

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №4
дисциплины «Алгоритмизация»
Вариант 7

Выполнил:
Горбунов Данила Евгеньевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. технических
наук, доцент кафедры
инфокоммуникаций

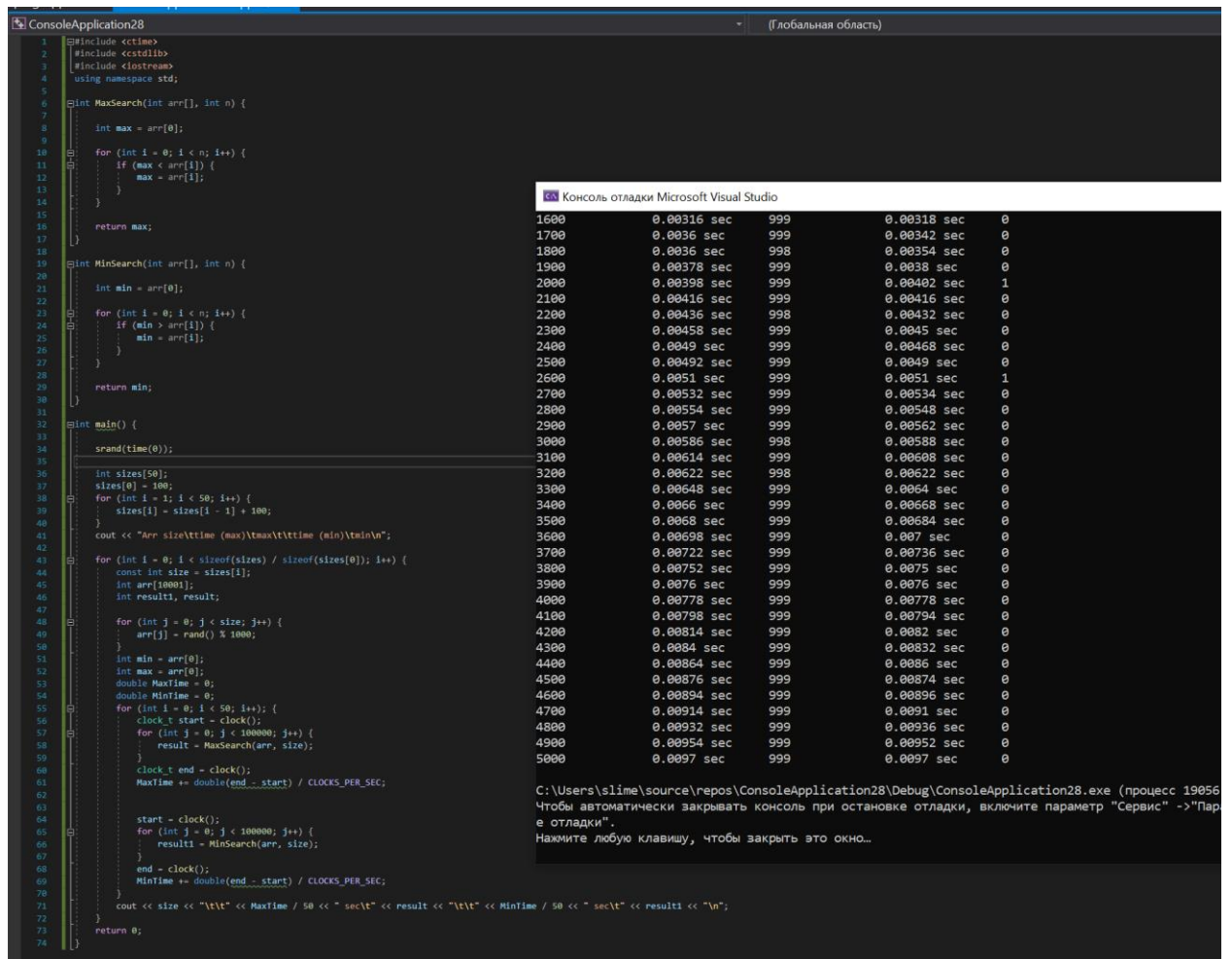
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Ход работы

1. Написал программу, которая подсчитывает время, затрачиваемое на выполнение алгоритма линейного поиска, предусмотрел варианты среднего (искомый элемент находится где-то в середине массива) и худшего (искомый элемент не найден) случая. (Рисунок 1)



The screenshot displays the Visual Studio IDE with a C++ project named 'ConsoleApplication28'. The source code is visible on the left, and the debug console on the right shows the program's execution results.

Source Code (ConsoleApplication28.cpp):

```
1 #include <iostream>
2 #include <ctime>
3 #include <cstdlib>
4 using namespace std;
5
6 int MaxSearch(int arr[], int n) {
7     int max = arr[0];
8     for (int i = 0; i < n; i++) {
9         if (max < arr[i]) {
10             max = arr[i];
11         }
12     }
13     return max;
14 }
15
16 int MinSearch(int arr[], int n) {
17     int min = arr[0];
18     for (int i = 0; i < n; i++) {
19         if (min > arr[i]) {
20             min = arr[i];
21         }
22     }
23     return min;
24 }
25
26 int main() {
27     srand(time(0));
28     int sizes[50];
29     sizes[0] = 100;
30     for (int i = 1; i < 50; i++) {
31         sizes[i] = sizes[i - 1] + 100;
32     }
33     cout << "Arr size\ttime (max)\ttime (min)\t\n";
34     for (int i = 0; i < sizeof(sizes) / sizeof(sizes[0]); i++) {
35         const int size = sizes[i];
36         int arr[10001];
37         int result1, result;
38         for (int j = 0; j < size; j++) {
39             arr[j] = rand() % 1000;
40         }
41         int min = arr[0];
42         int max = arr[0];
43         double MaxTime = 0;
44         double MinTime = 0;
45         for (int i = 0; i < 50; i++) {
46             clock_t start = clock();
47             for (int j = 0; j < 100000; j++) {
48                 result = MaxSearch(arr, size);
49             }
50             clock_t end = clock();
51             MaxTime += double(end - start) / CLOCKS_PER_SEC;
52
53             start = clock();
54             for (int j = 0; j < 100000; j++) {
55                 result1 = MinSearch(arr, size);
56             }
57             end = clock();
58             MinTime += double(end - start) / CLOCKS_PER_SEC;
59         }
60         cout << size << "\t\t" << MaxTime / 50 << " sec\t" << result << "\t\t" << MinTime / 50 << " sec\t" << result1 << "\n";
61     }
62     return 0;
63 }
```

Debug Console Output:

Size	MaxTime (sec)	MaxSearch Result	MinTime (sec)	MinSearch Result
100	0.00316	999	0.00318	0
1700	0.0036	999	0.00342	0
1800	0.0036	998	0.00354	0
1900	0.00378	999	0.0038	0
2000	0.00398	999	0.00402	1
2100	0.00416	999	0.00416	0
2200	0.00436	998	0.00432	0
2300	0.00458	999	0.0045	0
2400	0.0049	999	0.00468	0
2500	0.00492	999	0.0049	0
2600	0.0051	999	0.0051	1
2700	0.00532	999	0.00534	0
2800	0.00554	999	0.00548	0
2900	0.0057	999	0.00562	0
3000	0.00586	998	0.00588	0
3100	0.00614	999	0.00608	0
3200	0.00622	998	0.00622	0
3300	0.00648	999	0.0064	0
3400	0.0066	999	0.00668	0
3500	0.0068	999	0.00684	0
3600	0.00698	999	0.007	0
3700	0.00722	999	0.00736	0
3800	0.00752	999	0.0075	0
3900	0.0076	999	0.0076	0
4000	0.00778	999	0.00778	0
4100	0.00798	999	0.00794	0
4200	0.00814	999	0.0082	0
4300	0.0084	999	0.00832	0
4400	0.00864	999	0.0086	0
4500	0.00876	999	0.00874	0
4600	0.00894	999	0.00896	0
4700	0.00914	999	0.0091	0
4800	0.00932	999	0.00936	0
4900	0.00954	999	0.00952	0
5000	0.0097	999	0.0097	0

Path: C:\Users\slime\source\repos\ConsoleApplication28\Debug\ConsoleApplication28.exe (процесс 19056)
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" -> "Параметры отладки".
Нажмите любую клавишу, чтобы закрыть это окно...

Рисунок 1. Работа программы

```
#include <iostream>
#include <ctime>
#include <cstdlib>
```

```
using namespace std;
```

```
int MaxSearch(int arr[], int n) {
```

```
    int max = arr[0];
```

```
    for (int i = 0; i < n; i++) {
        if (max < arr[i]) {
            max = arr[i];
        }
    }
```

```
    return max;
```

```
}
```

```

int MinSearch(int arr[], int n) {

    int min = arr[0];

    for (int i = 0; i < n; i++) {
        if (min > arr[i]) {
            min = arr[i];
        }
    }

    return min;
}

int main() {

    srand(time(0));

    int sizes[50];
    sizes[0] = 100;
    for (int i = 1; i < 50; i++) {
        sizes[i] = sizes[i - 1] + 100;
    }
    cout << "Arr size\ttime (max)\tmax\t\ttime (min)\tmin\n";

    for (int i = 0; i < sizeof(sizes) / sizeof(sizes[0]); i++) {
        const int size = sizes[i];
        int arr[10001];
        int result1, result;

        for (int j = 0; j < size; j++) {
            arr[j] = rand() % 1000;
        }
        int min = arr[0];
        int max = arr[0];
        double MaxTime = 0;
        double MinTime = 0;
        for (int i = 0; i < 50; i++) {
            clock_t start = clock();
            for (int j = 0; j < 100000; j++) {
                result = MaxSearch(arr, size);
            }
            clock_t end = clock();
            MaxTime += double(end - start) / CLOCKS_PER_SEC;

            start = clock();
            for (int j = 0; j < 100000; j++) {
                result1 = MinSearch(arr, size);
            }
            end = clock();
            MinTime += double(end - start) / CLOCKS_PER_SEC;
        }
        cout << size << "\t\t" << MaxTime / 50 << " sec\t" << result << "\t\t" << MinTime / 50 << " sec\t" << result1 <<
        "\n";
    }
    return 0;
}

```

Таблица 1. Время работы алгоритмов поиска минимума и максимума

Размер массива (n)	Время для поиска максимума (сек)	Время для поиска минимума (сек)
100	0,0000033000	0,0000028000
200	0,0000047000	0,0000062000

300	0,0000073000	0,0000080000
400	0,0000111000	0,0000127000
500	0,0000114000	0,0000112000
600	0,0000152000	0,0000152000
700	0,0000196000	0,0000186000
800	0,0000225000	0,0000216000
900	0,0000223000	0,0000200000
1000	0,0000272000	0,0000250000
1100	0,0000304000	0,0000247000
1200	0,0000275000	0,0000268000
1300	0,0000305000	0,0000287000
1400	0,0000317000	0,0000315000
1500	0,0000343000	0,0000360000
1600	0,0000352000	0,0000351000
1700	0,0000376000	0,0000385000
1800	0,0000401000	0,0000398000
1900	0,0000425000	0,0000426000
2000	0,0000444000	0,0000450000
2100	0,0000475000	0,0000484000
2200	0,0000488000	0,0000496000
2300	0,0000514000	0,0000516000
2400	0,0000530000	0,0000523000
2500	0,0000572000	0,0000570000
2600	0,0000589000	0,0000598000
2700	0,0000601000	0,0000604000
2800	0,0000636000	0,0000620000
2900	0,0000640000	0,0000635000
3000	0,0000691000	0,0000671000
3100	0,0000691000	0,0000696000
3200	0,0000705000	0,0000710000
3300	0,0000759000	0,0000728000
3400	0,0000760000	0,0000757000
3500	0,0000797000	0,0000766000
3600	0,0000801000	0,0000793000
3700	0,0000845000	0,0000821000
3800	0,0000849000	0,0000837000
3900	0,0000868000	0,0000885000
4000	0,0000878000	0,0000893000
4100	0,0000926000	0,0000892000
4200	0,0000933000	0,0000930000
4300	0,0000950000	0,0000956000
4400	0,0000991000	0,0000973000
4500	0,0000981000	0,0000997000
4600	0,0001047000	0,0001016000
4700	0,0001034000	0,0001054000
4800	0,0001071000	0,0001104000
4900	0,0001097000	0,0001084000

5000	0,0001124000	0,0001100000
------	--------------	--------------

2. Перенес данные по алгоритму поиска максимума в таблицу Excel и произвел необходимые расчеты.

L18																
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	n		time*10000	time	n*n	t ²	time*n	Y								
2	100		0,033	0,0000033000	10000	0,0000000001089	0,00033	0,000002230485746			207850000	45500		4,63872000000000		
3	200		0,047	0,0000047000	40000	0,0000000002209	0,00094	0,000004462277629			45500	50		0,0010154000		
4	300		0,073	0,0000073000	90000	0,0000000005329	0,00219	0,000006694069512								
5	400		0,111	0,0000111000	160000	0,0000000012321	0,00444	0,000008925861396			6,008E-09	-5,5E-06	a=	2,23179E-08		
6	500		0,114	0,0000114000	250000	0,0000000012996	0,0057	0,000011157653279			-5,47E-06	0,02498	b=	-1,30614E-09		
7	600		0,152	0,0000152000	360000	0,0000000023104	0,00912	0,000013389445162								
8	700		0,196	0,0000196000	490000	0,0000000038416	0,01372	0,000015621237045								
9	800		0,225	0,0000225000	640000	0,0000000050625	0,018	0,000017853028928								
10	900		0,223	0,0000223000	810000	0,0000000049729	0,02007	0,000020084820812								
11	1000		0,272	0,0000272000	1000000	0,0000000073984	0,0272	0,000022316612695								
12	1100		0,304	0,0000304000	1210000	0,0000000092416	0,03344	0,000024548404578								
13	1200		0,275	0,0000275000	1440000	0,0000000075625	0,033	0,000026780196461								
14	1300		0,305	0,0000305000	1690000	0,0000000093025	0,03965	0,000029011988344								
15	1400		0,317	0,0000317000	1960000	0,00000000100489	0,04438	0,000031243780228								
16	1500		0,343	0,0000343000	2250000	0,00000000117649	0,05145	0,000033475572111								
17	1600		0,352	0,0000352000	2560000	0,00000000123904	0,05632	0,000035707363994								
18	1700		0,376	0,0000376000	2890000	0,00000000141376	0,06392	0,000037939155877								
19	1800		0,401	0,0000401000	3240000	0,00000000160801	0,07218	0,000040170947761								
20	1900		0,425	0,0000425000	3610000	0,00000000180625	0,08075	0,000042402739644								
21	2000		0,444	0,0000444000	4000000	0,00000000197136	0,0888	0,000044634531527								
22	2100		0,475	0,0000475000	4410000	0,00000000225625	0,09975	0,000046866323410								
23	2200		0,488	0,0000488000	4840000	0,00000000238144	0,10736	0,000049098115293								
24	2300		0,514	0,0000514000	5290000	0,00000000264196	0,11822	0,000051329907177								
25	2400		0,53	0,0000530000	5760000	0,00000000280900	0,1272	0,000053561699080								
26	2500		0,572	0,0000572000	6250000	0,00000000327184	0,143	0,000055793490943								
27	2600		0,589	0,0000589000	6760000	0,00000000346921	0,15314	0,000058025282826								
28	2700		0,601	0,0000601000	7290000	0,00000000361201	0,16227	0,000060257074709								
29	2800		0,636	0,0000636000	7840000	0,00000000404496	0,17808	0,000062488866593								
30	2900		0,64	0,0000640000	8410000	0,00000000409600	0,1856	0,000064720658476								
31	3000		0,691	0,0000691000	9000000	0,00000000477481	0,2073	0,000066952450359								
32	3100		0,691	0,0000691000	9610000	0,00000000477481	0,21421	0,000069184242242								
33	3200		0,705	0,0000705000	10240000	0,00000000497025	0,2256	0,000071416034125								
34	3300		0,759	0,0000759000	10890000	0,00000000576081	0,25047	0,000073647826009								
35	3400		0,76	0,0000760000	11560000	0,00000000577600	0,2584	0,000075879617892								
36	3500		0,797	0,0000797000	12250000	0,00000000635209	0,27895	0,000078111409775								
37	3600		0,801	0,0000801000	12960000	0,00000000641601	0,28836	0,000080343201658								
38	3700		0,845	0,0000845000	13690000	0,00000000714025	0,31265	0,000082574993541								
39	3800		0,849	0,0000849000	14440000	0,00000000720801	0,32262	0,000084806785425								
40	3900		0,888	0,0000888000	15210000	0,00000000753424	0,33852	0,000087038577308								
41	4000		0,878	0,0000878000	16000000	0,00000000770884	0,3512	0,000089270369191								
42	4100		0,926	0,0000926000	16810000	0,00000000857476	0,37966	0,000091502161074								
43	4200		0,933	0,0000933000	17640000	0,00000000870489	0,39186	0,000093733952957								
44	4300		0,95	0,0000950000	18490000	0,00000000902500	0,4085	0,000095965744841								
45	4400		0,991	0,0000991000	19360000	0,00000000982081	0,43604	0,000098197536724								
46	4500		0,981	0,0000981000	20250000	0,00000000982361	0,44145	0,000100429328607								
47	4600		1,047	0,0001047000	21160000	0,00000001096209	0,48162	0,000102661120490								
48	4700		1,034	0,0001034000	22090000	0,00000001069156	0,48598	0,000104892912373								
49	4800		1,071	0,0001071000	23040000	0,00000001147041	0,51408	0,000107124704257								
50	4900		1,097	0,0001097000	24010000	0,00000001203409	0,53753	0,000109356496140								
51	5000		1,124	0,0001124000	25000000	0,00000001263376	0,562	0,000111588288023								
52	сумма	45500	10,154	0,0010154000	207850000	0,00000010354098	4,63872									
53																

Рисунок 2. Расчет линейной зависимости

3. Построил график линейной зависимости времени выполнения алгоритма поиска максимума в массиве от размера массива.

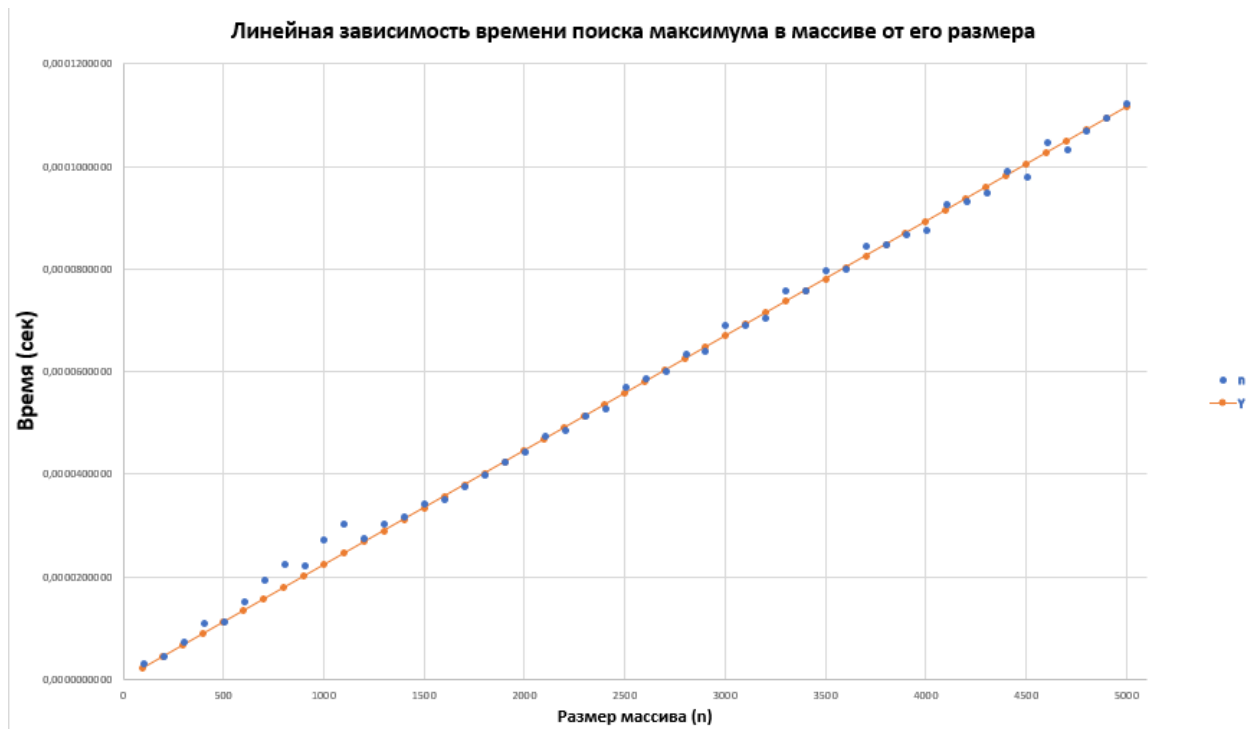


Рисунок 3. График для поиска максимума

4. Произвел аналогичные расчеты для получения необходимой функции.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	n	time*1000	time	n*n	t*t	time*n	Y							
2	100	0,028	0,0000028000	10000	0,000000000000784	0,00028	0,000002214110968				207850000	45500		4,617990000000000
3	200	0,062	0,0000062000	40000	0,000000000003844	0,00124	0,000004436072577				45500	50		0,0010106000
4	300	0,08	0,0000080000	90000	0,000000000006400	0,0024	0,000006658034185							
5	400	0,127	0,0000127000	160000	0,000000000016129	0,00508	0,000008879995794				6,00799E-09	-5,5E-06	a=	2,22196E-08
6	500	0,112	0,0000112000	250000	0,000000000012544	0,0056	0,000011101957403				-5,46727E-06	0,02498	b=	-7,85064E-09
7	600	0,152	0,0000152000	360000	0,000000000023104	0,00912	0,000013323919012							
8	700	0,186	0,0000186000	490000	0,000000000034596	0,01302	0,000015545880621							
9	800	0,216	0,0000216000	640000	0,000000000046556	0,01728	0,000017767841230							
10	900	0,2	0,0000200000	810000	0,000000000040000	0,018	0,000019989903839							
11	1000	0,25	0,0000250000	1000000	0,000000000062500	0,025	0,000022211765448							
12	1100	0,247	0,0000247000	1210000	0,000000000061009	0,02717	0,000024433727057							
13	1200	0,268	0,0000268000	1440000	0,000000000071824	0,03216	0,000026655688666							
14	1300	0,287	0,0000287000	1690000	0,000000000082369	0,03731	0,000028877650275							
15	1400	0,315	0,0000315000	1960000	0,000000000099225	0,0441	0,000031099611884							
16	1500	0,36	0,0000360000	2250000	0,000000000129600	0,054	0,000033321573493							
17	1600	0,351	0,0000351000	2560000	0,00000000013201	0,05616	0,000035543535102							
18	1700	0,385	0,0000385000	2890000	0,000000000148225	0,06545	0,000037765496711							
19	1800	0,398	0,0000398000	3240000	0,000000000158404	0,07164	0,000039987458320							
20	1900	0,426	0,0000426000	3610000	0,000000000181476	0,08094	0,000042209419929							
21	2000	0,45	0,0000450000	4000000	0,000000000202500	0,09	0,000044431381537							
22	2100	0,484	0,0000484000	4410000	0,000000000234256	0,10164	0,000046653343146							
23	2200	0,496	0,0000496000	4840000	0,000000000246016	0,10912	0,000048875304755							
24	2300	0,516	0,0000516000	5290000	0,000000000266256	0,11868	0,000051097266364							
25	2400	0,523	0,0000523000	5760000	0,000000000273529	0,12552	0,000053319227973							
26	2500	0,57	0,0000570000	6250000	0,000000000324900	0,1425	0,000055541189582							
27	2600	0,598	0,0000598000	6760000	0,000000000357604	0,15548	0,000057763151191							
28	2700	0,604	0,0000604000	7290000	0,000000000364816	0,16308	0,000059985112800							
29	2800	0,62	0,0000620000	7840000	0,000000000384400	0,1736	0,000062207074409							
30	2900	0,635	0,0000635000	8410000	0,000000000403225	0,18415	0,000064429036018							
31	3000	0,671	0,0000671000	9000000	0,000000000450241	0,2013	0,000066650997627							
32	3100	0,696	0,0000696000	9610000	0,000000000484416	0,21576	0,000068872859236							
33	3200	0,71	0,0000710000	10240000	0,000000000504100	0,2272	0,000071094920845							
34	3300	0,728	0,0000728000	10890000	0,000000000529984	0,24024	0,000073316882454							
35	3400	0,757	0,0000757000	11560000	0,000000000573049	0,25738	0,000075538844063							
36	3500	0,766	0,0000766000	12250000	0,000000000586756	0,2681	0,000077760805672							
37	3600	0,793	0,0000793000	12960000	0,000000000628849	0,28548	0,000079982767280							
38	3700	0,821	0,0000821000	13690000	0,000000000674041	0,30377	0,000082204728889							
39	3800	0,837	0,0000837000	14440000	0,000000000700569	0,31806	0,000084426690498							
40	3900	0,885	0,0000885000	15210000	0,000000000783225	0,34515	0,000086648652107							
41	4000	0,893	0,0000893000	16000000	0,000000000797449	0,3572	0,000088870613716							
42	4100	0,892	0,0000892000	16810000	0,000000000795664	0,36572	0,000091092575325							
43	4200	0,93	0,0000930000	17640000	0,000000000864900	0,3906	0,000093314536934							
44	4300	0,956	0,0000956000	18490000	0,000000000913936	0,41108	0,000095536498543							
45	4400	0,973	0,0000973000	19360000	0,000000000946729	0,42812	0,000097758460152							
46	4500	0,997	0,0000997000	20250000	0,000000000994009	0,44865	0,000099980421761							
47	4600	1,016	0,0001016000	21160000	0,000000001032256	0,46736	0,000102202383370							
48	4700	1,054	0,0001054000	22090000	0,00000000110916	0,49538	0,000104424344979							
49	4800	1,104	0,0001104000	23040000	0,000000001218816	0,52992	0,000106646306588							
50	4900	1,084	0,0001084000	24010000	0,000000001175056	0,53116	0,000108868268197							
51	5000	1,1	0,0001100000	25000000	0,000000001210000	0,55	0,000111090229806							
52	сумма	45500	10,106	0,0010106000	207850000	0,00000010262282	4,61799							

Рисунок 4. Расчет функции линейной зависимости для поиска минимума

5. Построил график линейной зависимости времени выполнения алгоритма поиска минимума в массиве от размера массива.

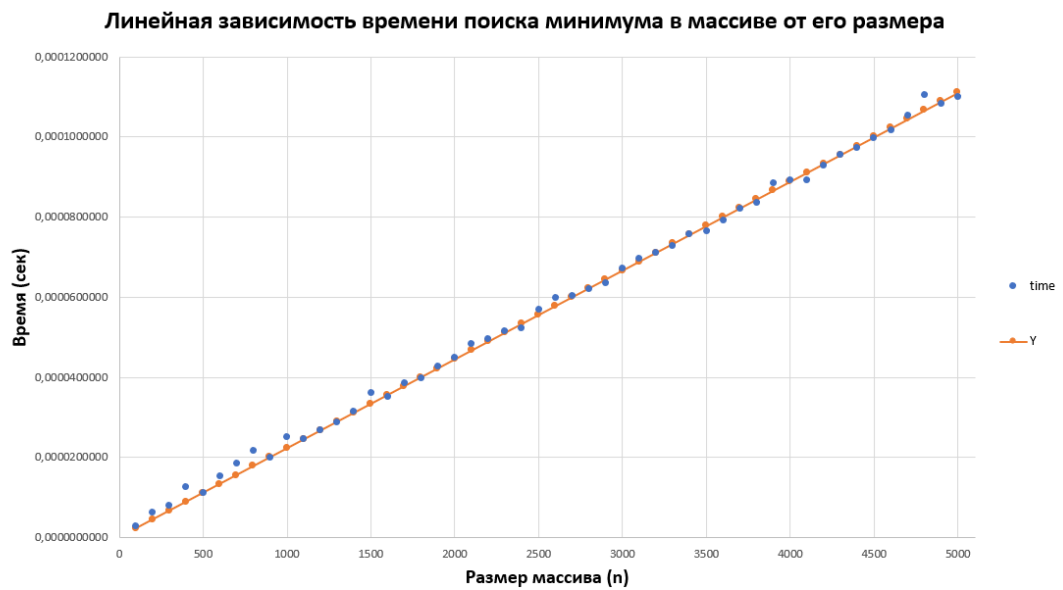


Рисунок 6. График для поиска минимума

6. Рассчитал коэффициенты парной корреляции для поиска максимума ($r = 0,999905146$) и минимума ($r = 0,999873533$).

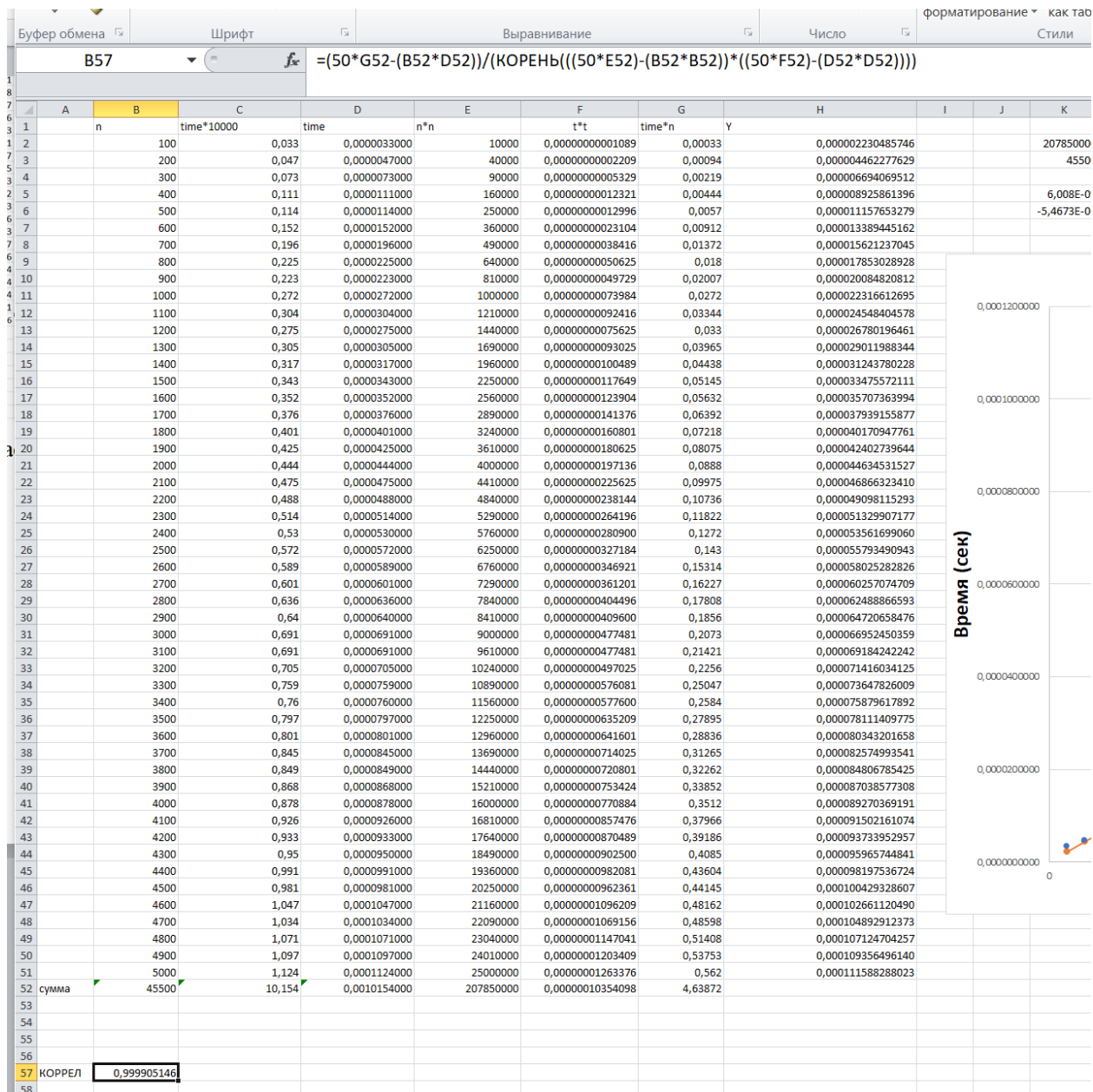


Рисунок 7. Расчет коэффициента парной корреляции для поиска максимума

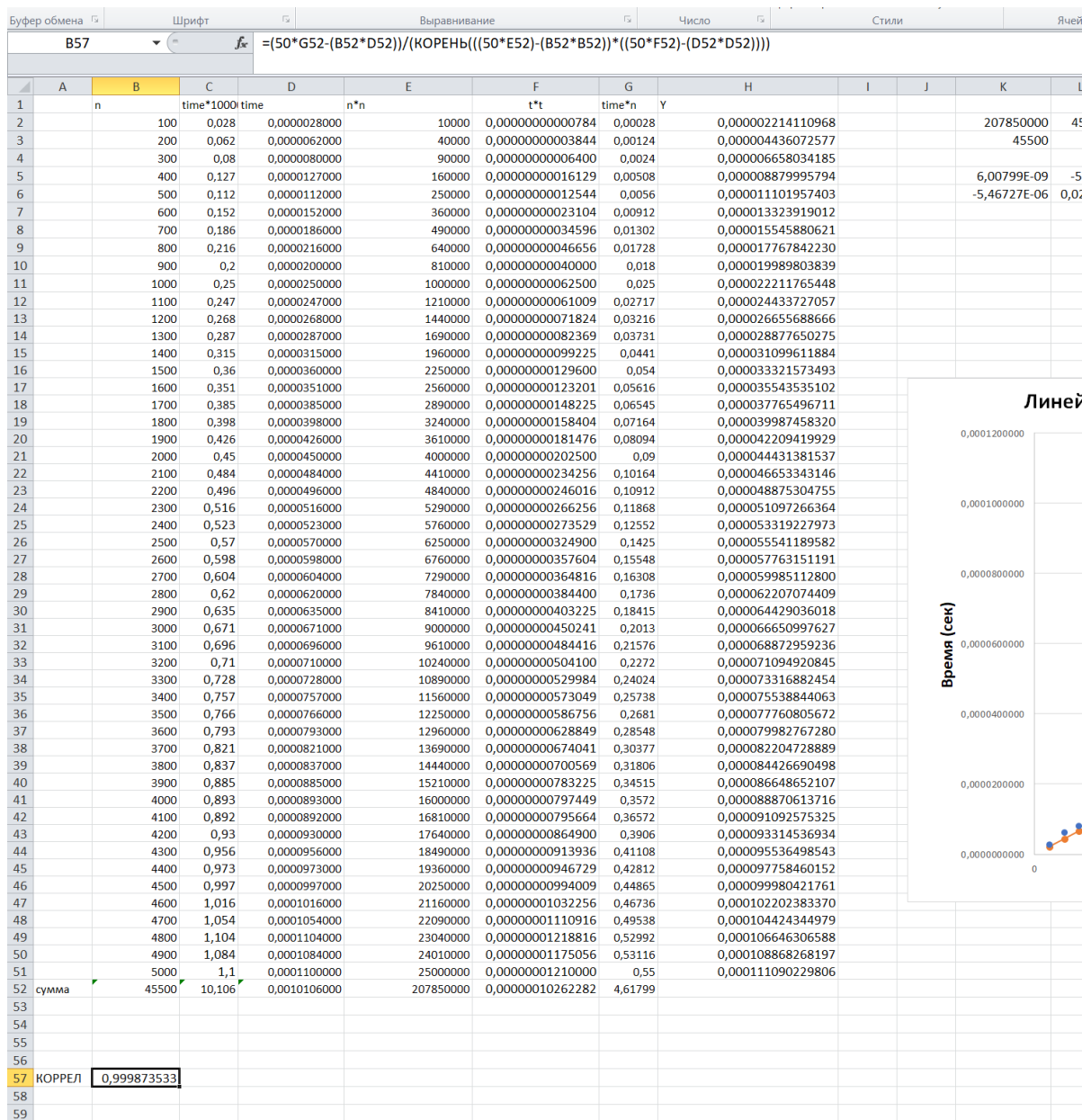


Рисунок 8. Расчет коэффициента парной корреляции для поиска минимума

Вывод: в ходе выполнения лабораторной работы был проведен анализ алгоритмов поиска минимума и максимума в массиве. Поскольку эти алгоритмы подразумевают перебор всех членов массива, можно предположить, что время работы алгоритма напрямую зависит от размера массива, что было подтверждено экспериментальным и статистическим методами. Из полученных результатов, а также из расчета коэффициента парной корреляции, можно сделать вывод о том, что данные алгоритмы действительно линейно зависят от размера массива, в котором производится поиск.