

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра информационных систем**

**ОТЧЕТ**  
**по практической работе №2**  
**по дисциплине «Программирование»**  
**Тема: Одномерные статические массивы**

Студент гр.

Ильбильдин Р.М., 3372

Преподаватель

Глущенко А. Г.

Санкт-Петербург

2023

### Цель работы.

Разработать программу, позволяющая выполнять различные операции с массивами, включая сортировку и поиск.

### Основные теоретические положения.

Массив – структура данных, хранящая набор значений, идентифицируемых по индексу или набору индексов, принимающих целые значения из некоторого заданного непрерывного диапазона. Одномерный массив можно рассматривать как реализацию абстрактного типа данных. Подробная информация о массиве представлена на рисунке 1.



Рисунок 1 – Одномерный массив

Сортировка массива – это расположение элементов массива в некотором заданном порядке (по возрастанию или убыванию). Например, на рисунке 2 представлена сортировка пузырьком.

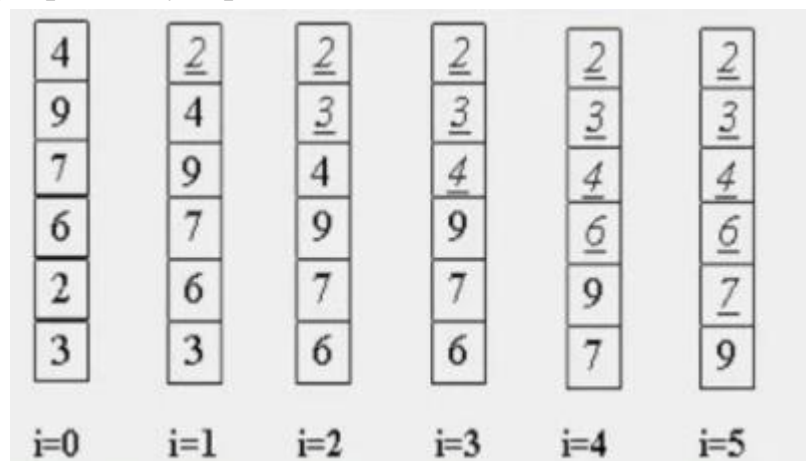


Рисунок 2 – Сортировка пузырьком

В данной работе представлены следующие сортировки:

– сортировка полным перебором (от меньше к большему) – это сортировка выполняющая полный перебор всех элементов массива сравнивая и переставляя их при удовлетворенности условий;

- «быстрая сортировка» – один из самых популярных алгоритмов, в котором используется принцип «разделяй и властвуй»;
- сортировка «слиянием» – это сортировка, где массив делится пополам, каждый из них сортируется слиянием и потом соединяются оба массива.

Поиск элемента в массиве дает возможность удостовериться о наличии данного элемента в части массива.

В неотсортированном массиве поиск элемента необходимо просматривать весь массив (линейный поиск).

В отсортированном массиве поиск элемента можно осуществлять, не просматривая весь массив. Например, бинарный поиск представлен на рисунке 3.

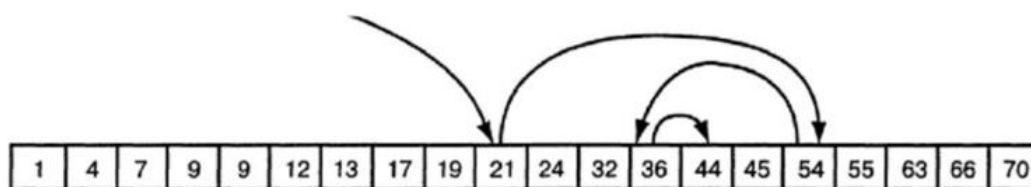


Рисунок 3 – Бинарный поиск

### Постановка задачи.

Разработать алгоритм и написать программу, которая позволяет:

1. Создать целочисленный массив размерности  $N = 100$ . Элементы массива должны принимать случайное значение в диапазоне от -99 до 99.
2. Отсортировать заданный в пункте 1 массив [...] сортировкой (от меньшего к большему). Определить время, затраченное на сортировку, используя библиотеку `chrono`.
3. Найти максимальный и минимальный элемент массива. Подсчитать время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку `chrono`.
4. Выводить среднее значение (если необходимо, число нужно округлить) максимального и минимального значения в отсортированном и неотсортированном. Выводить индексы всех элементов, которые равны этому значению, и их количество. Подсчитать время поиска.

5. Выводить количество элементов в отсортированном массиве, которые меньше числа *a*, которое инициализируется пользователем.

6. Выводить количество элементов в отсортированном массиве, которые больше числа *b*, которое инициализируется пользователем.

7. Выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализуйте алгоритм бинарного поиска. Сравните скорость его работы с обычным перебором.

8. Менять местами элементы массива, индексы которых вводит пользователь. Выводить скорость обмена, используя библиотеку `chrono`.

### **Выполнение работы.**

Код программы представлен в приложении А.

### **Выводы.**

Данная практическая работа позволяет сформировать навыки и умения работы с массивами, усвоить принцип сортировки и поиска, сравнить время работы различных функций и методов.

## ПРИЛОЖЕНИЕ А

### РАБОЧИЙ КОД

#### Листинг А.1 – Рабочий код программы

```
#include <iostream>
#include <chrono>
#include <thread>

using namespace std;
using namespace std::chrono;

const int sizeArray = 100;
int minValueArr, maxValueArr;
steady_clock::time_point startTime;
steady_clock::time_point endTime;

void displayArray(int arr[], int sizeArr) {
    for (int i = 0; i < sizeArr; i++)
        cout << arr[i] << " ";

    cout << "\n\n";
}

void fillArray(int arr[], int sizeArr) {
    const int minRand = -99, maxRand = 99;
    for (int i = 0; i < sizeArr; i++)
        arr[i] = rand() % (maxRand - minRand + 1) + minRand;
}

void bruteForceSort() {
    int arrayNum[sizeArray];
    fillArray(arrayNum, sizeArray);
    displayArray(arrayNum, sizeArray);

    startTime = steady_clock::now();
    for (int i = 0; i <= sizeArray; i++)
        for (int j = 0; j <= sizeArray; j++)
            if (arrayNum[j] < arrayNum[i])
            {
                int temp = arrayNum[j];
                arrayNum[j] = arrayNum[i];
                arrayNum[i] = temp;
            }
    endTime = steady_clock::now();

    displayArray(arrayNum, sizeArray);
}
```

```

        cout << "Time spent Brute Force Sorting: "
              << duration_cast<nanoseconds>(endTime - startTime).count()
              << " nanoseconds.\n";
    }

    void quickSort(int arr[], int left, int right) {
        if (left > right)
            return;

        int supportElem = arr[(left + right) / 2];
        int leftArr = left, rightArr = right;

        while (leftArr <= rightArr)
        {
            while (arr[leftArr] < supportElem)
                leftArr++;

            while (arr[rightArr] > supportElem)
                rightArr--;

            if (leftArr <= rightArr)
            {
                int temp = arr[leftArr];
                arr[leftArr] = arr[rightArr];
                arr[rightArr] = temp;
                leftArr++;
                rightArr--;
            }
        }

        quickSort(arr, left, rightArr);
        quickSort(arr, leftArr, right);
    }

    void merge(int array[], int const left, int const mid,
              int const right)
    {
        int const subArrayOne = mid - left + 1;
        int const subArrayTwo = right - mid;

        auto* leftArray = new int[subArrayOne],
              * rightArray = new int[subArrayTwo];

        for (auto i = 0; i < subArrayOne; i++)
            leftArray[i] = array[left + i];
        for (auto j = 0; j < subArrayTwo; j++)
            rightArray[j] = array[mid + 1 + j];
    }

```

```

auto indexOfSubArrayOne = 0, indexOfSubArrayTwo = 0;
int indexOfMergedArray = left;

while (indexOfSubArrayOne < subArrayOne
      && indexOfSubArrayTwo < subArrayTwo) {
    if (leftArray[indexOfSubArrayOne]
        <= rightArray[indexOfSubArrayTwo]) {
        array[indexOfMergedArray]
            = leftArray[indexOfSubArrayOne];
        indexOfSubArrayOne++;
    }
    else {
        array[indexOfMergedArray]
            = rightArray[indexOfSubArrayTwo];
        indexOfSubArrayTwo++;
    }
    indexOfMergedArray++;
}

while (indexOfSubArrayOne < subArrayOne) {
    array[indexOfMergedArray]
        = leftArray[indexOfSubArrayOne];
    indexOfSubArrayOne++;
    indexOfMergedArray++;
}

while (indexOfSubArrayTwo < subArrayTwo) {
    array[indexOfMergedArray]
        = rightArray[indexOfSubArrayTwo];
    indexOfSubArrayTwo++;
    indexOfMergedArray++;
}

delete[] leftArray;
delete[] rightArray;
}

void mergeSort(int array[], int const begin, int const end)
{
    if (begin >= end)
        return;

    int mid = begin + (end - begin) / 2;
    mergeSort(array, begin, mid);
    mergeSort(array, mid + 1, end);
    merge(array, begin, mid, end);
}

void searchMinMax(int arr[], int sizeArr, bool isSort) {

```

```

    startTime = steady_clock::now();
    if (isSort)
    {
        minValueArr = arr[0];
        maxValueArr = arr[sizeArr - 1];
    }
    else {
        minValueArr = arr[0], maxValueArr = arr[0];
        for (int i = 0; i < sizeArr; i++)
        {
            if (arr[i] <= minValueArr)
                minValueArr = arr[i];

            if (arr[i] >= maxValueArr)
                maxValueArr = arr[i];
        }
    }
    endTime = steady_clock::now();
}

void averageMinMax(int arr[], int sizeArr, bool isSort) {
    int avgValue = 0;
    int countAvg = 0;

    searchMinMax(arr, sizeArr, isSort);
    avgValue = (minValueArr + maxValueArr) / 2;
    cout << "Average Value in Array: " << avgValue << "\n";

    cout << "Index element array equal to the average value: ";
    startTime = steady_clock::now();
    for (int i = 0; i < sizeArr; i++)
    {
        if (arr[i] == avgValue)
        {
            cout << i << " ";
            countAvg++;
        }
    }
    endTime = steady_clock::now();
    cout << "\n";
    countAvg == 0 ? cout << "Index not found, Count 0\n" : cout << "Count index equal to the
average value: " << countAvg << "\n";
    cout << "Time spent search count average value in array: " <<
duration_cast<nanoseconds>(endTime - startTime).count() << " nanoseconds.\n";
}

void countLessValue(int arr[], int sizeArr, int value) {
    int countLess = 0;

```



```

        for (countLess = 0; countLess < sizeArr; countLess++)
        {
            if (arr[countLess] >= value)
                break;
        }

        cout << "Count less value a: " << countLess << "\n";
    }

void countMoreValue(int arr[], int sizeArr, int value) {
    int countMore;

    for (countMore = sizeArray - 1; countMore >= 0; countMore--)
    {
        if (arr[countMore] <= value)
            break;
    }

    cout << "Count more value: " << sizeArray - 1 - countMore << "\n";
}

int linearSearch(int arr[], int sizeArr, int value) {
    startTime = steady_clock::now();
    for (int i = 0; i < sizeArr; i++)
    {
        if (value == arr[i])
        {
            endTime = steady_clock::now();
            return i;
        }
    }
    endTime = steady_clock::now();
    return -1;
}

int binarySearch(int arr[], int sizeArr, int value) {
    startTime = steady_clock::now();
    int low = 0, high = sizeArr - 1;
    while (low <= high)
    {
        int middle = low + (high - low) / 2;
        if (value == arr[middle]) {
            endTime = steady_clock::now();
            return middle;
        }

        if (value > arr[middle]) {
            low = middle + 1;
        }
    }
}

```

```

        if (value < arr[middle]) {
            high = middle - 1;
        }
    }
    endTime = steady_clock::now();
    return -1;
}

void swapElement(int arr[], int sizeArr, int first, int second) {
    startTime = steady_clock::now();
    if (first < 0 || second < 0 || first >= sizeArr || second >= sizeArr) {
        endTime = steady_clock::now();
        cout << "Error! Incorrect values.\n\n";
        return;
    }
    else {
        int temp = arr[first];
        arr[first] = arr[second];
        arr[second] = temp;
    }
    endTime = steady_clock::now();
    cout << "Time spent Swap Element: "
        << duration_cast<nanoseconds>(endTime - startTime).count()
        << " nanoseconds.\n\n";
}

int main()
{
    bool isLive = true;
    while (isLive)
    {
        int arrayNum[sizeArray];
        fillArray(arrayNum, sizeArray);
        int userSelectNum, chooseUserArray, numberUser, result;

        int firstIndex, secondIndex;

        bool isSort;

        cout << "Please, Select a number to choose actions with array: \n";
        cout << "\t 1 - Brute Force Sorting\n";
        cout << "\t 2 - Quick Sorting\n";
        cout << "\t 3 - Merge Sorting\n";
        cout << "\t 4 - Search Min and Max value\n";
        cout << "\t 5 - Display Average Min and Max\n";
        cout << "\t 6 - Display Count Less Value\n";
        cout << "\t 7 - Display Count More Value\n";
        cout << "\t 8 - Binary Search\n";
    }
}

```

```

cout << "\t 9 - Swap value\n";
cout << "Number: ";
cin >> userSelectNum;
cout << "\n";

switch (userSelectNum)
{
case 1:
    cout << "===== Brute Force Sort =====\n";
    bruteForceSort();
    cout << "===== \n\n";
    break;

case 2:
    cout << "===== Quick Sort =====\n";
    displayArray(arrayNum, sizeArray);

    startTime = steady_clock::now();
    quickSort(arrayNum, 0, sizeArray - 1);
    endTime = steady_clock::now();

    displayArray(arrayNum, sizeArray);
    cout << "Time spent Quick Sorting: "
        << duration_cast<nanoseconds>(endTime - startTime).count()
        << " nanoseconds.\n\n";
    cout << "===== \n\n";
    break;

case 3:
    cout << "===== Merge Sort =====\n";
    displayArray(arrayNum, sizeArray);
    startTime = steady_clock::now();
    mergeSort(arrayNum, 0, sizeArray - 1);
    endTime = steady_clock::now();
    displayArray(arrayNum, sizeArray);
    cout << "Time spent Merge Sorting: "
        << duration_cast<nanoseconds>(endTime - startTime).count()
        << " nanoseconds.\n\n";
    cout << "===== \n\n";
    break;

case 4:
    cout << "===== Search Min and Max value =====\n";
    cout << "Do you want Search Min and Max in?\n";
    cout << "\t1 - UnSorted Array\n";
    cout << "\t2 - Sorted Array\n";
    cout << "Number: ";
    cin >> chooseUserArray;
    switch (chooseUserArray)

```

```

{
case 1:
    isSort = false;
    searchMinMax(arrayNum, sizeArray, isSort);
    break;

case 2:
    quickSort(arrayNum, 0, sizeArray - 1);
    isSort = true;
    searchMinMax(arrayNum, sizeArray, isSort);
    break;
default:
    break;
}
cout << "Min value in array: " << minValueArr << "\n";
cout << "Max value in array: " << maxValueArr << "\n";
cout << "Time spent search min and max value in not sorting array: " <<
duration_cast<nanoseconds>(endTime - startTime).count() << " nanoseconds.\n";
cout << "=====\n\n";
break;

case 5:
    cout << "==== Average Min and Max value =====\n";
    cout << "Do you want Average search Min and Max in?\n";
    cout << "\t1 - UnSorted Array\n";
    cout << "\t2 - Sorted Array\n";
    cout << "Number: ";
    cin >> chooseUserArray;
    switch (chooseUserArray)
    {
    case 1:
        isSort = false;
        displayArray(arrayNum, sizeArray);
        averageMinMax(arrayNum, sizeArray, isSort);
        break;

    case 2:
        quickSort(arrayNum, 0, sizeArray - 1);
        isSort = true;
        displayArray(arrayNum, sizeArray);
        averageMinMax(arrayNum, sizeArray, isSort);
        break;
    default:
        break;
    }
    cout << "=====\n\n";
    break;

case 6:

```

```

cout << "===== Display Count Less value =====\n";
quickSort(arrayNum, 0, sizeArray - 1);
displayArray(arrayNum, sizeArray);
cout << "Add a number for search count less value: ";
cin >> numberUser;
countLessValue(arrayNum, sizeArray, numberUser);
cout << "===== \n\n";
break;

```

case 7:

```

cout << "===== Display Count More value =====\n";
quickSort(arrayNum, 0, sizeArray - 1);
displayArray(arrayNum, sizeArray);
cout << "Add a number for search count more value: ";
cin >> numberUser;
countMoreValue(arrayNum, sizeArray, numberUser);
cout << "===== \n\n";
break;

```

case 8:

```

cout << "===== Binary Search =====\n";
quickSort(arrayNum, 0, sizeArray - 1);
displayArray(arrayNum, sizeArray);
cout << "Add a number for search element in array: ";
cin >> numberUser;

result = linearSearch(arrayNum, sizeArray, numberUser);
cout << "\nTime spent Linear Search: "
    << duration_cast<nanoseconds>(endTime - startTime).count()
    << " nanoseconds.\n\n";

result = binarySearch(arrayNum, sizeArray, numberUser);
cout << "Time spent Binary Search: "
    << duration_cast<nanoseconds>(endTime - startTime).count()
    << " nanoseconds.\n\n";

result == -1
    ? cout << "Given value does not exist in the array!\n"
    : cout << "The Index of the value is: " << result << "!\n";
cout << "===== \n\n";
break;

```

case 9:

```

cout << "===== Swap Elements =====\n";
displayArray(arrayNum, sizeArray);
cout << "Add a first index: ";
cin >> firstIndex;

cout << "Add a second index: ";

```

```

        cin >> secondIndex;

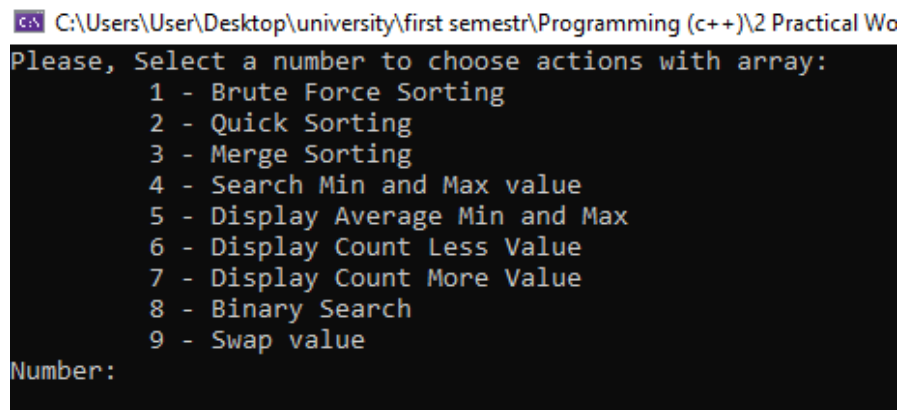
        swapElement(arrayNum, sizeArray, firstIndex, secondIndex);

        displayArray(arrayNum, sizeArray);
        cout << "=====\\n\\n";
        break;
    default:
        break;
    }
}
cout << "Hello World!\\n";
}

```

## Ход работы

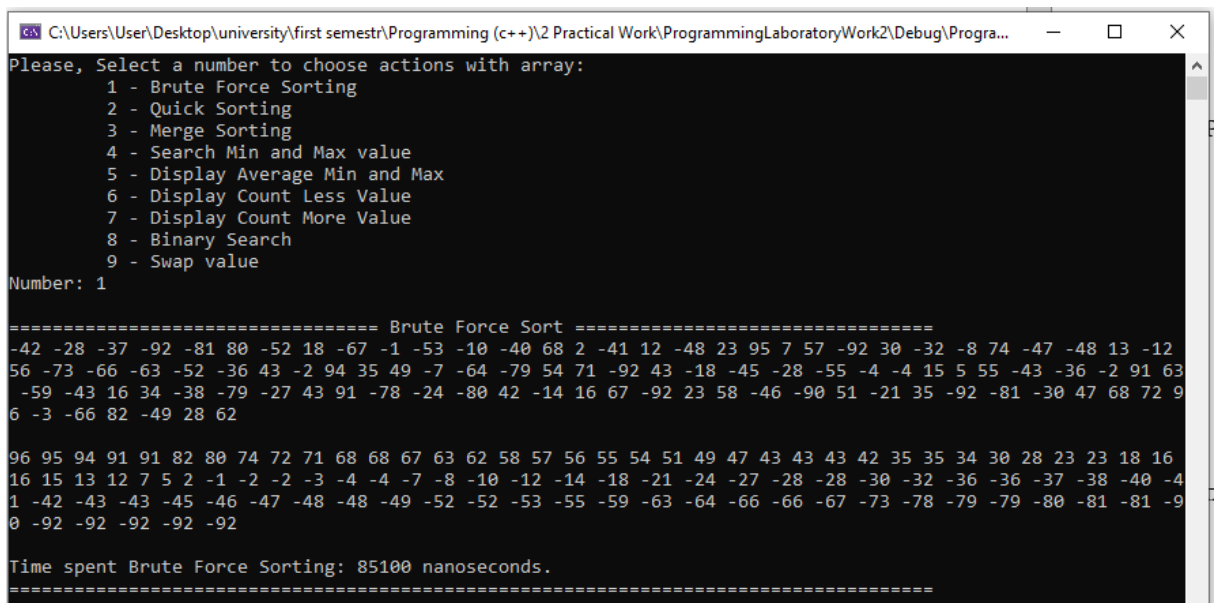
1. При запуске программы необходимо выбрать номер действия (рисунок 1).



```
C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Wo
Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number:
```

Рисунок 1 – Выбор номер поведения программы

2. При выборе пункта «1» происходит сортировка полным перебором всех элементов, отображая массив до сортировки и после, при этом идет подсчет времени самой сортировки (рисунок 2).



```
C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Work\ProgrammingLaboratory\Work2\Debug\Progra...
Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number: 1

===== Brute Force Sort =====
-42 -28 -37 -92 -81 80 -52 18 -67 -1 -53 -10 -40 68 2 -41 12 -48 23 95 7 57 -92 30 -32 -8 74 -47 -48 13 -12
56 -73 -66 -63 -52 -36 43 -2 94 35 49 -7 -64 -79 54 71 -92 43 -18 -45 -28 -55 -4 -4 15 5 55 -43 -36 -2 91 63
-59 -43 16 34 -38 -79 -27 43 91 -78 -24 -80 42 -14 16 67 -92 23 58 -46 -90 51 -21 35 -92 -81 -30 47 68 72 9
6 -3 -66 82 -49 28 62

96 95 94 91 91 82 80 74 72 71 68 68 67 63 62 58 57 56 55 54 51 49 47 43 43 43 42 35 35 34 30 28 23 23 18 16
16 15 13 12 7 5 2 -1 -2 -2 -3 -4 -4 -7 -8 -10 -12 -14 -18 -21 -24 -27 -28 -28 -30 -32 -36 -36 -37 -38 -40 -4
1 -42 -43 -43 -45 -46 -47 -48 -48 -49 -52 -52 -53 -55 -59 -63 -64 -66 -66 -67 -73 -78 -79 -79 -80 -81 -81 -9
0 -92 -92 -92 -92 -92

Time spent Brute Force Sorting: 85100 nanoseconds.
=====
```

Рисунок 2 – Сортировка полного перебора

3. При выборе пункта «2» осуществляется «быстрая» сортировка и отображение время выполнения сортировки (рисунок 3).

```

C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Work\ProgrammingLaboratoryWork2\Debug\Progra...
Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number: 2

===== Quick Sort =====
81 -45 -69 39 4 -24 -97 15 13 -44 -44 -36 -26 73 -35 -33 -88 88 9 -84 -23 95 -38 51 -14 27 26 -48 -16 -4 -27
58 -12 3 50 -6 75 41 -68 54 -31 1 -18 -93 98 -77 15 -96 -47 69 -16 -52 94 56 -69 60 -5 11 70 76 -80 -56 64
-76 91 59 74 45 -28 28 75 -80 -67 12 1 66 -86 23 9 -95 51 62 -31 -61 32 6 -72 -61 4 37 -75 84 61 -23 -18 -12
90 65 -49 -24

-97 -96 -95 -93 -88 -86 -84 -80 -80 -77 -76 -75 -72 -69 -69 -68 -67 -61 -61 -56 -52 -49 -48 -47 -45 -44 -44
-38 -36 -35 -33 -31 -31 -28 -27 -26 -24 -24 -23 -23 -18 -18 -16 -16 -14 -12 -12 -6 -5 -4 1 1 3 4 4 6 9 9 11
12 13 15 15 23 26 27 28 32 37 39 41 45 50 51 51 54 56 58 59 60 61 62 64 65 66 69 70 73 74 75 75 76 81 84 88
90 91 94 95 98

Time spent Quick Sorting: 32200 nanoseconds.
=====

```

Рисунок 3 – «Быстрая» сортировка

4. При выборе пункта «3» выполняется сортировка «слиянием» и отображается время выполнение этой сортировки (рисунок 4).

```

C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Work\ProgrammingLaboratoryWork2\Debug\Progra...
Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number: 3

===== Merge Sort =====
-23 -13 80 -63 33 -91 -48 -67 -21 60 -3 51 -72 72 -79 -14 61 27 71 92 -37 25 66 -85 94 12 14 19 -71 86 -88 6
4 18 -36 -94 60 -26 75 -84 25 35 -22 42 54 17 -37 -58 -49 35 -4 -36 -43 65 23 -16 65 13 24 65 72 63 53 -70 1
9 5 68 -75 72 40 25 -14 -41 69 53 -47 55 -3 63 79 -11 -18 55 -51 -71 -46 -56 -89 -63 61 54 -18 -95 -5 -39 -7
-88 25 -60 56 -81

-95 -94 -91 -89 -88 -88 -85 -84 -81 -79 -75 -72 -71 -71 -70 -67 -63 -63 -60 -58 -56 -51 -49 -48 -47 -46 -43
-41 -39 -37 -37 -36 -36 -26 -23 -22 -21 -18 -18 -16 -14 -14 -13 -11 -7 -5 -4 -3 -3 5 12 13 14 17 18 19 19 23
24 25 25 25 25 27 33 35 35 40 42 51 53 53 54 54 55 55 56 60 60 61 61 63 63 64 65 65 65 66 68 69 71 72 72 72
75 79 80 86 92 94

Time spent Merge Sorting: 265200 nanoseconds.
=====

```

Рисунок 4 – Сортировка «слиянием»

5. При выборе пункта «4» осуществляется поиск минимального и максимального элемента в неотсортированном (рисунок 5) и отсортированном (рисунок 6) массиве и расчет времени этого поиска.



```
C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Work\ProgrammingLaboratoryWork2\Debug\Progra...
Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number: 4

===== Search Min and Max value =====
Do you want Search Min and Max in?
1 - UnSorted Array
2 - Sorted Array
Number: 1
Min value in array: -99
Max value in array: 93
Time spent search min and max value in not sorting array: 3000 nanoseconds.
=====
```

Рисунок 5 – Поиск минимального и максимального в неотсортированном массиве

```
C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Work\ProgrammingLaboratoryWork2\Debug\Progra...
Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number: 4

===== Search Min and Max value =====
Do you want Search Min and Max in?
1 - UnSorted Array
2 - Sorted Array
Number: 2
Min value in array: -97
Max value in array: 99
Time spent search min and max value in not sorting array: 1700 nanoseconds.
=====
```

Рисунок 6 – Поиск минимального и максимального в отсортированном массиве

6. При выборе пункта «5» отображается среднее минимального и максимального элемента в неотсортированном (рисунок 7) и отсортированном (рисунок 8) массиве, также осуществляется поиск этого среднего в массиве.

```
C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Work\ProgrammingLaboratoryWork2\Debug\Progra...
Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number: 5
===== Average Min and Max value =====
Do you want Average search Min and Max in?
1 - UnSorted Array
2 - Sorted Array
Number: 1
-26 44 -44 -23 67 15 -62 60 -65 -5 -18 43 70 -14 91 -84 -45 98 -54 -32 84 -96 -72 -65 97 -98 61 2 -14 25 55
80 -91 20 50 -67 27 -57 61 36 -61 37 -9 60 -20 73 -17 -86 -77 -92 -46 -50 -64 51 41 -78 -70 86 95 -97 21 -78
-49 -88 45 -25 55 15 37 -96 -51 -56 91 -57 -53 -51 -5 12 -60 -72 -62 5 -10 86 0 -10 -39 97 53 21 5 8 17 27
-45 13 54 -70 10 -61
Average Value in Array: 0
Index element array equal to the average value: 84
Count index equal to the average value: 1
Time spent search count average value in array: 2136000 nanoseconds.
=====
```

Рисунок 7 – Подсчет и поиск среднего в массиве в неотсортированном массиве

```
C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Work\ProgrammingLaboratoryWork2\Debug\Progra...
Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number: 5
===== Average Min and Max value =====
Do you want Average search Min and Max in?
1 - UnSorted Array
2 - Sorted Array
Number: 2
-99 -96 -95 -89 -88 -86 -82 -81 -78 -75 -74 -72 -71 -71 -70 -68 -66 -65 -63 -61 -50 -48 -48 -47 -46 -44 -44
-43 -42 -41 -39 -34 -29 -25 -25 -21 -20 -20 -19 -17 -16 -14 -11 -6 -4 -4 -1 1 4 5 5 5 5 5 6 8 13 14 17 19 22
26 28 28 31 32 32 35 38 38 39 45 47 49 52 54 54 58 61 62 62 67 67 68 68 72 75 76 77 78 78 86 89 89 93 93 95
96 96 99
Average Value in Array: 0
Index element array equal to the average value:
Index not found, Count 0
Time spent search count average value in array: 1500 nanoseconds.
=====
```

Рисунок 8 – Подсчет и поиск среднего в массиве в отсортированном массиве

7. При выборе пункта «6» и «7» отображается количество элементов массива, меньше и больше введенного значения пользователя соответственно (рисунок 9-10).

```
C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Work\ProgrammingLaboratoryWork2\Debug\ProgrammingLaboratory...
Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number: 6

===== Display Count Less value =====
-98 -97 -97 -96 -94 -89 -89 -88 -87 -86 -82 -82 -75 -73 -72 -70 -68 -66 -64 -59 -58 -55 -55 -54 -53 -52 -49 -49 -48 -48
-45 -42 -42 -36 -34 -32 -25 -24 -22 -21 -20 -18 -14 -14 -13 -13 -12 -10 -7 -7 -6 -6 -2 0 2 3 6 9 12 16 21 22 22 24 25 28
31 34 36 38 44 50 50 54 54 60 61 62 62 63 65 66 66 67 70 78 80 82 83 84 87 87 90 91 95 95 96 96 98 99

Add a number for search count less value: -89
Count less value a: 5
=====
```

Рисунок 9 – Отображение количества элементов, меньше значения

```
C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Work\ProgrammingLaboratoryWork2\Debug\ProgrammingLaboratory...
Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number: 7

===== Display Count More value =====
-98 -97 -97 -96 -94 -89 -89 -88 -87 -86 -82 -82 -75 -73 -72 -70 -68 -66 -64 -59 -58 -55 -55 -54 -53 -52 -49 -49 -48 -48
-45 -42 -42 -36 -34 -32 -25 -24 -22 -21 -20 -18 -14 -14 -13 -13 -12 -10 -7 -7 -6 -6 -2 0 2 3 6 9 12 16 21 22 22 24 25 28
31 34 36 38 44 50 50 54 54 60 61 62 62 63 65 66 66 67 70 78 80 82 83 84 87 87 90 91 95 95 96 96 98 99

Add a number for search count more value: 88
Count more value: 8
=====
```

Рисунок 10 – Отображение количества элементов, больше значения

8. При выборе пункта «8» выполняется бинарный поиск элемента массива, введенного пользователем (рисунок 11).

```
C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Work\ProgrammingLaboratoryWork2\Debug\ProgrammingLaboratory...
Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number: 8

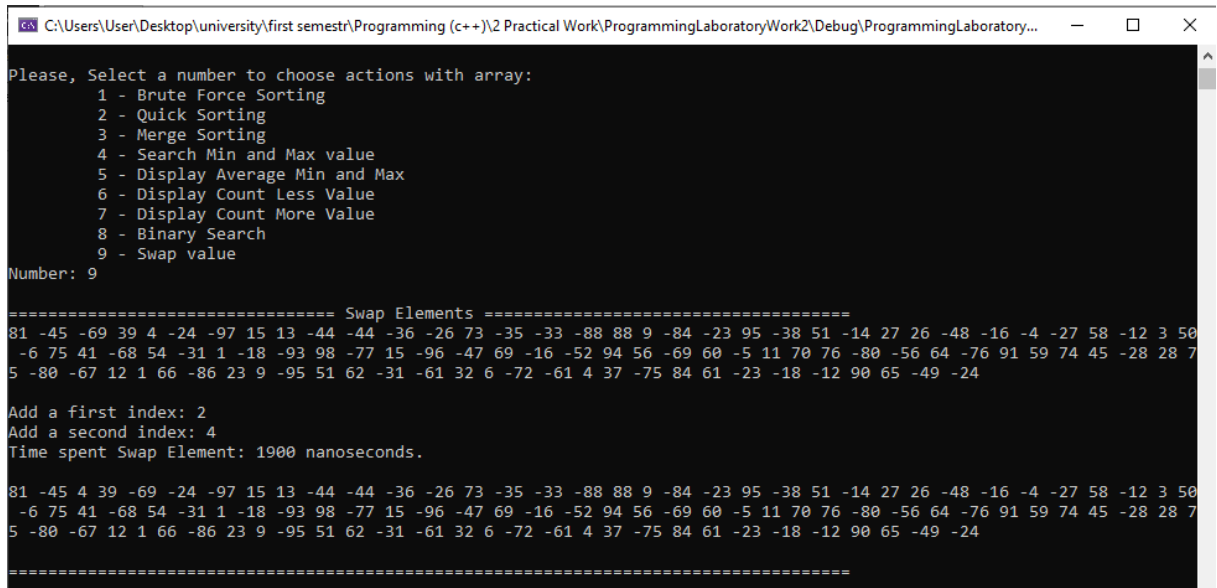
===== Binary Search =====
-92 -92 -92 -92 -92 -90 -81 -81 -80 -79 -79 -78 -73 -67 -66 -66 -64 -63 -59 -55 -53 -52 -52 -49 -48 -48 -47 -46 -45 -43
-43 -42 -41 -40 -38 -37 -36 -36 -32 -30 -28 -28 -27 -24 -21 -18 -14 -12 -10 -8 -7 -4 -4 -3 -2 -2 -1 2 5 7 12 13 15 16 16
18 23 23 28 30 34 35 35 42 43 43 47 49 51 54 55 56 57 58 62 63 67 68 68 71 72 74 80 82 91 91 94 95 96

Add a number for search element in array: 7

Time spent Linear Search: 2200 nanoseconds.
Time spent Binary Search: 1000 nanoseconds.
The Index of the value is: 59!
=====
```

Рисунок 11 – Бинарный поиск

9. При выборе пункта «9» осуществляется смена элементов массива при помощи индексов, введенных пользователем (рисунок 12).



```
C:\Users\User\Desktop\university\first semestr\Programming (c++)\2 Practical Work\ProgrammingLaboratory\Work2\Debug\ProgrammingLaboratory...

Please, Select a number to choose actions with array:
1 - Brute Force Sorting
2 - Quick Sorting
3 - Merge Sorting
4 - Search Min and Max value
5 - Display Average Min and Max
6 - Display Count Less Value
7 - Display Count More Value
8 - Binary Search
9 - Swap value
Number: 9

===== Swap Elements =====
81 -45 -69 39 4 -24 -97 15 13 -44 -44 -36 -26 73 -35 -33 -88 88 9 -84 -23 95 -38 51 -14 27 26 -48 -16 -4 -27 58 -12 3 50
-6 75 41 -68 54 -31 1 -18 -93 98 -77 15 -96 -47 69 -16 -52 94 56 -69 60 -5 11 70 76 -80 -56 64 -76 91 59 74 45 -28 28 7
5 -80 -67 12 1 66 -86 23 9 -95 51 62 -31 -61 32 6 -72 -61 4 37 -75 84 61 -23 -18 -12 90 65 -49 -24

Add a first index: 2
Add a second index: 4
Time spent Swap Element: 1900 nanoseconds.

81 -45 4 39 -69 -24 -97 15 13 -44 -44 -36 -26 73 -35 -33 -88 88 9 -84 -23 95 -38 51 -14 27 26 -48 -16 -4 -27 58 -12 3 50
-6 75 41 -68 54 -31 1 -18 -93 98 -77 15 -96 -47 69 -16 -52 94 56 -69 60 -5 11 70 76 -80 -56 64 -76 91 59 74 45 -28 28 7
5 -80 -67 12 1 66 -86 23 9 -95 51 62 -31 -61 32 6 -72 -61 4 37 -75 84 61 -23 -18 -12 90 65 -49 -24

=====
```

Рисунок 12 – Смена элементов