

ГУАП

КАФЕДРА № 53

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_  
ПРЕПОДАВАТЕЛЬ

старший преподаватель  
должность, уч. степень, звание

подпись, дата

Ушаков В.А.  
инициалы, фамилия

## ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7

### Оценка сложности алгоритмов

Вариант 3

по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ(А)

СТУДЕНТ ГР. № \_\_\_\_\_ 5138

подпись, дата

Воробьев В.А.  
инициалы, фамилия

Санкт-Петербург 2022

**Задание:** реализовать алгоритм на языке C/C++, выполняющий поставленную задачу. Глобальные параметры использовать запрещено; допустимо использование дополнительных функций. Разработанный алгоритм должен быть реализован в виде цельной программной функции (или нескольких функций) так, чтобы мог быть многократно применен с различными исходными данными и при этом не включал команды, не относящиеся к решаемой задаче, например, ввод и вывод исходных данных на консоль или в файл. Произвести теоретическую оценку количества используемых операций разработанного алгоритма. Произвести экспериментальную проверку времени работы разработанного алгоритма, определив его класс сложности для среднего случая. Измерить среднее время для нескольких тестов повторений при различных размерностях входных данных.

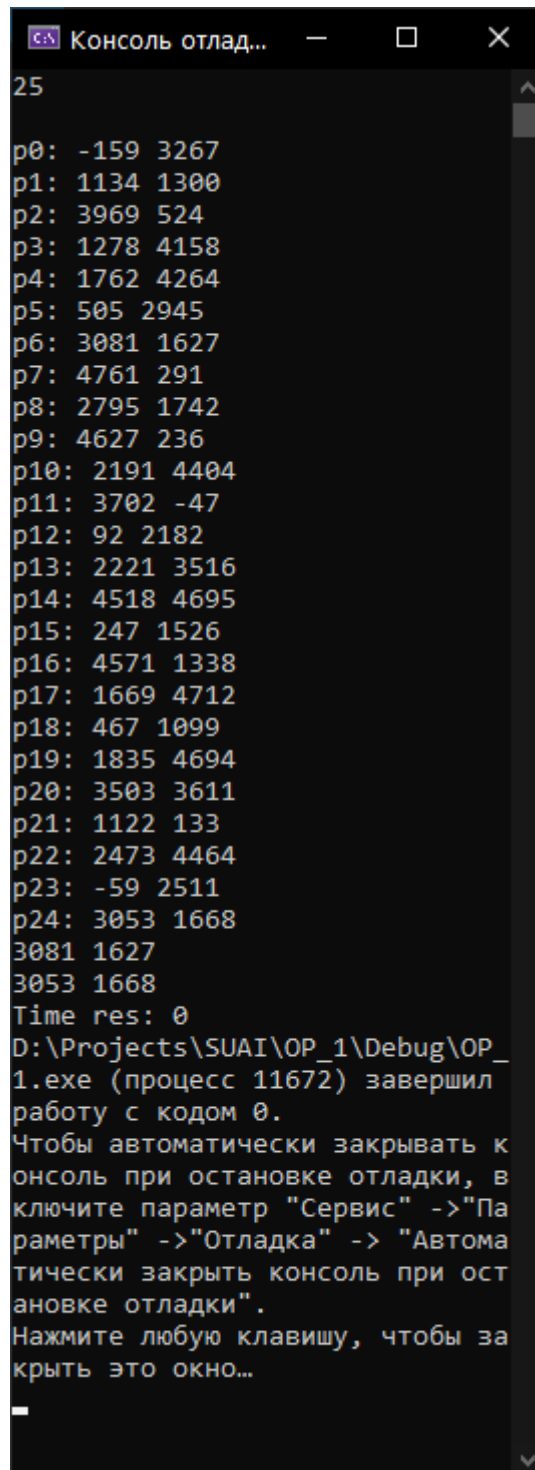
3	Алгоритм нахождения ближайшей пары двумерных точек $P_i, P_j$ (с наименьшим евклидовым расстоянием) в множестве точек размерностью $N$ .		
---	--	--	--

Рисунок 1 - вариант задания

### Выполнение задания:

```
1  #include <iostream>
2  #include <math.h>
3  #include <ctime>
4  #include <cstdlib>
5
6  using namespace std;
7
8  struct Point { int x, y; };
9
10 unsigned long sqrDistance(Point p1, Point p2) {
11     return pow(p1.x - p2.x, 2) + pow(p1.y - p2.y, 2);
12 }
13
14 Point* minDistanceBtw(Point* points, int n) {
15     unsigned long sqr_dist = 0 - 1;
16     unsigned long now_dist;
17     int p1_index = 0, p2_index = 0;
18     for (int i = 0; i < n - 1; i++) {
19         for (int j = i + 1; j < n; j++) {
20             now_dist = sqrDistance(points[i], points[j]);
21             if (now_dist < sqr_dist) {
22                 sqr_dist = now_dist;
23                 p1_index = i;
24                 p2_index = j;
25             }
26         }
27     }
28     return new Point[]{ points[p1_index], points[p2_index] };
29 }
30
31 void main()
32 {
33     int n;
34     cin >> n;
35     Point* points = new Point[n];
36     for (int i = 0; i < n; i++) {
37         int x, y;
38         x = rand() % 5000 - 200;
39         y = rand() % 5000 - 200;
40         cout << "\n" << "p" << i << ": "
41             << x << " " << y;
42         points[i] = Point{ x, y };
43     }
44
45     auto startTct = clock();
46     Point* res = minDistanceBtw(points, n);
47     auto end = clock();
48     cout << "\n" << res[0].x << " " << res[0].y << " \n"
49         << res[1].x << " " << res[1].y;
50     cout << "\nTime res: " << (end - startTct);
51     delete[] res;
52     delete[] points;
53 }
```

Рисунок 2 – код



```
25
r0: -159 3267
r1: 1134 1300
r2: 3969 524
r3: 1278 4158
r4: 1762 4264
r5: 505 2945
r6: 3081 1627
r7: 4761 291
r8: 2795 1742
r9: 4627 236
r10: 2191 4404
r11: 3702 -47
r12: 92 2182
r13: 2221 3516
r14: 4518 4695
r15: 247 1526
r16: 4571 1338
r17: 1669 4712
r18: 467 1099
r19: 1835 4694
r20: 3503 3611
r21: 1122 133
r22: 2473 4464
r23: -59 2511
r24: 3053 1668
3081 1627
3053 1668
Time res: 0
D:\Projects\SUAI\OP_1\Debug\OP_
1.exe (процесс 11672) завершил
работу с кодом 0.
Чтобы автоматически закрывать к
онсоль при остановке отладки, в
ключите параметр "Сервис" ->"Па
раметры" ->"Отладка" -> "Автома
тически закрыть консоль при ост
ановке отладки".
Нажмите любую клавишу, чтобы за
крыть это окно...
```

Рисунок 3 - ввод и вывод



Рисунок 4 - Блок-схема

#### Анализ сложности алгоритма:

Сделаем замеры скорости выполнения для  $N = 80; 150; 250; 500; 1000; 2500$ .

Таблица 1 – измерения  $tacts(N)$

N	80	150	250	500	1000	2500	5000
tacts	1	2	4	16	58	380	1480

Построим график и оценим сложность реализованного алгоритма.



Рисунок 5 – такты от числа входных данных

**Вывод:** текущий алгоритм имеет временную сложность  $O(N^2)$ , это понятно из графика (рис.5), которые стремительно растет с увеличением количества входных данных. В ходе выполнения задания был выполнен алгоритм, на основе которого мы научились измерять временную сложность алгоритма. Также освоили метод «грубой силы».