**Лабораторная работа №3**

**Сортировка массивов**

Составить программу, проводящую сравнительную характеристику методов сортировки массивов.

Программа должна выполнять следующие действия:

1. Производить сортировку массива соответствующими методами.

2. Иллюстрировать работу каждого метода на небольших массивах (размером до 10 элементов).

3. Производить сортировку каждым из методов случайного массива, уже отсортированного массива, массива, отсортированного в обратном порядке. Засечь время. Размер массива при этом должен выбираться пользователем. После проведения сортировки, вывести данные о скорости работы методов.

**Вариант 17 (5)**: Метод прямого включения, шейкерная сортировка, метод Шелла.

**Текст программы**

#include <iostream>

#include <chrono>

#include <iomanip>

using namespace std;

void insertion(int\* m, int n) { //метод прямого включения

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

int x = m[i], j = i;

while (j != 0 && x < m[j - 1]) m[j] = m[j - 1], j--;

m[j] = x;

}

}

void shaker(int\* m, int n) { //шейкерная сортировка

int left = 0, right = n - 1, k = right;

do {

for (int j = right; j >= left; j--)

if (m[j] < m[j - 1]) {

swap(m[j], m[j - 1]);

k = j;

}

left = k + 1;

for (int j = left; j <= right; j++)

if (m[j] < m[j - 1]) {

swap(m[j], m[j - 1]);

k = j;

}

right = k - 1;

} while (left <= right);

}

void shell(int\* m, int n) { //метод Шелла

int step = n / 2;

while (step > 0) {

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

int j = i;

while (j >= 0 && m[j] > m[j + step]) {

swap(m[j], m[j + step]);

j -= step;

}

}

step /= 2;

}

}

int main() {

setlocale(LC\_ALL, "Russian");

srand(time(0));

cout << "СОРТИРОВКА НЕБОЛЬШИХ МАССИВОВ" << endl;

cout << "Исходный массив:" << endl;

const int n = 10;

int original[n], sorted[n];

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

original[i] = rand() % 1000 - 500;

cout << original[i] << ' ';

}

cout << endl;

cout << "Метод прямого включения:" << endl;

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

sorted[i] = original[i];

}

insertion(sorted, n);

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

cout << sorted[i] << ' ';

cout << endl;

cout << "Шейкерная сортировка:" << endl;

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

sorted[i] = original[i];

}

shaker(sorted, n);

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

cout << sorted[i] << ' ';

cout << endl;

cout << "Метод Шелла:" << endl;

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

sorted[i] = original[i];

}

shell(sorted, n);

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

cout << sorted[i] << ' ';

cout << endl << endl;

int m;

double result[3][3];

cout << "СОРТИРОВКА БОЛЬШИХ МАССИВОВ" << endl;

cout << "Введите размер массива: ";

cin >> m;

if (m < 1) {

cout << "Введен недопустимый размер массива!" << endl;

return 0;

}

int\* random = new int[m];

int\* random\_sorted = new int[m];

int\* forward = new int[m];

int\* forward\_sorted = new int[m];

int\* reverse = new int[m];

int\* reverse\_sorted = new int[m];

cout << "Рандомный массив" << endl;

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

random[i] = rand() % 1000 - 500;

cout << "Метод прямого включения:" << endl;

for (int i = 0; i < m; i++) {

random\_sorted[i] = random[i];

}

auto start = chrono::steady\_clock::now();

insertion(random\_sorted, m);

auto end = chrono::steady\_clock::now();

result[0][0] = chrono::duration\_cast<chrono::nanoseconds>(end - start).count() / 1000000.0;

cout << result[0][0] << " мс" << endl;

cout << "Шейкерная сортировка:" << endl;

for (int i = 0; i < m; i++) {

random\_sorted[i] = random[i];

}

start = chrono::steady\_clock::now();

shaker(random\_sorted, m);

end = chrono::steady\_clock::now();

result[1][0] = chrono::duration\_cast<chrono::nanoseconds>(end - start).count() / 1000000.0;

cout << result[1][0] << " мс" << endl;

cout << "Метод Шелла:" << endl;

for (int i = 0; i < m; i++) {

random\_sorted[i] = random[i];

}

start = chrono::steady\_clock::now();

shell(random\_sorted, m);

end = chrono::steady\_clock::now();

result[2][0] = chrono::duration\_cast<chrono::nanoseconds>(end - start).count() / 1000000.0;

cout << result[2][0] << " мс" << endl;

cout << "Отсортированный массив" << endl;

forward[0] = rand() % 1000 - 500;

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

forward[i] = forward[i - 1] + rand() % 10;

cout << "Метод прямого включения:" << endl;

for (int i = 0; i < m; i++) {

forward\_sorted[i] = forward[i];

}

start = chrono::steady\_clock::now();

insertion(forward\_sorted, m);

end = chrono::steady\_clock::now();

result[0][1] = chrono::duration\_cast<chrono::nanoseconds>(end - start).count() / 1000000.0;

cout << result[0][1] << " мс" << endl;

cout << "Шейкерная сортировка:" << endl;

for (int i = 0; i < m; i++) {

forward\_sorted[i] = forward[i];

}

start = chrono::steady\_clock::now();

shaker(forward\_sorted, m);

end = chrono::steady\_clock::now();

result[1][1] = chrono::duration\_cast<chrono::nanoseconds>(end - start).count() / 1000000.0;

cout << result[1][1] << " мс" << endl;

cout << "Метод Шелла:" << endl;

for (int i = 0; i < m; i++) {

forward\_sorted[i] = forward[i];

}

start = chrono::steady\_clock::now();

shell(forward\_sorted, m);

end = chrono::steady\_clock::now();

result[2][1] = chrono::duration\_cast<chrono::nanoseconds>(end - start).count() / 1000000.0;

cout << result[2][1] << " мс" << endl;

cout << "Отсортированный в обратном порядке массив" << endl;

reverse[0] = rand() % 1000;

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

reverse[i] = reverse[i - 1] - rand() % 10;

cout << "Метод прямого включения:" << endl;

for (int i = 0; i < m; i++) {

reverse\_sorted[i] = reverse[i];

}

start = chrono::steady\_clock::now();

insertion(reverse\_sorted, m);

end = chrono::steady\_clock::now();

result[0][2] = chrono::duration\_cast<chrono::nanoseconds>(end - start).count() / 1000000.0;

cout << result[0][2] << " мс" << endl;

cout << "Шейкерная сортировка:" << endl;

for (int i = 0; i < m; i++) {

reverse\_sorted[i] = reverse[i];

}

start = chrono::steady\_clock::now();

shaker(reverse\_sorted, m);

end = chrono::steady\_clock::now();

result[1][2] = chrono::duration\_cast<chrono::nanoseconds>(end - start).count() / 1000000.0;

cout << result[1][2] << " мс" << endl;

cout << "Метод Шелла:" << endl;

for (int i = 0; i < m; i++) {

reverse\_sorted[i] = reverse[i];

}

start = chrono::steady\_clock::now();

shell(reverse\_sorted, m);

end = chrono::steady\_clock::now();

result[2][2] = chrono::duration\_cast<chrono::nanoseconds>(end - start).count() / 1000000.0;

cout << result[2][2] << " мс" << endl;

cout << "------------------------------------------------------------------------------------------\n";

cout << "| Метод | Рандомный | Отсортированный | Обратный |\n";

cout << "|Метод прямого включения|" << setw(13) << result[0][0] << " |" << setw(15) << result[0][1] << " |" << setw(13) << result[0][2] << " |\n";

cout << "|Шейкерная сортировка |" << setw(13) << result[1][0] << " |" << setw(15) << result[1][1] << " |" << setw(13) << result[1][2] << " |\n";

cout << "|Метод Шелла |" << setw(13) << result[2][0] << " |" << setw(15) << result[2][1] << " |" << setw(13) << result[2][2] << " |\n";

cout << "------------------------------------------------------------------------------------------\n";

delete[] random;

delete[] random\_sorted;

delete[] forward;

delete[] forward\_sorted;

delete[] reverse;

delete[] reverse\_sorted;

system("pause");

return 0;

}

**Описание программы**

Функция **insertion** (реализует метод прямого включения):

1. На вход принимает массив **m** размером **n**
2. Цикл **for** выполняется от 1 до **n**:
   * Переменной **x** присваивается значение **i**-того элемента массива, переменной **j** – значение **i**
   * Пока значение переменной **j** не равно 0, а значение переменной x меньше значения элемента **m[j-1]**, элементу **m[j]** присваивается значение элемента **m[j-1]**, а значение **j** уменьшается на 1
   * Элементу **m[j]** присваивается значение переменной **x**

Функция **shaker** (реализует шейкерную сортировку):

1. На вход принимает массив **m** размером **n**
2. Переменной **left** присваивается значение 0, переменной **right** – **n-1**, переменной **k** – значение переменной **right**
3. Цикл **for** выполняется от **right** до **left**:
   * Если элемент **m[j]** меньше, чем элемент **m[j-1]**, то они меняются местами, а переменной **k** присваивается значение переменной **j**
4. Переменной **left** присваивается значение **k+1**
5. Цикл **for** выполняется от **left** до **right**:
   * Если элемент **m[j]** меньше, чем элемент **m[j-1]**, то они меняются местами, а переменной **k** присваивается значение переменной **j**
6. Переменной **right** присваивается значение **k-1**
7. Пока значение переменной **left** меньше или равно значению переменной **right**, повторять выполнение пунктов 3-6

Функция **shell** (реализует метод прямого включения):

1. На вход принимает массив **m** размером **n**
2. Переменной **step** присваивается значение **n/2**
3. Пока значение переменной **step** больше 0, выполняется цикл:

* Цикл **for** выполняется от 0 до **n-step**
  + Переменной **j** присваивается значение переменной **i**
  + Пока **j** больше или равно 0, а значение элемента **m[j]** больше значения элемента **m[j+step]**, значение переменной **j** уменьшается на **step**
* Значение переменной **step** делится на 2

Основная часть программы:

1 часть программы – сортировка небольших массивов

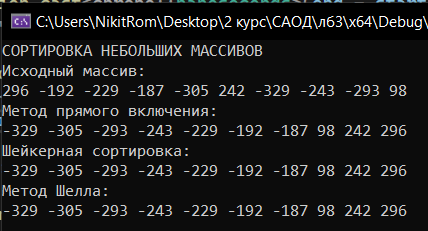
1. Инициализируется константа **n** (размер массива), ей присваивается значение 10
2. Инициализируются 2 массива размером **n**: **original** (исходный массив) и **sorted** (отсортированный массив)
3. Массив **original** заполняется рандомными значениями от -500 до 499 и выводится на экран
4. Значениям элементов массива **sorted** присваиваются значения соответствующие значения элементов массива **original**
5. Массив **sorted** сортируется методом прямого включения с помощью функции **insertion** и выводится на экран
6. Значениям элементов массива **sorted** присваиваются значения соответствующие значения элементов массива **original**
7. Массив **sorted** сортируется шейкерной сортировкой с помощью функции **shaker** и выводится на экран
8. Значениям элементов массива **sorted** присваиваются значения соответствующие значения элементов массива **original**
9. Массив **sorted** сортируется методом Шелла с помощью функции **shell** и выводится на экран

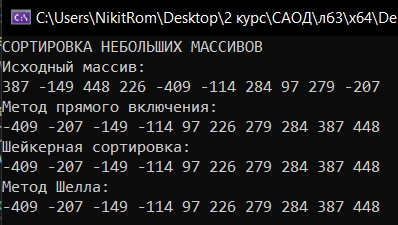
2 часть программы – сортировка больших массивов

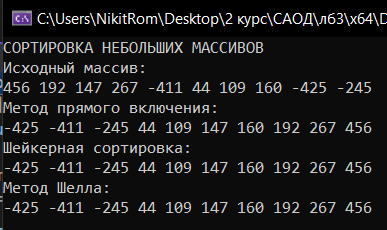
1. Инициализируется переменная **m** (размер массива) и двумерный массив **result** типа **double** размером 3х3 (для записи времени выполнения сортировок)
2. Программа запрашивает у пользователя ввод размера массива (значение переменной **m**)
3. Если пользователь ввел значение меньше 1, на экран выводится сообщение *«Введен недопустимый размер массива!»* и выполнение программы прекращается
4. Инициализируются динамические массивы размером **m**:
   * **random** – исходный массив, заполненный рандомными числами
   * **random\_sorted** – отсортированный массив **random**
   * **forward** – исходный уже отсортированный массив
   * **forward\_sorted** – отсортированный массив **forward**
   * **reverse** – исходный массив, отсортированный в обратном порядке
   * **reverse\_sorted** – отсортированный массив **reverse**
5. Массив **random** заполняется рандомными числами от -500 до 499
6. Значениям элементов массива **random\_sorted** присваиваются значения соответствующие значения элементов массива **random**
7. Включается замер времени
8. Массив **random\_sorted** сортируется методом прямого включения с помощью функции **insertion**
9. Выключается замер времени, значение измерения записывается в элемент массива **result[0][0]** и выводится на экран
10. Значениям элементов массива **random\_sorted** присваиваются значения соответствующие значения элементов массива **random**
11. Включается замер времени
12. Массив **random\_sorted** сортируется шейкерной сортировкой с помощью функции **shaker**
13. Выключается замер времени, значение измерения записывается в элемент массива **result[1][0]** и выводится на экран
14. Значениям элементов массива **random\_sorted** присваиваются значения соответствующие значения элементов массива **random**
15. Включается замер времени
16. Массив **random\_sorted** сортируется методом Шелла с помощью функции **shell**
17. Выключается замер времени, значение измерения записывается в элемент массива **result[2][0]** и выводится на экран
18. 0 элементу массива **forward** присваивается рандомное значение от -500 до 499
19. Каждому последующему элементу массива присваивается значение, большее значения предыдущего элемента от 0 до 9
20. Значениям элементов массива **forward\_sorted** присваиваются значения соответствующие значения элементов массива **forward**
21. Включается замер времени
22. Массив **forward\_sorted** сортируется методом прямого включения с помощью функции **insertion**
23. Выключается замер времени, значение измерения записывается в элемент массива **result[0][1]** и выводится на экран
24. Значениям элементов массива **forward\_sorted** присваиваются значения соответствующие значения элементов массива **forward**
25. Включается замер времени
26. Массив **forward\_sorted** сортируется шейкерной сортировкой с помощью функции **shaker**
27. Выключается замер времени, значение измерения записывается в элемент массива **result[1][1]** и выводится на экран
28. Значениям элементов массива **forward\_sorted** присваиваются значения соответствующие значения элементов массива **forward**
29. Включается замер времени
30. Массив **forward\_sorted** сортируется методом Шелла с помощью функции **shell**
31. Выключается замер времени, значение измерения записывается в элемент массива **result[2][1]** и выводится на экран
32. 0 элементу массива **reverse** присваивается рандомное значение от 0 до 999
33. Каждому последующему элементу массива присваивается значение, меньшее значения предыдущего элемента от 0 до 9
34. Значениям элементов массива **reverse\_sorted** присваиваются значения соответствующие значения элементов массива **reverse**
35. Включается замер времени
36. Массив **reverse\_sorted** сортируется методом прямого включения с помощью функции **insertion**
37. Выключается замер времени, значение измерения записывается в элемент массива **result[0][2]** и выводится на экран
38. Значениям элементов массива **reverse\_sorted** присваиваются значения соответствующие значения элементов массива **reverse**
39. Включается замер времени
40. Массив **reverse\_sorted** сортируется шейкерной сортировкой с помощью функции **shaker**
41. Выключается замер времени, значение измерения записывается в элемент массива **result[1][2]** и выводится на экран
42. Значениям элементов массива **reverse\_sorted** присваиваются значения соответствующие значения элементов массива **reverse**
43. Включается замер времени
44. Массив **reverse\_sorted** сортируется методом Шелла с помощью функции **shell**
45. Выключается замер времени, значение измерения записывается в элемент массива **result[2][2]** и выводится на экран
46. Значения всех 9 измерений выводятся в таблицу
47. Массивы **random**, **random\_sorted**, **forward**, **forward\_sorted**, **reverse** и **reverse\_sorted** удаляются

**Результаты тестов**

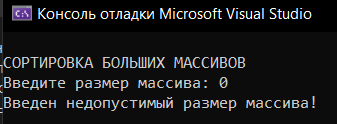
Сортировка небольших массивов:

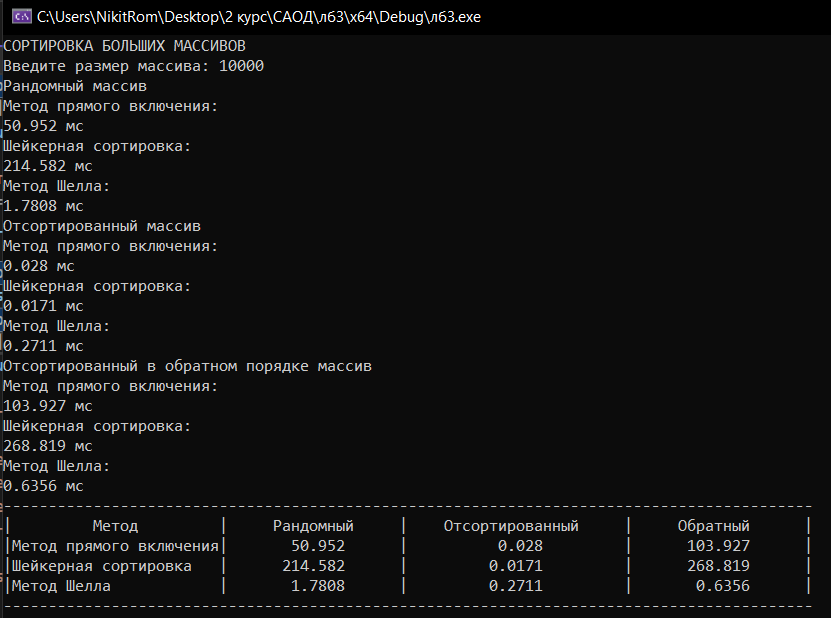


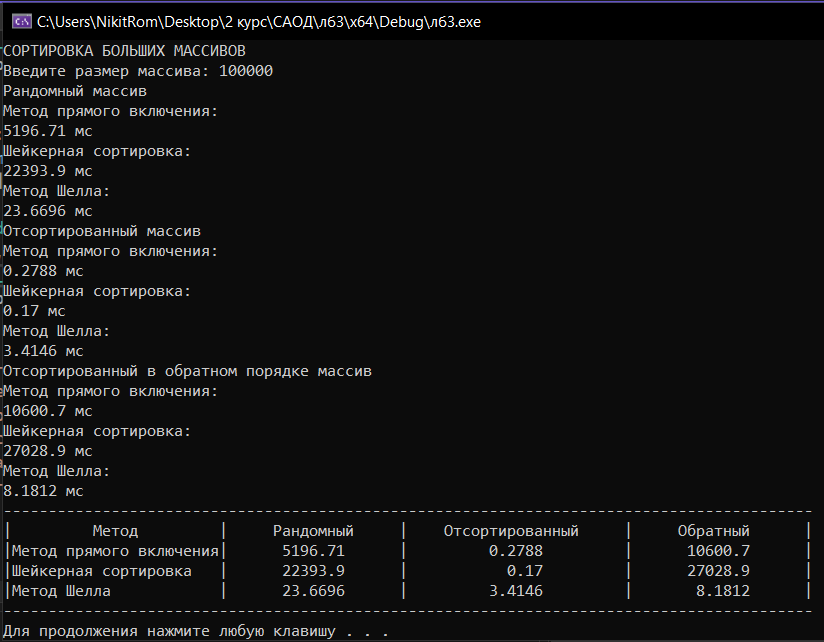




Сортировка больших массивов:







**Пример сортировки**

Для массива из примера №1 рассмотрим принципы сортировок.

Исходный массив: 296, -192, -229, -187, -305, 242, -329, -243, -293, 98

Метод прямого включения:

1) -192

296, -192, -229, -187, -305, 242, -329, -243, -293, 98

296, 296, -229, -187, -305, 242, -329, -243, -293, 98

-192, 296, -229, -187, -305, 242, -329, -243, -293, 98

2) -229

-192, 296, -229, -187, -305, 242, -329, -243, -293, 98

-192, 296, 296, -187, -305, 242, -329, -243, -293, 98

-192, -192, 296, -187, -305, 242, -329, -243, -293, 98

-229, -192, 296, -187, -305, 242, -329, -243, -293, 98

3) -187

-229, -192, 296, -187, -305, 242, -329, -243, -293, 98

-229, -192, 296, 296, -305, 242, -329, -243, -293, 98

-229, -192, -187, 296, -305, 242, -329, -243, -293, 98

4) -305

-229, -192, -187, 296, -305, 242, -329, -243, -293, 98

-229, -192, -187, 296, 296, 242, -329, -243, -293, 98

-229, -192, -187, -187, 296, 242, -329, -243, -293, 98

-229, -192, -192, -187, 296, 242, -329, -243, -293, 98

-229, -229, -192, -187, 296, 242, -329, -243, -293, 98

-305, -229, -192, -187, 296, 242, -329, -243, -293, 98

5) 242

-305, -229, -192, -187, 296, 242, -329, -243, -293, 98

-305, -229, -192, -187, 296, 296, -329, -243, -293, 98

-305, -229, -192, -187, 242, 296, -329, -243, -293, 98

6) -329

-305, -229, -192, -187, 242, 296, -329, -243, -293, 98

-305, -229, -192, -187, 242, 296, 296, -243, -293, 98

-305, -229, -192, -187, 242, 242, 296, -243, -293, 98

-305, -229, -192, -187, -187, 242, 296, -243, -293, 98

-305, -229, -192, -192, -187, 242, 296, -243, -293, 98

-305, -229, -229, -192, -187, 242, 296, -243, -293, 98

-305, -305, -229, -192, -187, 242, 296, -243, -293, 98

-329, -305, -229, -192, -187, 242, 296, -243, -293, 98

7) -243

-329, -305, -229, -192, -187, 242, 296, -243, -293, 98

-329, -305, -229, -192, -187, 242, 296, 296, -293, 98

-329, -305, -229, -192, -187, 242, 242, 296, -293, 98

-329, -305, -229, -192, -187, -187, 242, 296, -293, 98

-329, -305, -229, -192, -192, -187, 242, 296, -293, 98

-329, -305, -229, -229, -192, -187, 242, 296, -293, 98

-329, -305, -243, -229, -192, -187, 242, 296, -293, 98

8) -293

-329, -305, -243, -229, -192, -187, 242, 296, -293, 98

-329, -305, -243, -229, -192, -187, 242, 296, 296, 98

-329, -305, -243, -229, -192, -187, 242, 242, 296, 98

-329, -305, -243, -229, -192, -187, -187, 242, 296, 98

-329, -305, -243, -229, -192, -192, -187, 242, 296, 98

-329, -305, -243, -229, -229, -192, -187, 242, 296, 98

-329, -305, -243, -243, -229, -192, -187, 242, 296, 98

-329, -305, -293, -243, -229, -192, -187, 242, 296, 98

9) 98

-329, -305, -293, -243, -229, -192, -187, 242, 296, 98

-329, -305, -293, -243, -229, -192, -187, 242, 296, 296

-329, -305, -293, -243, -229, -192, -187, 242, 242, 296

-329, -305, -293, -243, -229, -192, -187, 98, 242, 296

Шейкерная сортировка:

1 проход:

296, -192, -229, -187, -305, 242, -329, -243, -293, 98

296, -192, -229, -187, -305, 242, -329, -243, -293, 98

296, -192, -229, -187, -305, 242, -329, -293, -243, 98

296, -192, -229, -187, -305, 242, -329, -293, -243, 98

296, -192, -229, -187, -305, -329, 242, -293, -243, 98

296, -192, -229, -187, -329, -305, 242, -293, -243, 98

296, -192, -229, -329, -187, -305, 242, -293, -243, 98

296, -192, -329, -229, -187, -305, 242, -293, -243, 98

296, -329, -192, -229, -187, -305, 242, -293, -243, 98

-329, 296, -192, -229, -187, -305, 242, -293, -243, 98

-329, 296, -192, -229, -187, -305, 242, -293, -243, 98

-329, -192, 296, -229, -187, -305, 242, -293, -243, 98

-329, -192, -229, 296, -187, -305, 242, -293, -243, 98

-329, -192, -229, -187, 296, -305, 242, -293, -243, 98

-329, -192, -229, -187, -305, 296, 242, -293, -243, 98

-329, -192, -229, -187, -305, 242, 296, -293, -243, 98

-329, -192, -229, -187, -305, 242, -293, 296, -243, 98

-329, -192, -229, -187, -305, 242, -293, -243, 296, 98

-329, -192, -229, -187, -305, 242, -293, -243, 98, 296

2 проход:

-329, -192, -229, -187, -305, 242, -293, -243, 98, 296

-329, -192, -229, -187, -305, 242, -293, -243, 98, 296

-329, -192, -229, -187, -305, 242, -293, -243, 98, 296

-329, -192, -229, -187, -305, -293, 242, -243, 98, 296

-329, -192, -229, -187, -305, -293, 242, -243, 98, 296

-329, -192, -229, -305, -187, -293, 242, -243, 98, 296

-329, -192, -305, -229, -187, -293, 242, -243, 98, 296

-329, -305, -192, -229, -187, -293, 242, -243, 98, 296

-329, -305, -192, -229, -187, -293, 242, -243, 98, 296

-329, -305, -229, -192, -187, -293, 242, -243, 98, 296

-329, -305, -229, -192, -187, -293, 242, -243, 98, 296

-329, -305, -229, -192, -293, -187, 242, -243, 98, 296

-329, -305, -229, -192, -293, -187, 242, -243, 98, 296

-329, -305, -229, -192, -293, -187, -243, 242, 98, 296

-329, -305, -229, -192, -293, -187, -243, 98, 242, 296

3 проход:

-329, -305, -229, -192, -293, -187, -243, 98, 242, 296

-329, -305, -229, -192, -293, -187, -243, 98, 242, 296

-329, -305, -229, -192, -293, -243, -187, 98, 242, 296

-329, -305, -229, -192, -293, -243, -187, 98, 242, 296

-329, -305, -229, -293, -192, -243, -187, 98, 242, 296

-329, -305, -293, -229, -192, -243, -187, 98, 242, 296

-329, -305, -293, -229, -192, -243, -187, 98, 242, 296

-329, -305, -293, -229, -192, -243, -187, 98, 242, 296

-329, -305, -293, -229, -243, -192, -187, 98, 242, 296

-329, -305, -293, -229, -243, -192, -187, 98, 242, 296

-329, -305, -293, -229, -243, -192, -187, 98, 242, 296

4 проход:

-329, -305, -293, -229, -243, -192, -187, 98, 242, 296

-329, -305, -293, -229, -243, -192, -187, 98, 242, 296

-329, -305, -293, -229, -243, -192, -187, 98, 242, 296

-329, -305, -293, -243, -229, -192, -187, 98, 242, 296

-329, -305, -293, -243, -229, -192, -187, 98, 242, 296

-329, -305, -293, -243, -229, -192, -187, 98, 242, 296

-329, -305, -293, -243, -229, -192, -187, 98, 242, 296

5 проход:

-329, -305, -293, -243, -229, -192, -187, 98, 242, 296

-329, -305, -293, -243, -229, -192, -187, 98, 242, 296

-329, -305, -293, -243, -229, -192, -187, 98, 242, 296

Метод Шелла:

296, -192, -229, -187, -305, 242, -329, -243, -293, 98

242, -329, -243, -293, -305, 296, -192, -229, -187, 98

242, -329, -243, -293, -305, 296, -192, -229, -187, 98

-305, -329, -243, -293, -192, -229, -187, 98, 242, 296

-305, -329, -243, -293, -192, -229, -187, 98, 242, 296

-329, -305, -293, -243, -229, -192, -187, 98, 242, 296

**Вывод**

Мною была написана программа, выполняющая сортировку массивов методом прямого включения, шейкерной сортировкой и методом Шелла. Также программа измеряет время выполнения всех сортировок.

По результатам измерения можно сделать следующие выводы:

* На массивах, заполненных рандомными элементами, эффективнее всего себя показала сортировка Шелла, хуже всего – шейкерная сортировка
* На уже отсортированном массиве эффективнее всего была шейкерная сортировка, а наименее эффективной – сортировка Шелла
* На отсортированном в обратном порядке наиболее эффективной была сортировка Шелла, а наименее эффективной – шейкерная сортировка