МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Институт цифры

Прикладная математика и информатика

**Программирование**

Отчёт по домашней работе №7

«**Сортировки одномерных массивов**»

Индивидуальный вариант **№2**

Выполнил студент группы ПМИ-231  
**Медведев Владимир Андреевич**

Кемерово-2024

Задание 1

# Программный код

h-файл

#ifndef FUNC\_H\_INCLUDED

#define FUNC\_H\_INCLUDED

void menu(double\*,int n);

void menu2(double\*, int, int&, int&);

void input\_vect(double\*, int);

bool input\_vect\_file(double\*, int);

void input\_vect\_rand(double\*, int, double, double);

void input\_vect\_rule(double\*, int);

void output\_vect(double\*, int);

double Get\_max(double\*, int , int&);

double Get\_min(double\*, int , int&);

void reverse1(double\*, int, int, int);

void bubble\_sort(double \*, int , int, int&,int&);

void SelectSort(double\*, int, int, int&,int&);

void InsertSort (double\*, int , int, int&,int&);

void shell\_sort(double\*, int , int, int&, int&);

#endif // FUNC\_H\_INCLUDED

cpp

#include <iostream>

#include "func.h"

#include <fstream>

#include <stdlib.h>

#include <iomanip>

#include <ctime>

using namespace std;

void menu(double\*a,int n)

{

cout << "[1] Ввод с экрана" << endl;

cout << "[2] Ввод из файла" << endl;

cout << "[3] Заполнение случайными числами" << endl;

cout << "[4] Заполнение массива по правилу n, n-1, n-2,...,3,2,1" << endl;

int kkk;

cout << "Введите позицию из меню: "; cin >> kkk;

switch (kkk)

{

case 1:

input\_vect(a, n);

break;

case 2:

input\_vect\_file(a, n);

break;

case 3:

double p1, p2;

cout << "Введите пределы поиска рандомного числа:";

cin >> p1; cin >> p2; cout << endl;

input\_vect\_rand(a, n, p1, p2);

break;

case 4:

input\_vect\_rule(a, n);

break;

}

}

void input\_vect(double\* a, int n)

{

for (int i = 0; i < n; i++)

{

cin >> a[i];

}

}

bool input\_vect\_file(double\* a, int n)

{

ifstream f1("1.txt");

if (!f1)

{

cout << "Файла для считывания нет" << endl;

return false;

}

double\* c = new double[1];

int kk = 0;

while (f1 >> c[0])

kk++;

delete[]c;

cout << "Количество чисел в файле: " << kk << endl;

if (kk < n)

{

cout << "Чисел в файле не хватает для заполнения всего массива. Чисел в файле " << kk << " а в массиве " << n << " элементов" << endl;

f1.close();

return false;

}

f1.clear();;

f1.seekg(0);

for (int i = 0; i < n; i++)

{

f1 >> a[i];

}

f1.close();

return true;

}

void input\_vect\_rand(double\* a, int n, double p1, double p2)

{

srand(time(NULL));

for (int i = 0; i < n; i++)

{

a[i] = rand() \* (p2 - p1) / RAND\_MAX + p1;

}

}

void input\_vect\_rule(double\* a, int n)

{

for (int i = 0; i < n; i++)

{

a[i]=n-i;

}

}

void output\_vect(double\* a, int n)

{

for (int i = 0; i < n; i++)

{

cout << a[i] << " ";

}

cout << endl;

}

void zdvig(double\* a, int n)

{

double temp;

for (int i=0; i<n; i+=2)

{

temp = a[n-1];

for (int j=n-1; j>=i+1; j--)

{

a[j]=a[j-1];

}

a[i]=temp;

}

}

void menu2(double\*a,int n, int& cravn, int& perest)

{

cout << "[1] Сортировка пузырьком" << endl;

cout << "[2] Сортировка выбором" << endl;

cout << "[3] Сортировка вставками" << endl;

cout << "[4] Сортировка Шелла" << endl;

cravn=0;

perest=0;

int kkk;

cout << "Введите позицию из меню: "; cin >> kkk;

switch (kkk)

{

case 1:

cout << "[1] Сортировка пузырьком по возрастанию" << endl;

cout << "[2] Сортировка пузырьком по убыванию" << endl;

int ppp;

cout << "Введите позицию из меню: "; cin >> ppp;

switch (ppp)

{

case 1:

bubble\_sort(a, n , 1,cravn, perest);

break;

case 2:

bubble\_sort(a, n , -1,cravn, perest);

break;

}

break;

case 2:

cout << "[1] Сортировка выбором по возрастанию" << endl;

cout << "[2] Сортировка выбором по убыванию" << endl;

int ppp1;

cout << "Введите позицию из меню: "; cin >> ppp1;

switch (ppp1)

{

case 1:

SelectSort(a, n , 1,cravn, perest);

break;

case 2:

SelectSort(a, n , -1,cravn, perest);

break;

}

break;

case 3:

cout << "[1] Сортировка вставками по возрастанию" << endl;

cout << "[2] Сортировка вставками по убыванию" << endl;

int pp3;

cout << "Введите позицию из меню: "; cin >> pp3;

switch (pp3)

{

case 1:

InsertSort(a, n , 1,cravn, perest);

break;

case 2:

InsertSort(a, n , -1,cravn, perest);

break;

}

break;

case 4:

cout << "[1] Сортировка Шелла по возрастанию" << endl;

cout << "[2] Сортировка Шелла по убыванию" << endl;

int pp4;

cout << "Введите позицию из меню: "; cin >> pp4;

switch (pp4)

{

case 1:

shell\_sort(a, n , 1,cravn, perest);

break;

case 2:

shell\_sort(a, n , -1,cravn, perest);

break;

}

break;

}

}

void bubble\_sort(double \*a, int n, int por, int& cravn, int& perest)

{

double temp;

for(int i=0; i<n-1; i++)

{

for (int j=n-2; j>=i; j--)

{

cravn++;

if (por==-1)

{

if (a[j]<a[j+1])

{

temp=a[j];a[j]=a[j+1];a[j+1]=temp;

perest++;

}

}

if (por==1)

{

if (a[j]>a[j+1])

{

temp=a[j];a[j]=a[j+1];a[j+1]=temp;

perest++;

}

}

}

}

}

void SelectSort(double\*a, int n, int por, int& cravn, int& perest)

{

int irep;

double temp;

for (int i=0; i<n; i++)

{

irep=i;

for (int j=i+1; j<n; j++)

{

cravn++;

if (por == -1)

{

if (a[j]>a[irep])

{

irep=j;

}

}

if (por == 1)

{

if (a[j]<a[irep])

{

irep=j;

}

}

}

if (irep != i)

{

temp =a[i];

a[i]=a[irep];

a[irep]=temp;

perest++;

}

}

}

void InsertSort (double\*a, int n, int por, int& cravn, int& perest)

{

int j;

double x;

for(int i=1; i<n; i++)

{

x=a[i];

j=i-1;

cravn++;

if (por==-1)

{

while(x>a[j] && j>=0)

{

a[j+1]=a[j];

j=j-1;

perest++;

}

}

if (por==1)

{

while(x<a[j] && j>=0)

{

a[j+1]=a[j];

j=j-1;

perest++;

}

}

a[j+1]=x;

}

}

void shell\_sort(double\*a, int n, int por, int& cravn, int& perest)

{

int j, h=1;

double x;

while(h<=n)

{

h=3\*h+1;

}

h=(h-1)/3;

while (h>=1)

{

for (int p=0; p<h; p++)

{

for (int i=p+h; i<n;i+=h)

{

x=a[i];

j=i-h;

cravn++;

if (por==1)

{

while (x<a[j] && j>=0)

{

a[j+h]=a[j];

j=j-h;

perest++;

}

}

if (por==-1)

{

while (x>a[j] && j>=0)

{

a[j+h]=a[j];

j=j-h;

perest++;

}

}

a[j+h]=x;

}

}

h=(h-1)/3;

}

}

main

#include <iostream>

#include "func.h"

#include <fstream>

#include <iomanip>

#include <ctime>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

int n;

do

{

cout << "Введите положительное n:";

cin >> n;

cout << endl;

} while (n <= 0);

double\* a = new double[n];

menu(a,n);

cout << "Итоговый массив:" << endl;

output\_vect(a, n);

int cravn = 0;

int perest = 0;

menu2(a,n, cravn, perest);

output\_vect(a, n);

cout<<"Количество сравнений равно: "<<cravn<<endl;

cout<<"Количество перестановок равно: "<<perest<<endl;

delete []a;

return 0;

}

## Проверка работоспособности и проверочные данные

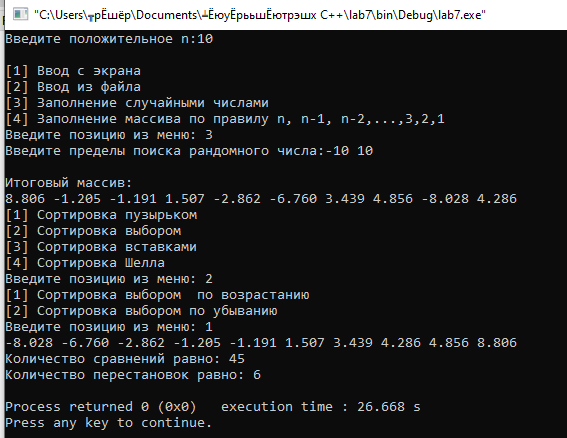
**Блок A**

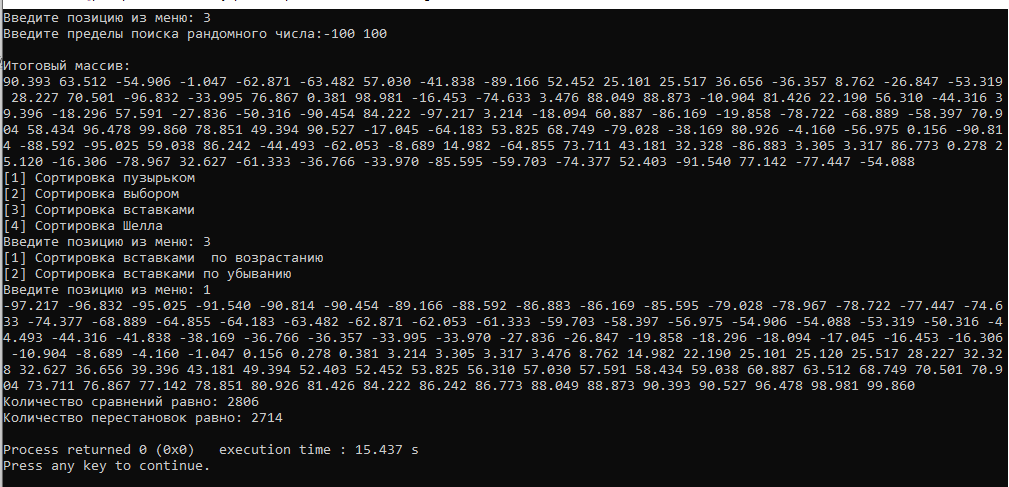
Упорядочить элементы одномерного массива по возрастанию. Использовать сортировку ***методом выбора***.

Взять массив из ***n*** чисел, заполненный случайными числами из интервала [-10,10].

Упорядочить элементы одномерного массива по возрастанию. Использовать сортировку ***методом вставки***.

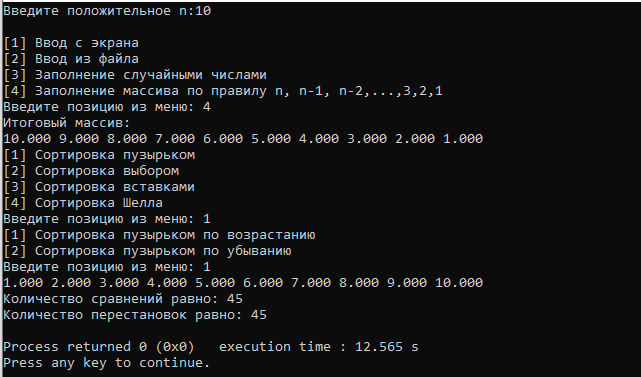
Взять массив из ***n*** чисел, заполненный случайными числами из интервала [-10,10].

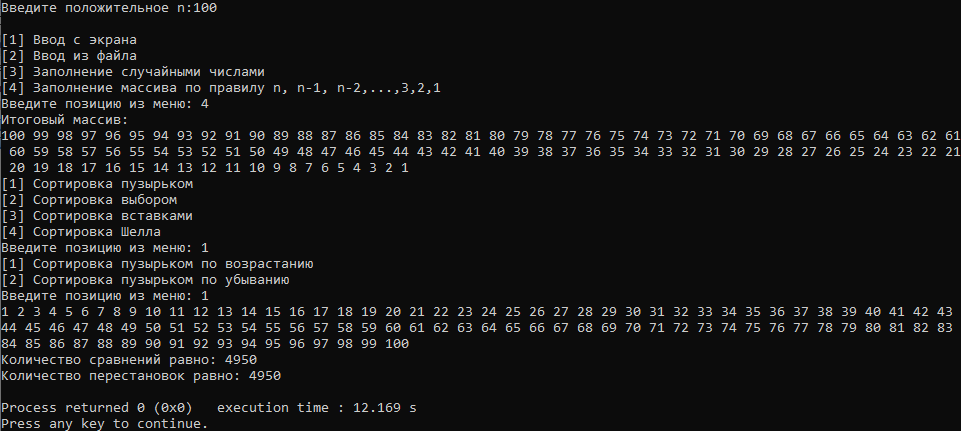
****

****

**Блок B**

**Метод пузырька**

****

****

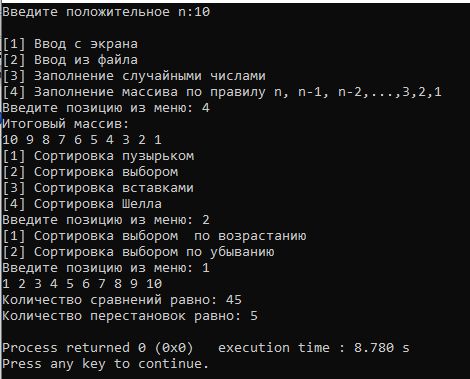
****

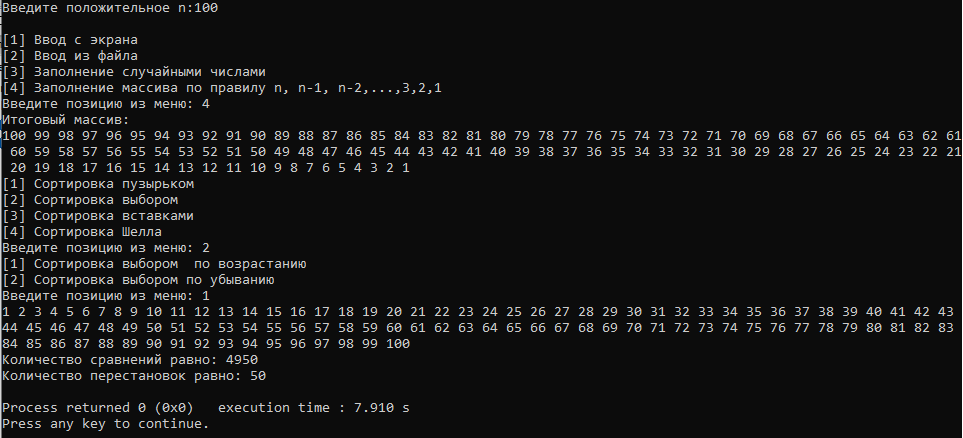
*Количество сравнений в «худшем случае»:* **(n2-n)/2**

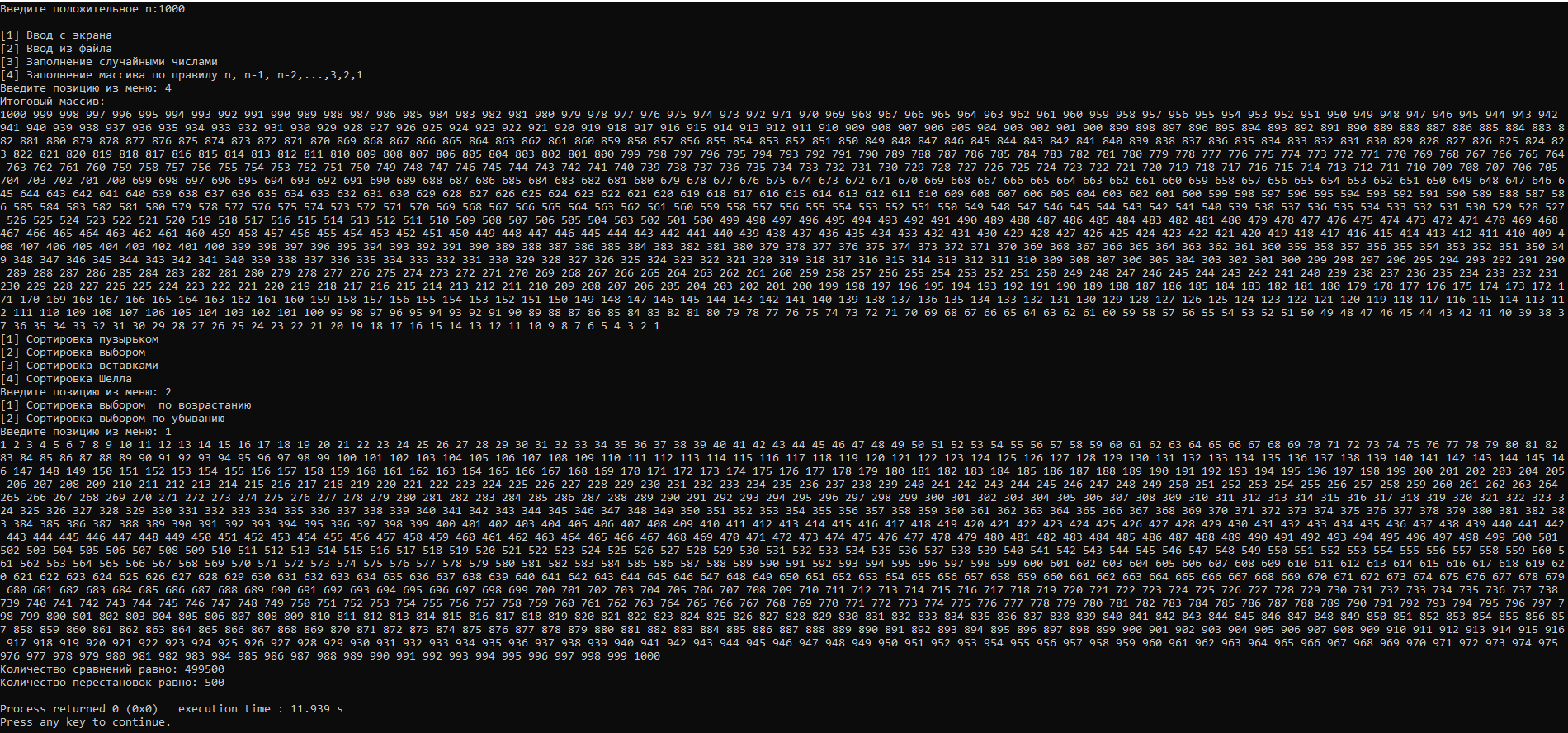
*Количество перестановок в «худшем случае»:* **3/2 (n2-n)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Размер массива** | **Количество сравнений** | | **Количество перестановок** | |
| *Теоретическая оценка* | *Полученный результат* | *Теоретическая оценка* | *Полученный результат* |
| N=10 | 45 | 45 | 135 | 45 |
| N=100 | 4950 | 4950 | 14850 | 4950 |
| N=1000 | 499500 | 499500 | 1498500 | 499500 |

**Метод выбора**

****

****

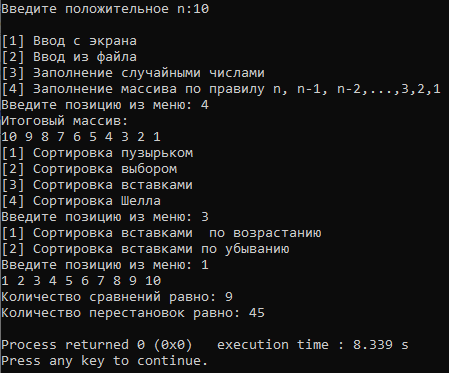
****

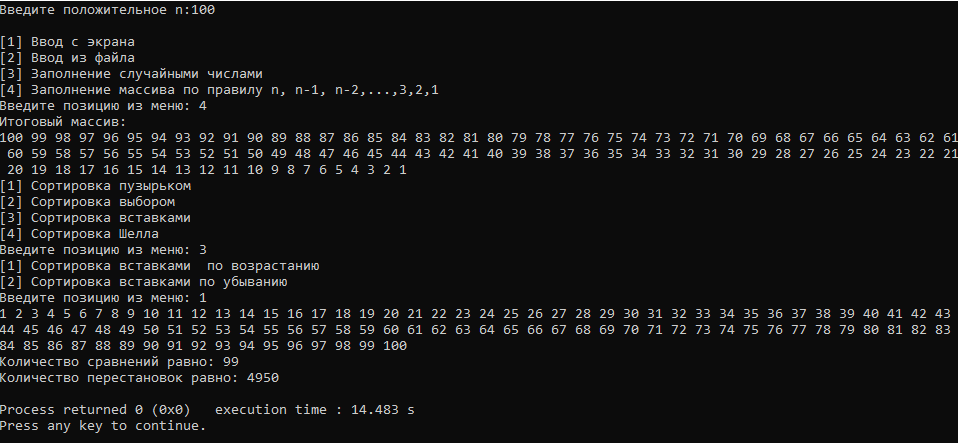
*Количество сравнений в «худшем случае»:* **(n2-n)/2**

*Количество перестановок в «худшем случае»:* **n-1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Размер массива** | **Количество сравнений** | | **Количество перестановок** | |
| *Теоретическая оценка* | *Полученный результат* | *Теоретическая оценка* | *Полученный результат* |
| N=10 | 45 | 45 | 9 | 5 |
| N=100 | 4950 | 4950 | 99 | 50 |
| N=1000 | 499500 | 499500 | 999 | 500 |

**Метод вставок**

****

****

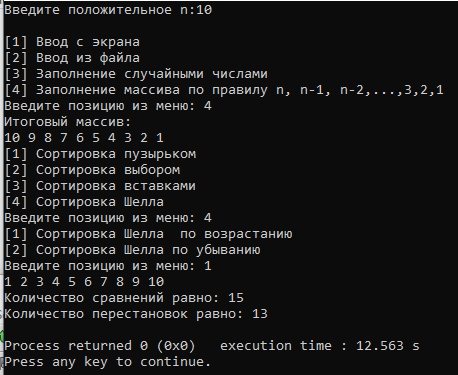
****

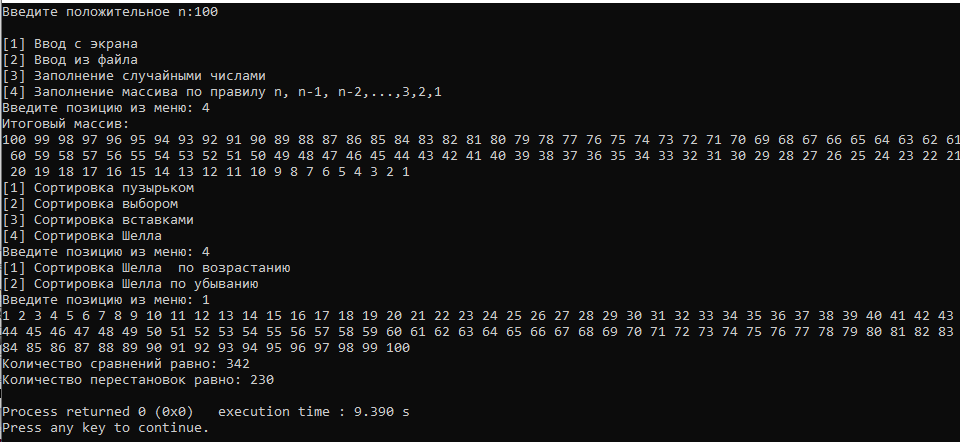
*Количество сравнений в «худшем случае»:* **1/4(n2+n-4)**

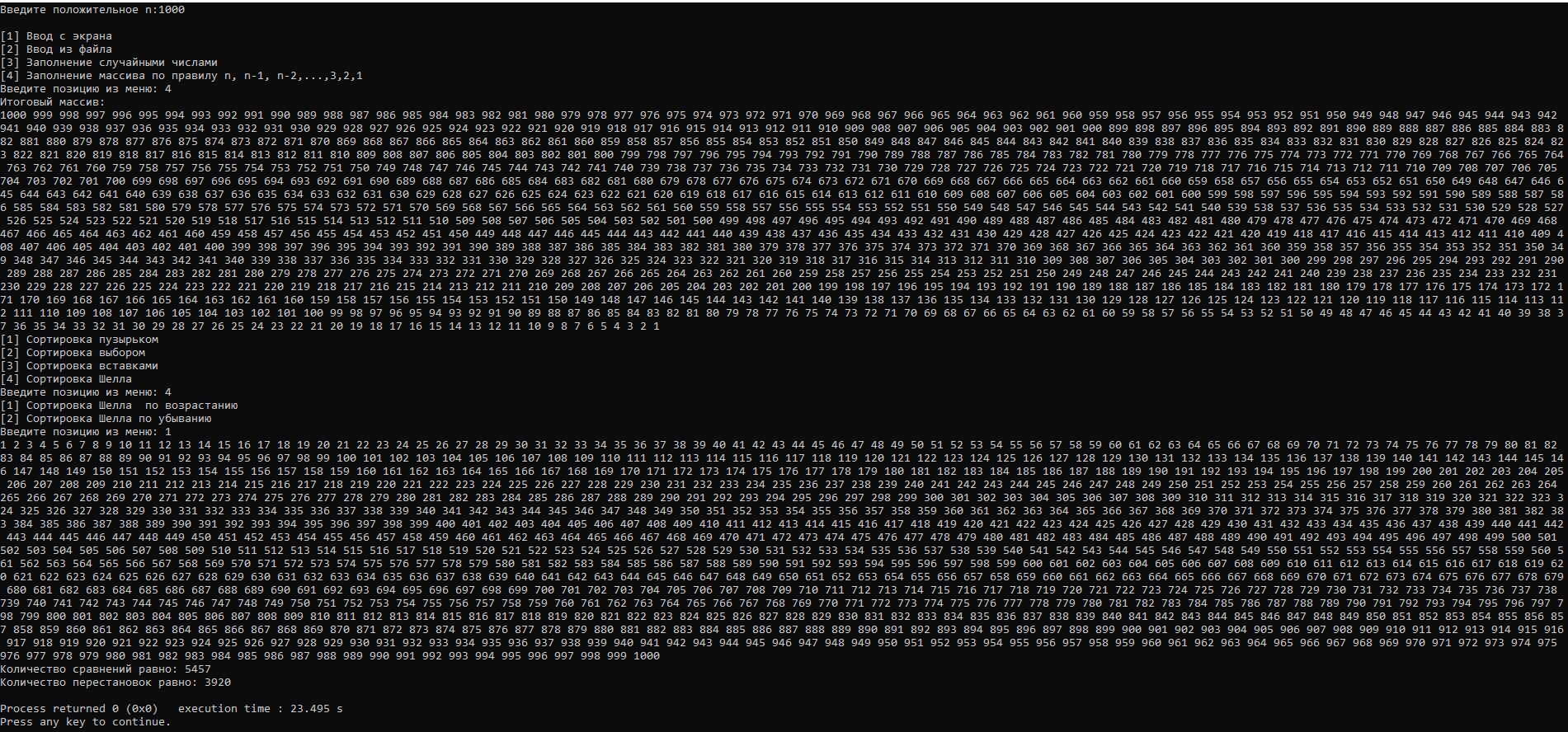
*Количество перестановок в «худшем случае»:* **1/2 (n2+3n - 4).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Размер массива** | **Количество сравнений** | | **Количество перестановок** | |
| *Теоретическая оценка* | *Полученный результат* | *Теоретическая оценка* | *Полученный результат* |
| N=10 | 26.5 | 9 | 63 | 45 |
| N=100 | 2524 | 99 | 5148 | 4950 |
| N=1000 | 250249 | 999 | 501498 | 499500 |

**Блок С**

****

****

****

*Количество сравнений в «худшем случае:* ***N1,5***

*Количество перестановок в «худшем случае»:* **0.66\*N1,25**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Размер массива** | **Количество сравнений** | | **Количество перестановок** | |
| *Теоретическая оценка* | *Полученный результат* | *Теоретическая оценка* | *Полученный результат* |
| N=10 | 31,62278 | 15 | 29,51944 | 13 |
| N=100 | 1000 | 342 | 524,9381 | 230 |
| N=1000 | 31622,78 | 5457 | 9334,866 | 3920 |

