Оглавление

[Задание 1 2](#_Toc137883658)

[Задание 2 3](#_Toc137883659)

[Задание 3 4](#_Toc137883660)

[Задание 4 5](#_Toc137883661)

[Задание 5 9](#_Toc137883662)

# Задание 1

**Задание1:**

В огромном офисе компании GraviLink кабинет Матвея находится на координатах (13, -1, 42), а кабинет Дениса – на координатах (7, 0, 3). Найдите координаты кабинета Дениса в системе отсчёта кабинета Матвея.

**Решение:**

Денис стоит относительно себяна нулевых координатах (Д(0,0,0)), следовательно, нам надо вычесть расстояние

* X: 7 - 13 = -6
* Y: 0 – (-1) = 1
* Z: 3 - 42 = -39

**Ответ:**

Относительно Матвея кабинет Дениса находится по координатам (-6,1,-39)

# Задание 2

Задание2:

Талгату на день рождения подарили многочлен. Многочлен ему понравился, поэтому он записал его производную и значение в точке 0 на листочек. На следующее утро Талгат забыл свой любимый многочлен, из-за чего очень расстроился. Можно ли по имеющейся информации помочь Талгату восстановить многочлен?  
Пусть p(x) — многочлен Талгата.

**Решение:**

Нам дана производная функции и её значение в точке 0. Для нахождения самой функции возьмем интеграл

Однако наша функция в точке p(0) даст нам 4, следовательно, итоговым ответом будет

**Ответ:**

Исходная функция имеет следующий вид:

# Задание 3

Задание3:

У Максима дома есть 2 одинаковые свиньи-копилки. Каждый раз возвращаясь после магазина он кладет одну монетку из сдачи в одну из копилок (равновероятно – его младшие сестры постоянно их переставляют местами), а оставшиеся отдает маме. В копилку может поместиться N монет. Изначально копилки пустые. Какова вероятность того, что в момент заполнения одной из копилок, в другой будет ровно K монет?

**Решение:**

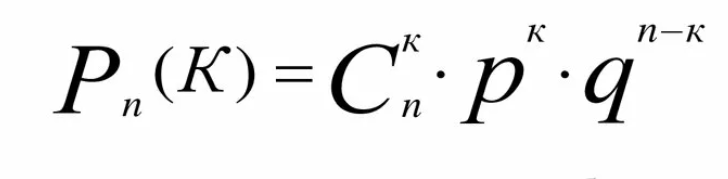
Разделим двух свиней: С1 и С2. Шанс положить монету в каждую из них равны 0.5.

Перефразируем условие задачи:

1: В момент заполнения одной из копилок, одна копилка содержит N монет.

2: В момент заполнения одной из копилок, в другой копилке будет ровно K монет.

Данная задача идеально ложится на теорему Бернулли. Воспользуемся следующей формулой:



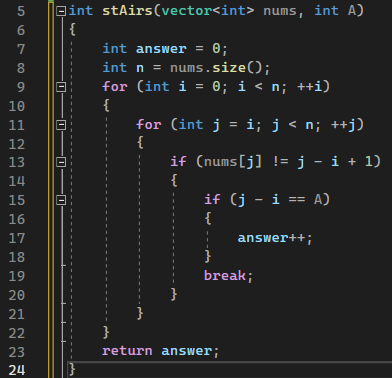
**Ответ:**

Вероятность того, что в момент заполнения одной из копилок, в другой будет ровно K монет высчитывается по следующей формуле:

# Задание 4

**Задание4:**

Дениса попросили написать функцию на языке программирования С++, которая будет считать количество подмассивов вида [1, 2, ..., А] для заданного массива и значения А. Переутомившись к концу рабочего дня, Денис написал следующий код:



Определите асимптотическую сложность решения Дениса и найдите логические дефекты в его программе. Реализуйте исправленное решение и определите его асимптотическую сложность.

**Решение:**

На асимтотическую сложность решения в нашем случае влияют только циклы for. Функция nums.size() имеет сложность O(1)*.*Так как у нас используется вложенный цикл, то сложность будет O(.

Далее рассмотрим логические ошибки.

1. Условие в 13 строке неверно, т.к., как я понял,оно должно проверяет является ли наше значение символом А, однако происходит сравние с какой-то разницей индексов, а не необходимым нам условием
2. Условие в 16 строке проверяет разность индексов, что нам не нужно, т.к. это не гарантирует наличие подмассива, который бы содержал значение А

Попробуем исправить и оптимизировать решение. Представлю полную логику решения, которое использовал.

Для начала проанализируем можно ли обойтись вообще без циклов, а только при помощи формулы. Типичная формула для подсчета подмассивов не подойдет, так как она не учитывает значение А, которое должно содержаться в каждом подмассиве.

Допустим массив состоит из 5 элементов [1,2,3,4,5]. Все возможные варианты:

1. 5
2. 1-5
3. 2-5
4. 3-5
5. 4-5
6. 1-2-5
7. 1-3-5
8. 1-4-5
9. 2-3-5
10. 2-4-5
11. 3-4-5
12. 1-2-3-5
13. 1-2-4-5
14. 1-3-4-5
15. 2-3-4-5
16. 1-2-3-4-5

Всего существует 16 подмассивов. Попробуем найти паттерн.

* 1 – сам элемент А
* 2-4 – все остальные элементы, кроме А
* 6-11 – все пары элементов, кроме А
* 12-15 – все тройки, кроме А
* 16 – сам подмассив

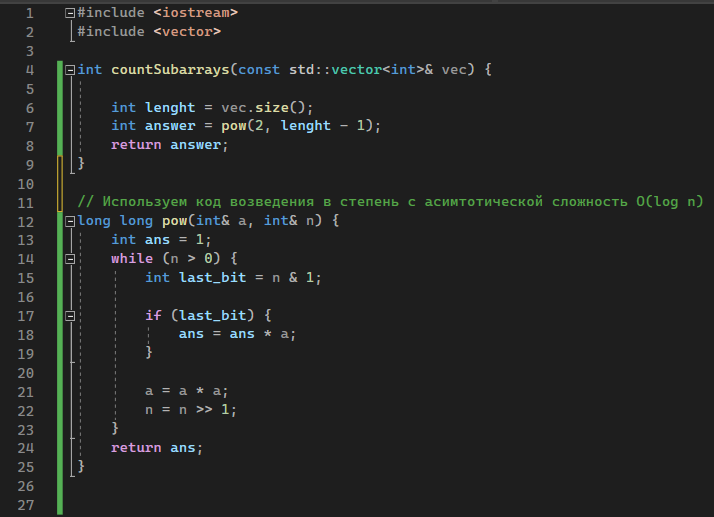
Пока что подходит формула . Проанализируем также для 4-х [1,2,3,4]

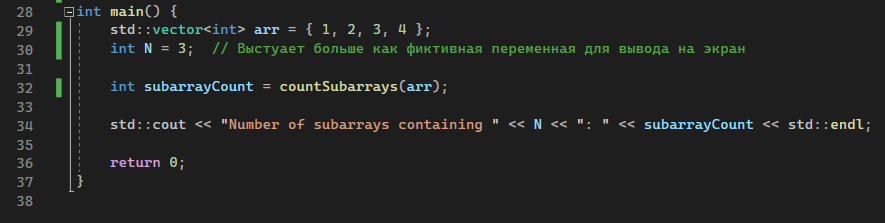
1. 4
2. 1-4
3. 2-4
4. 3-4
5. 1-2-4
6. 1-3-4
7. 2-3-4
8. 1-2-3-4

Пока что подходит формула . Проанализируем также для 6-ти: [1,2,3,4,5,6]

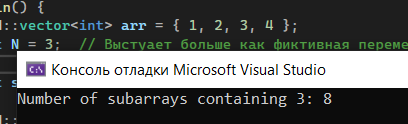
1. 6
2. 1-6
3. 2-6
4. 3-6
5. 4-6
6. 5-6
7. 1-2-6
8. 1-3-6
9. 1-4-6
10. 1-5-6
11. 2-3-6
12. 2-4-6
13. 2-5-6
14. 3-4-6
15. 3-5-6
16. 4-5-6
17. 1-2-3-6
18. 1-2-4-6
19. 1-2-5-6
20. 1-3-4-6
21. 1-3-5-6
22. 1-4-5-6
23. 2-3-4-6
24. 2-3-5-6
25. 2-4-5-6
26. 3-4-5-6
27. 1-2-3-4-6
28. 1-2-3-5-6
29. 1-2-4-5-6
30. 1-3-4-5-6
31. 2-3-4-5-6
32. 1-2-3-4-5-6

Также значния для массивов с 1 и 2-мя элементами все работает корректно. Остановимся на этой формуле. Код будет выглядеть следующим образом

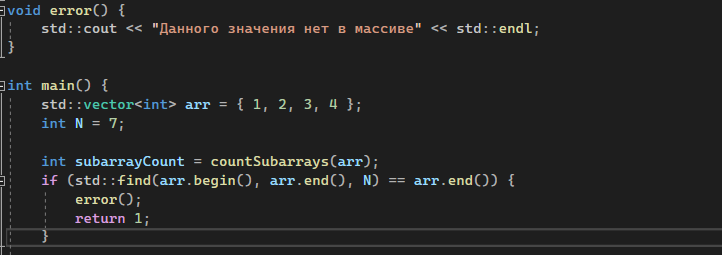




Вывод для 4-х элементов



Асимтотическую сложность алгоритма для нахождения количества подмассивов массива, при условии, что каждый подмассив должен включать в себя элемент А, будет равна **O(log n)**, что намного меньше исходной O(. Однако, если мы добавим сюда проверку на наличие элемента в векторе, то сложность возрастет до **O(**.



Также код будет продублирован в сообщении в архивe с названием «task 4».

# Задание 5

Задание5:

Диме дали следующую задачу. Дана кусочно-линейная функция, заданная целочисленными точками (x\_i, y\_i). Чтобы получить график функции необходимо соединить соседние по х координате точки. Необходимо определить, можно ли сдвинуть оси параллельным переносом (т.е x’ = x + C1; y’ = y + C2) так, чтобы эта заданная ломаная стала четной функцией в новой системе координат. Для этого требуется реализовать функцию со следующей сигнатурой:

bool solve(const vector<pair<int,int>> & points)

На вход подается массив попарно различных точек. Необходимо вернуть ответ на вопрос о существовании описанного преобразования.

Дополнительно определите асимптотическую сложность реализованного решения.

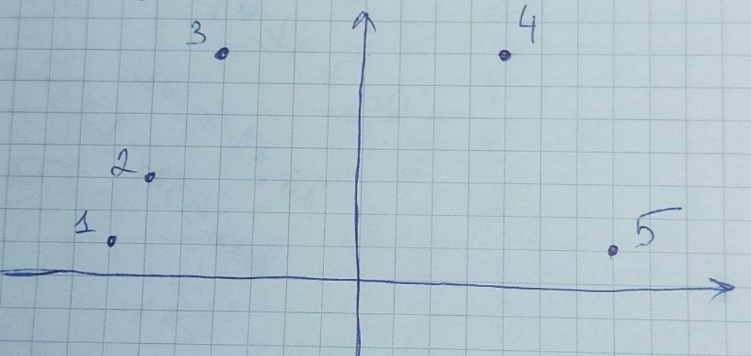
**Решение:**

Так как функция задана у нас линейно-кусочно, то нам будет достаточно проверить наличие близнецов у каждой точки, за одной маленькой оговоркой, которая будет сказана далее.

**Алгоритм решения задачи:**

1. Добавление точек в вектор
2. Сортировка вектора относительно Х
3. Проверка на то, лежат ли соседние точки на одной прямой или нет.
4. Нахождение центра функции и смещение всех точек относительно Х
5. Проверка каждой точки на наличие точки-близнеца, кроме центральной.

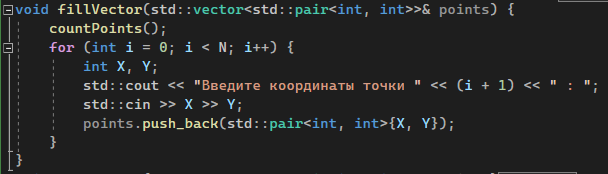
**Примечание к пункту 3:**



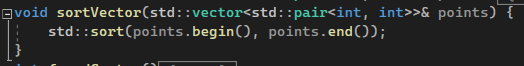
Данная кусочно-линейная функция является симметричной даже при условии, что точка 2 не имеет точку-близнеца, т.к. она лежит на прямой (1,3), поэтому для грамотной работы нашей программы необходимо данную точку удалить.

**Реализация алгоритма в коде:**

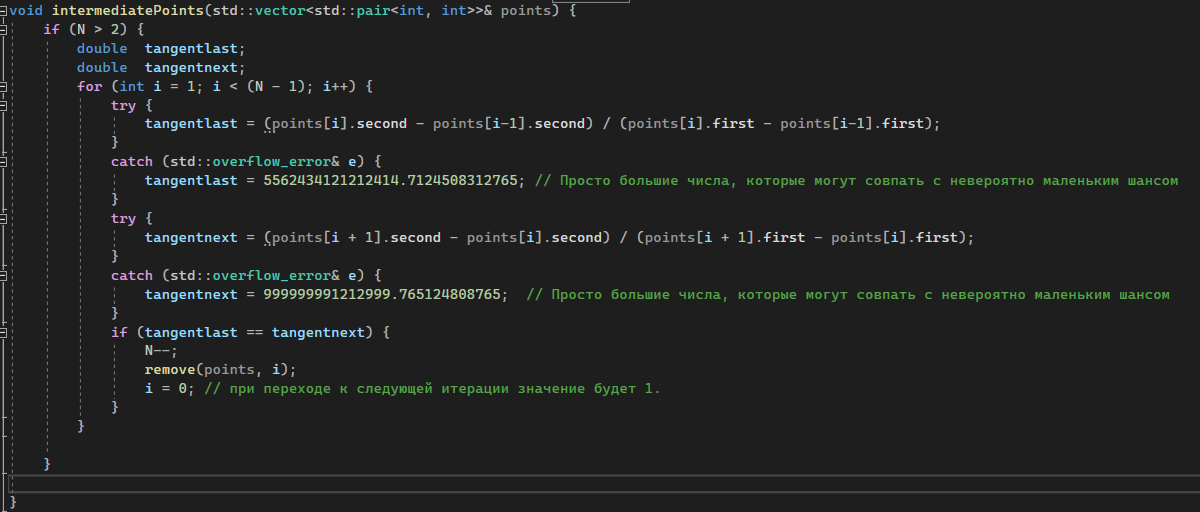
1. Добавление точек в вектор



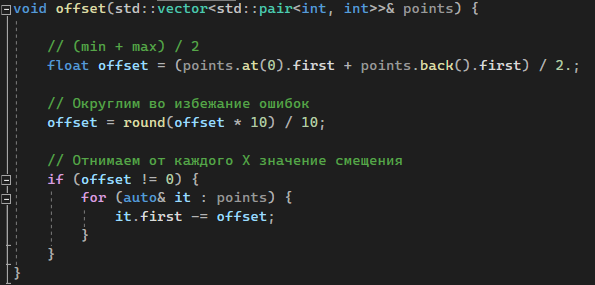
2. Сортировка вектора относительно Х



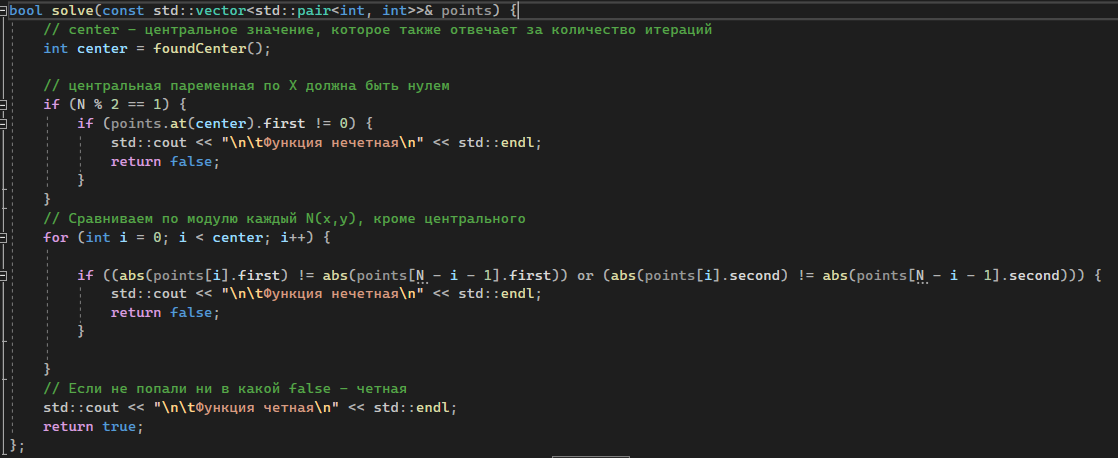
3. Проверка на то, лежат ли соседние точки на одной прямой или нет.



4. Нахождение центра функции и смещение всех точек относительно Х



5. Проверка каждой точки на наличие точки-близнеца, кроме центральной.

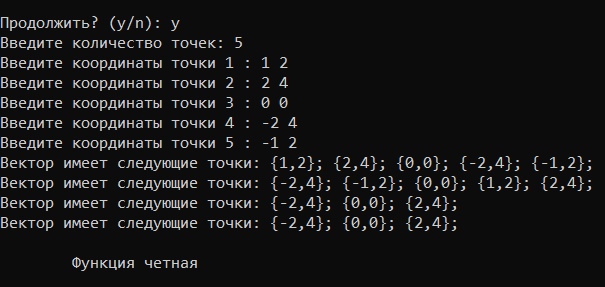


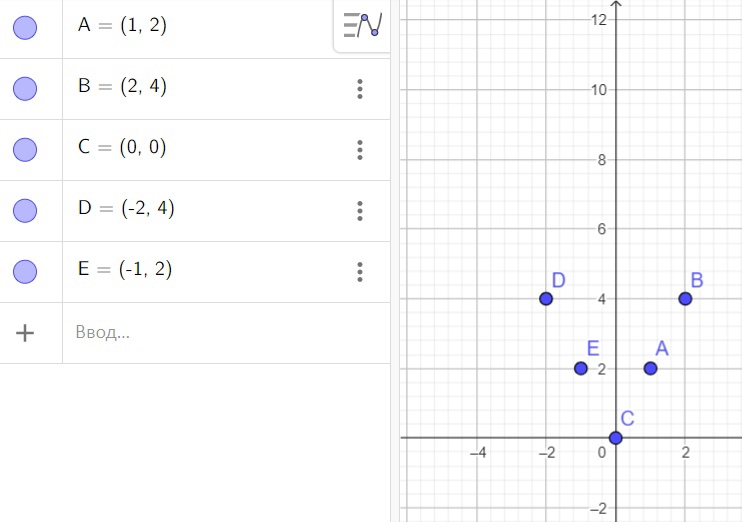
**Асимптотическая сложность решения:**

Общая асимптотическая сложность кода будет зависеть от наиболее сложной функции, и в данном случае это функция sortVector с сложностью **O(N log N).**

**Примеры работы программы:**

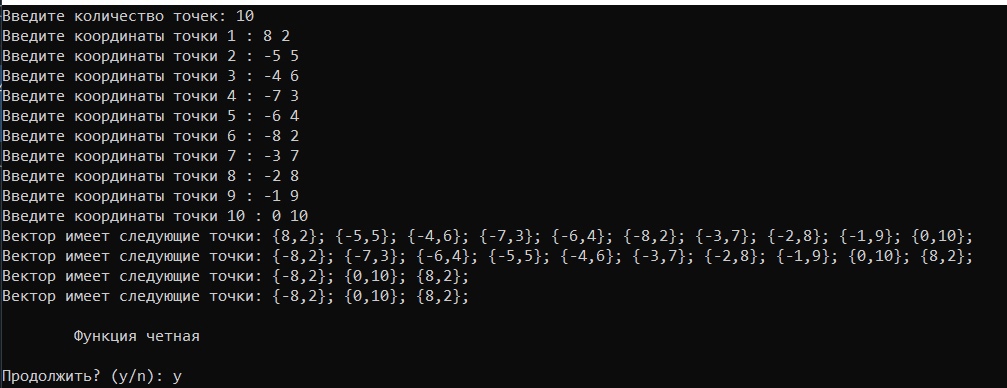
**Первая функция**

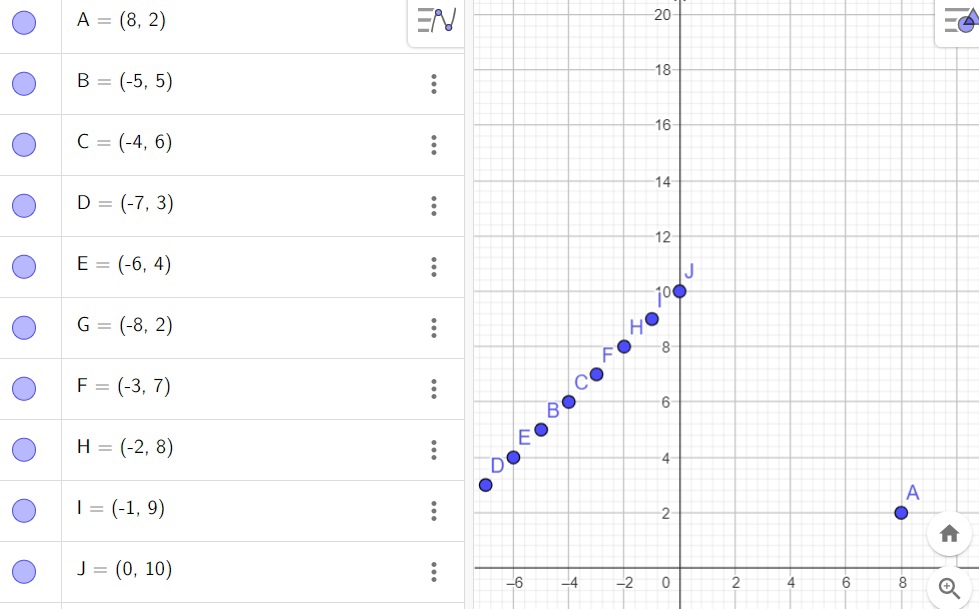
****



Точки Е и А – лишние.

**Вторая функция**





Точки – Е,B,C,F,H,I – лишние.

Также код будет продублирован в сообщении в архивe с названием «task 5».