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

Студент гр. 9383 _____ Корсунов А.А.

Преподаватель _____ Попова Е.В.

Санкт-Петербург
2020

Цель работы.

Познакомиться и научиться работать с алгоритмами сортировки и написать программу на языке программирования C++.

Основные теоретические положения.

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.

Задание.

6. Быстрая сортировка, рекурсивная реализация. Во время сортировки массив должен быть в состоянии:

элементы <x, неотсортированные элементы, элементы >=x.

Ход работы.

После анализа поставленного задания была разработана программа с использованием рекурсии и библиотеки vector.

- Функция main:

В данной функции создается вектор, вводятся элементы, которые необходимо отсортировать, вызывается функция сортировки (функция qsort) и производится вывод в консоль отсортированных элементов.

- Функция qsort:

Данная рекурсивная функция принимает на вход указатель на вектор из неотсортированных элементов, индекс начального элемента, индекс конечного элемента. После чего записываются опорный элемент, его индекс, индексы конца (вспомогательные индексы для прохода по текущей стороне и вывода вектора в консоль). В цикле while производится обход по текущей стороне вектора, где все элементы, меньшие опорного элемента, записываются слева от него, а элементы, большие опорного элемента, записываются в конце вектора (на деле они остаются на своем месте, потому как обход стороны производится с ее конца). В самом цикле вызывается функция print, которая выводит изменения вектора в консоль. После цикла так же вызывается функция print, чтобы увидеть полностью вектор после каждого вызова функции qsort.

- Функция print:

Данная функция выводит либо весь вектор целиком (если в параметр icount был передан нуль) либо выводит изменения текущей стороны вектора.

Пример работы программы.

№ п/п	Входные данные	Выходные данные	Комментарии
1	4 4 3 5 6	-4 - the mid element 435----6 43----56 3----4----56 the output array: 3456	

		5 - the mid element 5----6 the output array: 3456 the finished array: 3456	
2	2 2 1	2 - the mid element 1----2 the output array: 12 the finished array: 12	
3	3 3 3 3	3 - the mid element 33----3 3----33 the output array: 333 3 - the mid element 3----3 the output array: 333 the finished array: 333	
4	6 963163	9 - the mid element 3----96316 36----9631 361----963 3613----96 36136----9 the output array: 361369 3 - the mid element 3613----6 361----36 1----36----36 1----3----636 the output array: 136369 6 - the mid element 63----6 3----6----6 the output array: 133669	

		the finished array: 133669	
5	3 2 1 1	2 - the mid element 1----21 11----2 the output array: 112 1 - the mid element 1----1 the output array: 112 the finished array: 112	

Иллюстрация работы программы.

*IDE – Code::Blocks 20.03

```
Enter the size of the array
4
Enter an array
4 3 5 6

4 - the mid element
435----6
43----56
3----4----56
the output array:       3456


5 - the mid element
5----6
the output array:       3456


the finished array:     3456
```

Рисунок 1 - Пример работы программы с входными данными №1

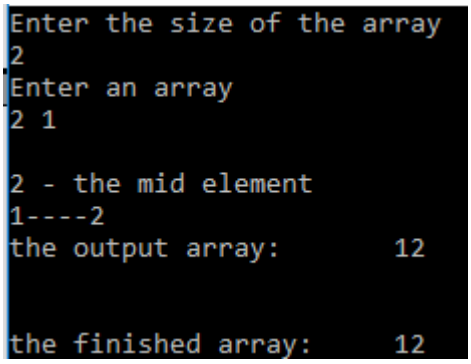

Рисунок 2 - Пример работы программы с входными данными №2

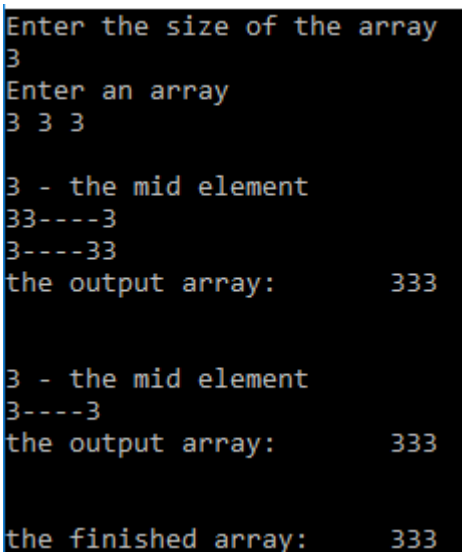

Рисунок 3 - Пример работы программы с входными данными №3

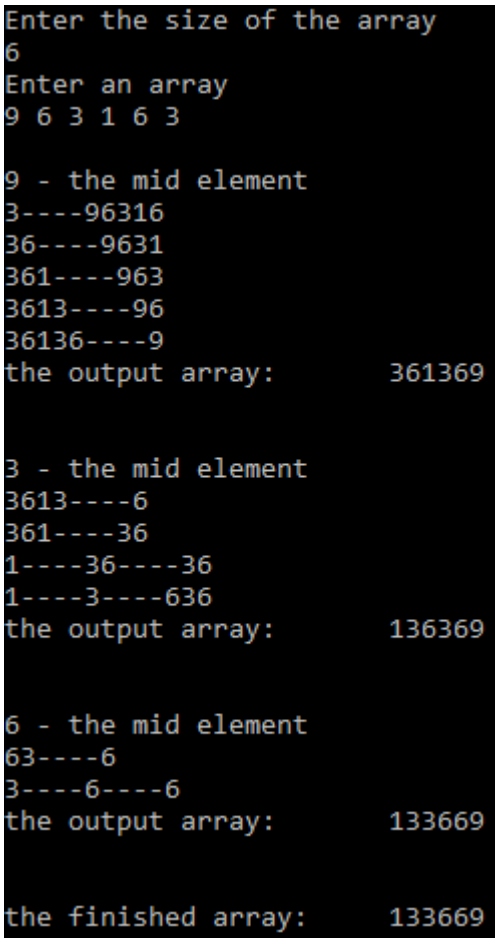

Рисунок 4 — Пример работы программы с входными данными №4

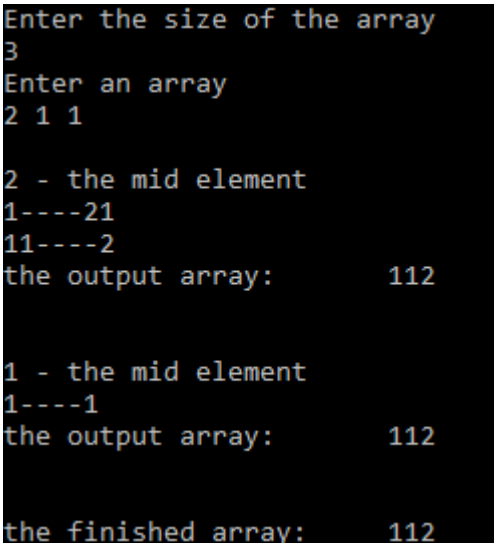

Рисунок 5 — Пример работы программы с входными данными №5

Выводы.

Произошло ознакомление с алгоритмами сортировки и была написана программа на языке программирования C++.

ПРИЛОЖЕНИЕ А

Файл main.cpp

```
#include <iostream>
#include <vector>

using namespace std;

template <typename T> void print(vector<T>* arr, int start, int finish, int icount, int index_mid) //вспомогательная функция для вывода вектора на консоль
{
   if (icount == 0) //вывод вектора в конце функции qsort
   {
      for (int i = 0; i <= finish; i++)
      {
         cout << arr->at(i);
      }
      cout << "\n";
      return;
   }
   for (int i = start; i <= finish; i++) //вывод вектора в ходе сортировки
   {
      if (i == index_mid && i != start)
      {
         cout<<"----";
      }
      else if (i == icount)
      {
         cout<<"----";
```

```
        }
        cout << arr->at(i);
    }
    cout << "\n";
}

template <typename T> void qsort(vector<T>* arr, int start, int finish)
{
    T mid = arr->at(start); //опорный элемент (элемент, относительно которого производится текущая быстрая сортировка
    int index_mid = start; //индекс опорного элемента
    int icount = finish; //переменная, с помощью которой производится обход по текущей стороне вектора
    int check = finish; //переменная, с помощью которой производится корректный вывод отсортированных эелементов
    cout << mid <<" - the mid element\n";
    do
    {
        if (mid > arr->at(check)) //если опорный элемент больше расматриваемого в цикле while, то он перемещается слева от опорного элемента
        {
            arr->insert(arr->begin()+index_mid, arr->at(check));
            index_mid++;
            arr->erase(arr->begin()+check+1);
            check++;
        }
        print(arr, start, finish, check, index_mid);
        icount--;
        check--;
    } while (icount >= start+1);
```

```
    cout << "the output array:\t";
    print(arr, 0, arr->size()-1, 0, 0); //вывод вектора в конце текущей функции
    cout << "\n\n";

    if (start < index_mid-1) //вызов сортировки для левой стороны
    {
        qsort(arr, start, index_mid-1);
    }
    if (index_mid+1 < finish) //вызов сортировеи для правой стороны
    {
        qsort(arr, index_mid+1, finish);
    }
}

int main()
{
    int key; // переменная для ввода значений

    cout << "Enter the size of the array\n";
    cin >> key;
    vector<int> arr(key);

    cout << "Enter an array\n";

    for (int i = 0; i < key; i++)
    {
        cin >> arr.at(i);
    }

    cout << "\n";
```

qsort(&arr, 0, arr.size()-1); //функция быстрой сортировки, выполненная согласно заданию (на вход поступают указатель на вектор, индекс начального элемента и индекс конечного элемента

int len = arr.size(); //len необходима для цикла (потому как i в цикле - значение типа int, а метод size() возвращает значение типа long long int

```
    cout << "the finished array:\t";
    for (int i = 0; i < len; i++) //вывод отсортированного вектора
    {
       cout << arr.at(i);
    }
    return 0;
}
```