательности.

Найдем точку с наименьшей координатой *y* и переместим ее на нулевое место списка. Далее отсортируем точки (все, кроме нулевой) по степени возрастания «левизны» против часовой стрелки относительно нулевой точки (сортировка вставками): внешний цикл — от 2-го элемента до последнего, внутренний цикл — пока не первый элемент, считая от индекса внешнего цикла i, и векторы [нулевая точка → точка (i -1)] и [точка (i - 1) → точка i] образуют правый поворот, меняем местами точку (i - 1) и точку i.

Следующим шагом прибегаем к действиям из части «Последовательный обход набора точек».

**Построение выпуклой оболочки набора точек**

Даже совершив обход отсортированного набора точек, мы все еще можем получить невыпуклый многоугольник, поэтому повысим вероятность его появления до максимума, соединив не все точки, а только те, которые образуют наименьший выпуклый многоугольник, содержащий остальные точки набора. Для этого будем использовать алгоритм Грэхема ([подробнее](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%93%D1%80%D1%8D%D1%85%D0%B5%D0%BC%D0%B0)).

Исходный список лучше сделать двунаправленным. Затем сам алгоритм: просматриваем точки (от второй и до конца, нулевую и первую считаем проверенными) и отслеживаем направление поворота в них с точки зрения последних двух проверенных точек: если это правый поворот, то можно срезать угол удалением из списка последней проверенной точки. Как только поворот оказывается левым, то срезание углов завершается.

В итоге получаем список точек, образующих искомый выпуклый многоугольник.