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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных »

Тема: «Сортировки»

Студент гр. 9384		Давыдов Д.С.
Преподаватель		Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Реализовать шаблонными по типу сортируемых элементов пузырьковую и чет-нечет сортировку.

Задание.

Вариант 4.

Пузырьковая сортировка оптимизированная; сортировка чет-нечет.

Анализ задачи.

Был реализован алгоритм пузырьковой сортировки и сортировки чет-нечет.

Оптимизация первой представляет из себя следующие доработки:

1. После i-ой прогонки массива внешним циклом, i последние элементы уже находятся на своих местах в отсортированном порядке, поэтому нет необходимости производить их сравнения друг с другом, так что идем не до n−1, а до n−i−1.

2. Если после выполнения внутреннего цикла (сравнивание элементов друг с другом) не произошло ни одного обмена, то массив уже отсортирован, и продолжать работу программы уже не нужно. Введем переменную isSorted. Если после внутреннего цикла она равна true, т.е. во внутреннем цикле она не изменилась на false, что происходит при условии, когда хотя бы один левый элемент больше правого соседнего, то прекращаем работу функции сортировки.

Преимуществ пузырьковой сортировки перед другим алгоритмами сортировки не существует, возможно лишь простота его написания, тк сам по себе он является «учебным» алгоритмом, но метод сортировки обменами лежит в основе более совершенных алгоритмов.

Сложность его в лучшем случае O(n), в среднем O(n^2) и в худшем O(n^2). Сравним сортировку пузырьком с шейкерной сортировкой. Проблема сортировки пузырьком в том, что он работает медленно на тестах, в которых маленькие по значению элементы стоят в конце, они же «черепахи». Такой

элемент на каждом шаге алгоритма будет сдвигаться всего на одну позицию влево. В шейкерной же сортировке мы идем не только слева направо, но и справа налево. В памяти храним два указателя begin и end, обозначающих, какой отрезок массива еще не отсортирован. На очередной итерации при достижении end вычитаем из него единицу и движемся справа налево, аналогично, при достижении begin прибавляем единицу и двигаемся слева направо. Асимптотика у алгоритма такая же, как и у сортировки пузырьком, однако реальное время работы лучше.

Другая же модификация - сортировка расческой, сравнивает два элемента не на расстоянии 1, как это сделано в сортировке пузырьком, а возможно на гораздо большем. При правильном выборе уменьшаемого промежутка между сравнимыми элементами (он примерно равен 1.247) средняя сложность алгоритма может быть равной O(nlogn).

Реализованная же в данной лабораторной работе сортировка чет-нечет по сравнению с обычной сортировкой пузырьком проталкивает все более-менее крупные элементы массива одновременно за один пробег вправо на одну позицию. Так получается немного быстрее. Хотя сама сортировка считается чуть медленней, чем сортировка расческой и шейкерной, средняя сложность может быть равной O(nlogn).

Выполнение работы.

Был разработан примитивный API, благодаря которому пользователь может выбрать, как ввести массив — из файла или из консоли.

Далее вызываются функции пузырьковой и чет-нечет сортировки. В ходе выполнения программы в консоль выводится какие элементы меняются между собой. Компаратор сортировок принимает два шаблонных значения a и b и сравнивает их по величине (обычный операнд «>»), он возвращает true, если a > b и false иначе. Компаратор в функцию сортировки передается как указатель на функцию.

Сортировка чет-нечет работает следующим образом: сначала

сравниваются элементы с четными и нечетными позициями в массиве, затем нечетные и четными. Если левое число будет больше правого, то они меняются местами. Если после выполнения обоих сравнений `isSorted` равен true, то цикл `while` заканчивается, т.к. массив стал (или являлся с самого начала) отсортированным.

Разработанный код смотреть в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 - результаты тестирования.

№ п/п	Входные данные	Выходные данные	Комментарии
1	//Введены из файла text1: 78 4 64 8 0 -1 -9	-9 -1 0 4 8 64 78 -9 -1 0 4 8 64 78	Программа успешно выполнила сортировки массива
2	//Введены из консоли 3 4 5 6 6 6	3 4 5 6 6 6 3 4 5 6 6 6	Программа успешно выполнила сортировки массива, а точнее не проводила её вовсе, тк массив уже является отсортированным.
3	//Введено несуществующее название файла notExist.txt	No such file	Программа выдала ошибку — не существует файла с данным названием
4	//Введены из консоли: 2 9 9 9 9 9 1	1 2 9 9 9 9 9 1 2 9 9 9 9 9	Программа успешно выполнила сортировки массива
5	//Введены из консоли 42	42 42	Программа успешно выполнила сортировки массива, состоящего из одно элемента

Вывод.

В ходе выполнения лабораторной работы была создана рабочая программа с необходимым API интерфейсом, которая может считывать из файла или командой строки массив и сортируя его затем двумя сортировками, выводя при этом на экран промежуточные результаты работы сортировок. На практике были закреплены навыки по написанию таких сортировок, как сортировка пузырьком и чет-нечет сортировка.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ.

Название файла: main.cpp

```
#include <iostream>
#include <fstream>

using namespace std;

template <typename T>
bool comparator(T a, T b){
    return a>b;
}

template <typename T>
void showInfo(T arr, size_t j){
     cout<< "\""<<arr[j]<<"\", "<< j+1<<" element in array was swapped
with \""<< arr[j+1]<<"\", "<< j+2 <<" element in array (";
    cout<<"because \""<<arr[j]<< "\" > \"" << arr[j+1]<<"\")\n"<<endl;
}

template <typename T>
void bubbleSort(T arr[],size_t n, bool(*cmp)(T,T)){
    bool isSorted;
    for (size_t i = 0; i < n-1; i++)
    {
        isSorted = true;
        for (size_t j = 0; j < n-i-1; j++)
        {
            if (cmp(arr[j],arr[j+1]))
            {
                showInfo(arr,j);
                T tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
                isSorted = false;
            }
```

```
        }

        if (isSorted)
            break;
    }
}

template <typename T>
void oddEvenSort(T arr[], size_t n,bool(*cmp)(T,T)){
    bool isSorted = false;
    if(n == 1){ return;}
    while (!isSorted){
        isSorted = true;

        for (size_t i=1; i<=n-2; i=i+2){
            if (cmp(arr[i],arr[i+1])){
                 cout<<"swapping element on even position with element on
odd position"<<endl;
                showInfo(arr,i);
                swap(arr[i], arr[i+1]);
                isSorted = false;
            }
        }

        for (size_t i=0; i<=n-2; i=i+2){
            if (cmp(arr[i],arr[i+1])){
                 cout<<"swapping element on odd position with element on
even position"<<endl;
                showInfo(arr,i);
                swap(arr[i], arr[i+1]);
                isSorted = false;
            }
        }
    }
}

int main(){
    cout<<"Enter number of elements:"<<endl;
```

```
    size_t N;
    cin>>N;
    getchar();
    int array[N];
    int array2[N];
     cout<<"Choose way to enter an array:\n1 - from console\n2 - from
file"<<endl;
    switch(getchar()){
        case '1' : {
            for(size_t i = 0;i<N;i++){
                cin>> array[i];
                array2[i] = array[i];
            }
            break;
        }
        case '2' : {
            std::string filePathIn;
            std::cout << "Enter input file:\n";
            std::cin >> filePathIn;
            std::ifstream in;
            in.open(filePathIn);
            int number = 0;
            if(in.is_open()){
                for(size_t i = 0;i<N;i++){
                    in >> number;
                    array[i] = number;
                    array2[i] = number;
                }
                break;
            } else cout<<"No such file"<<endl; return 1;
        }
        default: cout<<"You haven't chose anything" <<endl; return 1;
    }

    cout<< "Bubble sort:\n ---------------------------------"<<endl;
    bubbleSort(array,N,comparator);
    cout<<"\nsorted array: ";
    for(int i : array) cout<<i<< ' ';
    cout<< "\n---------------------------------\n\n\n"<<endl;
```

```
    cout<<"Odd even sort:\n----------------------------------"<<endl;
    oddEvenSort(array2,N,comparator);
    cout<<"\nsorted array: ";
    for(int i : array2) cout<<i<< ' ';
    cout<< "\n----------------------------------"<<endl;

}
```

Название файла: Makefile

```
all: goal
	./start

goal: main.o
	g++ main.o -o start

main.o: main.cpp
	g++ -c main.cpp

clean:
	rm -f *.o
```