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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

Тема: Сортировки

Студентка гр. 9383 _____ Лапина А.А.

Преподаватель _____ Попова Е.В.

Санкт-Петербург

2020

Цель работы.

Изучить алгоритмы сортировки, научиться использовать их на практике.

Задание.

Вариант №8 .

Быстрая сортировка, рекурсивная реализация. Процедура трёхчастного разделения. деление производится не на две группы, а на три: <x, =x, >x.

Теория.

Сортировка — последовательное расположение или разбиение на группы чего-либо в зависимости от выбранного критерия.

Быстрая сортировка — один из самых известных и широко используемых алгоритмов сортировки. Среднее время работы O(nlogn), что является асимптотически оптимальным временем работы для алгоритма, основанного на сравнении. Хотя время работы алгоритма для массива из n элементов в худшем случае может составить Θ(n 2), на практике этот алгоритм является одним из самых быстрых.

Рекурсия - это такой способ организации вычислительного процесса, при котором процедура или функция в ходе выполнения составляющих ее операторов обращается сама к себе. Рекурсия очень широко применяется в математике и программировании.

Описание алгоритма — быстрая сортировка по убыванию (возрастанию):

1) В массиве выбирается опорный элемент — первый.

2) Проходим по массиву, выделяем 1 группу элементов — больше (меньше) опорного. Если находим элемент больше (меньше) опорного, то ставим его перед опорным.

3) Далее, проходим по элементам, которые остались справа от отпорного, там находятся элементы, которые меньше (больше) или равны опорному, из них необходимо выделить 2 группу — равных опорному элементу. Если текущий элемент меньше (больше) опорного то оставляем его на месте.

3) Получается, что мы разделили все элементы на 3 группы — меньше опорного, равных опорному и больше опорного.

4) Затем функция вызывается для правой и левой части рекурсивно, если в этих подмассивах больше одного элемента.

Ход работы:

1. Проанализировать данные задачи, выделив рекурсивные действия с данными.
2. Составить схему выполнения программы.
3. Разработать программу с использованием рекурсии.
4. Протестировать программу

Выполнение работы:

В данной задаче нам дается массив, который нужно отсортировать. Реализуем 2 способа сортировки — по возрастанию (функция Qsort_v) и убыванию (функция Qsost_u).

1) В функции main() считываем количество элементов в массиве и сам массив, затем пользователь выбирает как отсортировать его — по возрастанию или убыванию. Для этого использована специальная переменная выбора — choose. Если она равна 1, то будет произведена сортировка по возрастанию; если 2, то по убыванию. Иначе (в случае некорректного ввода) будет выведено сообщение о неправильном выборе типа сортировки. Поэтому сортировка проводится не будет и будет выведен изначальный массив.

2) Шаблонная функция swap(), которая меняет местами 2 элемента в массиве.

3) Шаблонная функция Qsort_v() - сортирует массив по возрастанию (алгоритм быстрой сортировки).

ind_1 — опорный элемент;

i — счетчик для элементов, с которыми будем сравнивать опорный.

4) Шаблонная функция Qsort_u() - сортирует массив по убыванию (алгоритм быстрой сортировки).

ind_1 — опорный элемент;

i — счетчик для элементов, с которыми будем сравнивать опорный.

5) Функция Print() - выводит массив.

Пример работы программы:

1) Входные данные:

5 //количество элементов

1 2 3 4 5

2 //сортировка по убыванию

Выходные данные:

5 4 3 2 1

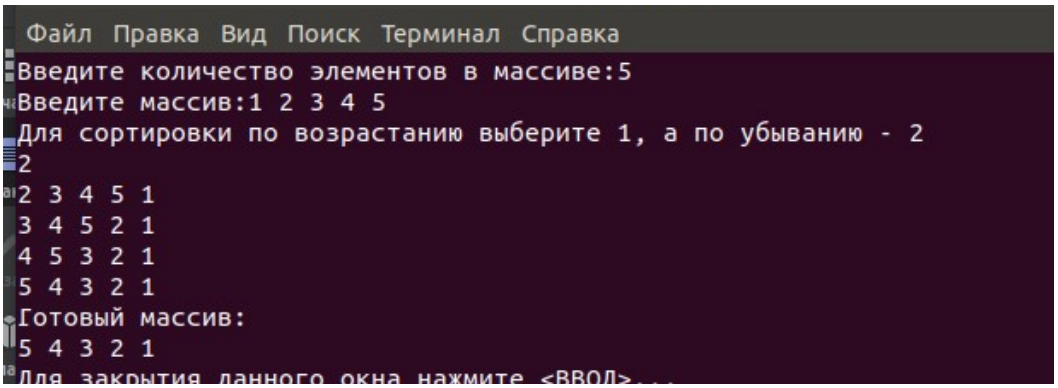

Рисунок 1 — Демонстрация работы программы с входными данными №1

2) Входные данные:

5

7 1 2 5 1

1

Выходные данные:

1 1 2 5 7

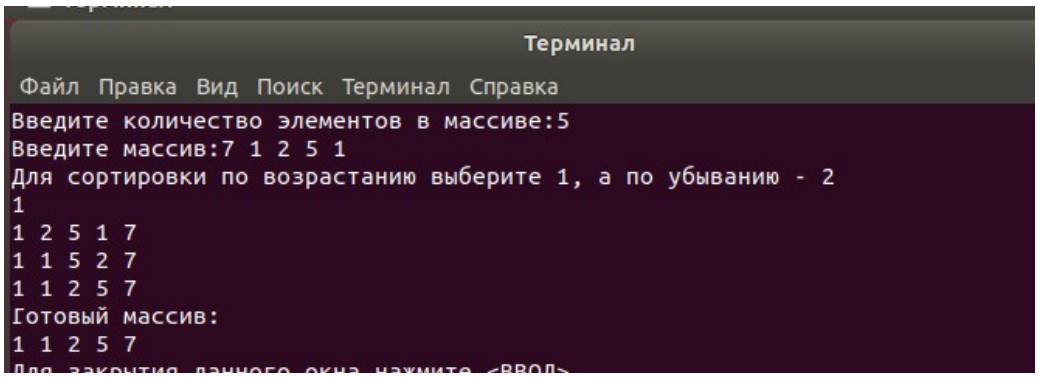

Рисунок 2 — Демонстрация работы программы с входными данными №2

3) Входные данные:

7

1 2 3 4 5 6 7

9 //так как выбран неверный тип сортировки, то выведет изначальный массив

Выходные данные:

1 2 3 4 5 6 7

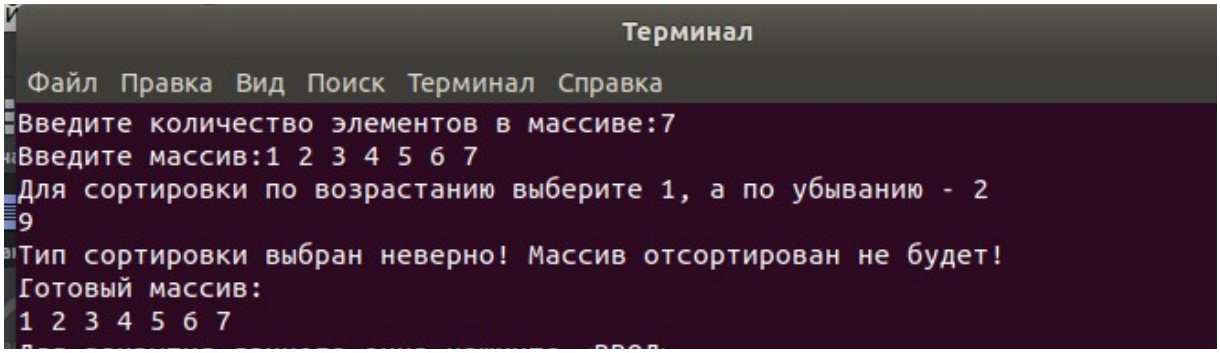

Рисунок 3 — Демонстрация работы программы с входными данными №3

4) **Входные данные:**

8

999 0 22 333 4 55 10 8

1

Выходные данные:

0 4 8 10 22 55 333 999

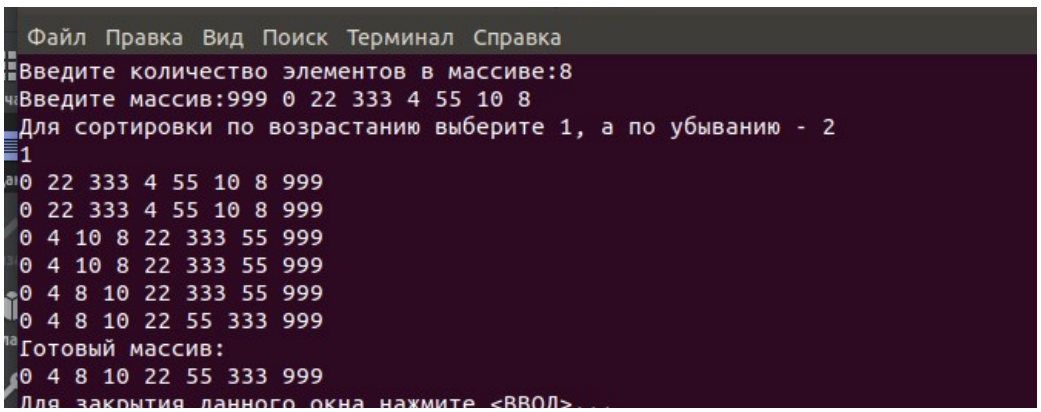

Рисунок 4 — Демонстрация работы программы с входными данными №4

Вывод.

Изучены алгоритмы сортировки, получены навыки их использования на практике.

Приложение А
Исходный код программы

Название файла: main.cpp

```
#include <iostream>

using namespace std;

template<class T>
void Print(T *arr, int n) //вывод массива
{
   cout << "Готовый массив: \n";
   for(int i = 0 ;i < n ;i++)
   {
      cout << arr[i] << ' ';
   }
   cout << "\n";
}

template<class T>
void Qsort_v(T *arr, int begin, int finish, int n)    //соритруем массив по возрастанию
{
   int i=begin+1; //счетчик для элементов
   int ind_1 = begin;  //опорный элемента

    while (i<=finish){
      if (arr[i]<arr[ind_1]){                 //группа 1, если текущий элемент меньше
опорного
          if(ind_1+1==i)
             swap(arr[i],arr[ind_1++]);
          else
          {
             swap(arr[ind_1+1],arr[ind_1]);
             swap(arr[i],arr[ind_1]);
             ind_1++;
          }
      }
      i++;
   }

    int j = 0;
    i=ind_1+1; //счетчик для элементов
    while (i<=finish){
      if (arr[i]==arr[ind_1]){                 //группа 2, если текущий элемент равен
опорному
          if(ind_1+1==i)
             swap(arr[i],arr[ind_1++]);
          else
          {
             swap(arr[ind_1+1],arr[ind_1]);
             swap(arr[i],arr[ind_1]);
             ind_1++;
```

```
            }
            j++;
        }
        i++;
    }

                                    //остальные элементы попадут в 3 группу - больше
опорного

     for (int j = 0; j<n;j++){    //вывод промежуточного результата
        cout << arr[j] << ' ';
     }
     cout << "\n";

    if (begin<ind_1-j-1)  //если левее опорного элемента есть что-то
        Qsort_v(arr, begin , ind_1-j-1, n);//для левого массива
    if (ind_1+1<finish) //если правее опорного элемента есть что-то
        Qsort_v(arr, ind_1+1, finish, n);//для правого массива
}

template<class T>
void Qsort_u(T *arr, int begin, int finish, int n)    //соритруем массив по убыванию
{
    int i=begin+1; //счетчик для элементов
    int ind_1 = begin;  //опорный элемента

     while (i<=finish){
        if (arr[i]>arr[ind_1]){                //группа 1, если текущий элемент больше
опорного
            if(ind_1+1==i)
                swap(arr[i],arr[ind_1++]);
            else
            {
                swap(arr[ind_1+1],arr[ind_1]);
                swap(arr[i],arr[ind_1]);
                ind_1++;
            }
        }
        i++;
    }

     int j = 0;
     i=ind_1+1; //счетчик для элементов
     while (i<=finish){
        if (arr[i]==arr[ind_1]){                //группа 2, если текущий элемент равен
опорному
            if(ind_1+1==i)
                swap(arr[i],arr[ind_1++]);
            else
            {
                swap(arr[ind_1+1],arr[ind_1]);
                swap(arr[i],arr[ind_1]);
                ind_1++;
            }
            j++;
        }
```

```
      i++;
   }

                                        //остальные элементы попадут в 3 группу - меньше
опорного

    for (int j = 0; j<n;j++){    //вывод промежуточного результата
      cout << arr[j] << ' ';
    }
    cout << "\n";

   if (begin<ind_1-j-1)  //если левее опорного элемента есть что-то
      Qsort_u(arr, begin , ind_1-j-1, n);//для левого массива
   if (ind_1+1<finish) //если правее опорного элемента есть что-то
      Qsort_u(arr, ind_1+1, finish, n);//для правого массива
}

template<class T>
void swap(T& a,T&b) //функция для перестановки двух элементов
{
   T temp = a;
   a = b;
   b = temp;
}

int main()
{
   int n ;
   cout<<"Введите количество элементов в массиве:";
   cin >> n;
   int arr[n];
   cout<<"Введите массив:";
   for (int i = 0 ; i<n;i++)
      cin>>arr[i];//ввод массива
   cout<<"Для сортировки по возрастанию выберите 1, а по убыванию - 2\n";
   int chouse;
   cin>>chouse;
   switch (chouse){
      case 1:{
         Qsort_v(arr,0,n-1, n);
         break;
         }
      case 2:{
         Qsort_u(arr,0,n-1, n);
         break;
      }
      default:{
         cout<<"Тип сортировки выбран неверно! Массив отсортирован не будет!\
n";
         break;
       }
   }
   Print(arr,n);
}
```