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. 9304 _____ Боблаков Д.С.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

Цель работы

Научиться реализовывать сортировки разных видов. Получить навыки работы с сортировками.

Задание

Вариант 4.

Пузырьковая сортировка оптимизированная; сортировка чёт-нечет.

Описание алгоритма работы

Сначала программа считывает строку str, включая пробелы. После чего программа проверяет корректность введенных данных. Если введенные данные некорректны, программа выведет сообщение об ошибке и завершит свою работу. Если введенные данные корректны, то программа преобразует строку в вектор и будет далее работать только с ним. Затем создаются еще две копии исходного вектора для работы с разными сортировками. После этого к полученным векторам по отдельности применяются следующие сортировки: std::sort(), BubleSort (Пузырьковая сортировка), oddEvenSort (сортировка чет-нечет). Во время выполнения алгоритмов сортировки происходит отслеживание состояния векторов, а также происходит нотация работы алгоритмов. Затем выводятся результаты, которые сравниваются с эталонным результатом от std::sort(). В случае совпадении результат сортировки с эталонным, будет выведено соответствующее сообщение, в ином случае будет выведено сообщение о несовпадении результатов сортировки.

Алгоритм пузырьковой сортировки: данный алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены

больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива. Оптимизация данного алгоритма заключается в уменьшении пути прохождения по массиву каждый раз на единицу, так как этот алгоритм дает гарантию, что последний из сравниваемых элементов на каждой итерации будет на своем искомом месте. Также, если в массиве не окажется не отсортированных элементов, алгоритм досрочно завершит работу.

Алгоритм сортировки чет-нечет: данный алгоритм является модификацией алгоритма пузырьковой сортировки. Его отличие заключается в том, чтобы сравнивать элементы массива под чётными и нечётными индексами с последующими элементами независимо.

Разработанный код см. в приложении А.

Формат входных и выходных данных

Программа принимает на вход строку формата <a b … n>, где на месте букв находятся числа, между которыми находятся пробелы.

На выходе программа выведет нотацию сортировок, их результат и результат сравнения с эталоном в случае корректного ввода данных, в ином случае программа выведет следующую строку: «Error: incorrect value».

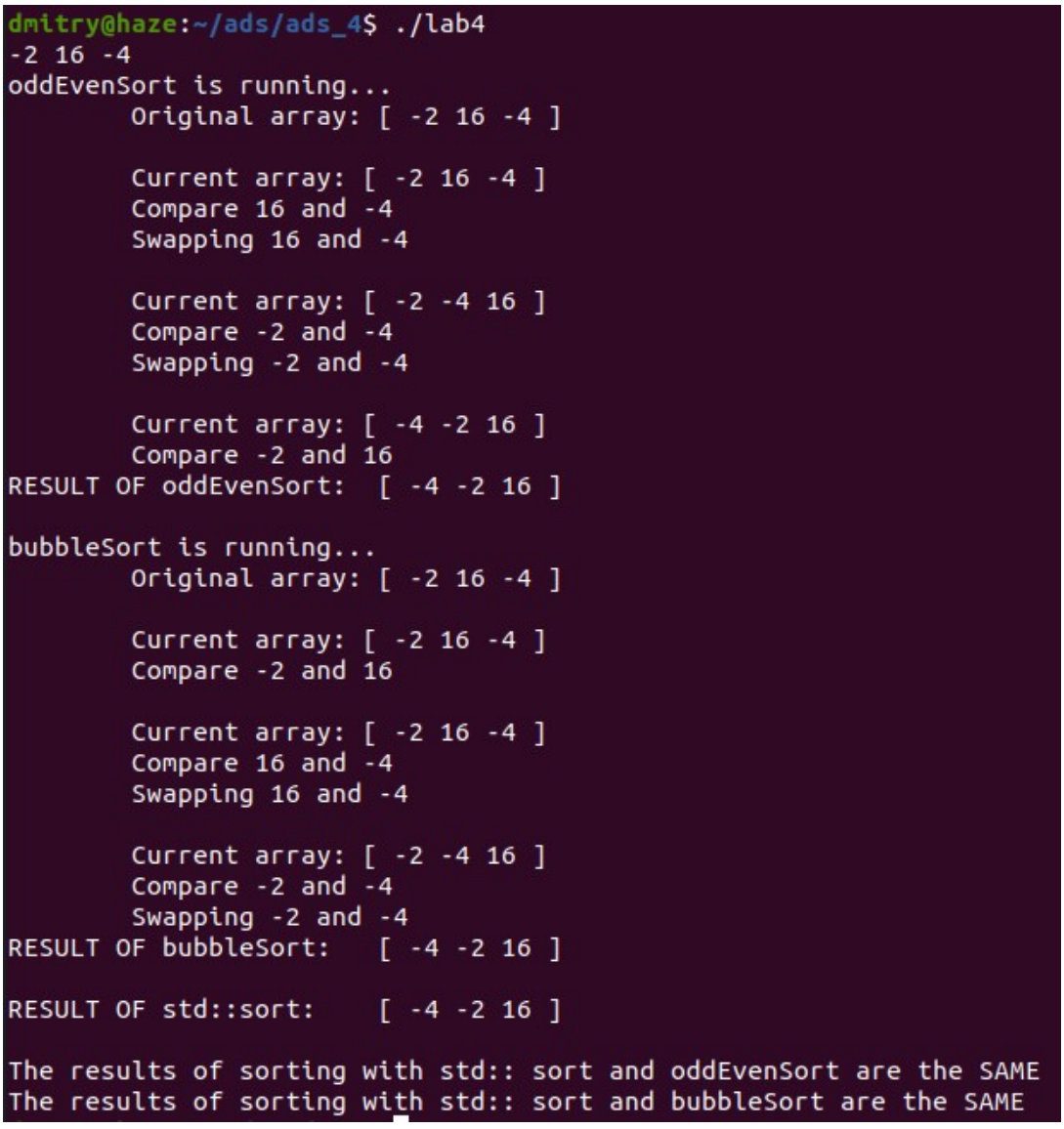

Рисунок 1 – Пример запуска програмы.

Описание основных структур данных и функций

Bool isCorrect(const std::string& str)

Данная функция принимает строку str и проверяет ее на корректность.

void print(std::vector<T>& vector)

Данная функция печатает переданный ей вектор.

void oddEvenSort(std::vector<T>& vector)

Данная функция принимает вектор и сортирует его по алгоритму чет-нечет.

void bubbleSort(std::vector<T>& vector)

Данная функция принимает вектор и сортирует его по алгоритму пузырьковой сортировки.

bool isEqual(std::vector<T> vector1, std::vector<T> vector2)

Данная функция сравненивает на идентичность два вектора.

void get_numbers(std::vector<int> & result, const std::string & str)

Данная функция преобразует строку в вектор.

Тестирование

Тестирование программы проводится с помощью bash-скрипта tests_script. Для запуска тестирования необходимо выполнить команду ./tests_scripts, предварительно собрав программу с помощью команды make.

Для тестирования было написано 10 тестов, первые 7 из которых являются корректными для программы. Последние 3 теста являются некорректными данными для программы.

Индикатором успешного выполнения тестирования служит сообщение программы о равных результатах сортировки или при совпадении ожидания ошибки и ее появления. Иные случаи свидетельствуют о некорректности работы программы.

Результаты тестирования см. в приложении Б.

Выводы

Были изучены разные виды сортировок и их устройство. Были приобретены практические навыки для работы с сортировками.

Была реализована программа на языке программирования C++, осуществляющая сортировку массива двумя видами алгоритмов сортировки, вывод состояния массива на каждом шаге работы алгоритмов, а также

сравнение результатов с результатом эталонного алгоритма. Проведено тестирование работоспособности программы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: main.cpp

```
#include <iostream>
##include <iostream>
#include <algorithm>
#include <vector>
//#include <array>

template <typename T>
void print(std::vector<T>& vector){
    std::cout<<"[ ";
    for (auto i : vector) {
        std::cout<<i<<" ";
    }
    std::cout<<"]";
    std::cout<<"\n";
}

template <typename T>
void oddEvenSort(std::vector<T>& vector){

    std::cout<<"oddEvenSort is running... \n";
    std::cout<<"\tOriginal array: ";
    print(vector);

    for (size_t i = 0; i < vector.size(); i++){
        for (size_t j = (i % 2) ? 0 : 1; j + 1 < vector.size(); j
+= 2){        // (i % 2) ? 0 : 1 возвращает 1, если i четное, 0, если
i не четное

            std::cout<<"\n\tCurrent array: ";
            print(vector);
```

```
                std::cout<<"\tCompare "<<vector[j]<<" and
"<<vector[j+1]<<"\n";
                if (vector[j] > vector[j + 1]) {
                    std::cout<<"\tSwapping "<<vector[j]<<" and
"<<vector[j+1]<<"\n";
                    std::swap(vector[j], vector[j + 1]);
                }
            }
        }
    }

    template <typename T>
    void bubbleSort(std::vector<T>& vector){

        std::cout<<"bubbleSort is running... \n";
        std::cout<<"\tOriginal array: ";
        print(vector);

        bool swapped;
        for (int i = 0; i < vector.size()-1; i++){
            swapped = false;
            for (int j = 0; j < vector.size()-i-1; j++){

                std::cout<<"\n\tCurrent array: ";
                print(vector);

                std::cout<<"\tCompare "<<vector[j]<<" and
"<<vector[j+1]<<"\n";
                if (vector[j] > vector[j+1]){
                    std::cout<<"\tSwapping "<<vector[j]<<" and
"<<vector[j+1]<<"\n";
                    std::swap(vector[j], vector[j+1]);
                    swapped = true;
                }
            }
```

```
            if (!swapped) {          // Если в процессе прохода не
было ни одной замены, то выход из функции
                break;
            }
        }
    }

    template <typename T>
    bool isEqual(std::vector<T> vector1, std::vector<T> vector2){
        return std::equal(vector1.begin(), vector1.end(),
vector2.begin());
    }

    bool isCorrect(const std::string& str){

        if (str.empty()){
            return false;
        }
        for(char i:str){
            if (!isdigit(i) && i != ' ' && i != '-'){
                return false;
            }
        }
        return true;
    }

    void get_numbers(std::vector<int> & result, const std::string &
str) {
        bool found = false;
        int number = 0;
        bool negative = false;

        for (char ch : str) {
            if (ch >= '0' && ch <= '9' || ch == '-') {
                if (ch =='-'){
```

```
                negative= true;
                continue;
            }
            const int digit = ch - '0';
            number = number*10 + digit;
            found = true;
        }
        else {
            if (found) {
                if(negative){
                    number=number*(-1);
                    negative= false;
                }
                result.push_back(number);

                number = 0;
                found = false;
            }
        }
    }

    if (found) {
        if(negative){
            number=number*(-1);
        }
        result.push_back(number);
    }
}

int main(){

    std::string str;
    getline(std::cin, str);
    if(!isCorrect(str)){
        std::cout<<"Error: incorrect value";
```

```
            return EXIT_FAILURE;
        }
        std::vector<int> vector;

        get_numbers(vector,str);
        std::vector<int> vector1 = vector;
        std::vector<int> vector2 = vector;

        oddEvenSort(vector1);
        std::cout<<"RESULT OF oddEvenSort:\t";
        print(vector1);
        std::cout<<"\n";

        bubbleSort(vector2);
        std::cout<<"RESULT OF bubbleSort:\t";
        print(vector2);
        std::cout<<"\n";

        std::sort(vector.begin(), vector.end());
        std::cout<<"RESULT OF std::sort: \t";
        print(vector);
        std::cout<<"\n";

        if (isEqual(vector,vector1)){
            std::cout<<"The results of sorting with std:: sort and
oddEvenSort are the SAME\n";
        } else std::cout<<"The results of sorting with std:: sort and
oddEvenSort are DIFFERENT\n";

        if (isEqual(vector,vector2)){
            std::cout<<"The results of sorting with std:: sort and
bubbleSort are the SAME\n";
        } else std::cout<<"The results of sorting with std:: sort and
bubbleSort are DIFFERENT\n";
```

```
    return 0;
}
```

Файл: Makefile

```
	g++ -std=c++17 Source/main.cpp -o lab4

run_tests: lab4
	./tests_script
```

Файл: tests_scripts.sh

```
#! /bin/bash

printf "\nLaunching the tests\n\n"
for n in {0..6}
do
    printf "Test$n:\n"
    ./lab4 < "./Tests/test$n.txt"
    printf "\n"
done
for n in {7..9}
do
	printf "Test$n:\n"
	printf "Error expected\nResult is\t"
	./lab4 < "./Tests/wrong_test$n.txt"
	printf "\n"
done
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Результаты тестирования представлены в таблице Б.1

Таблица Б.1 — Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	-1 0 1 2 3 4 5 6 7 8 9	SAME SAME	Уже отсортированный массив
2.	00 0 -12 -24 8987 -981	SAME SAME	Обычный массив с двумя нулями
3.	-94 -1 -51 -89 -63 215 11 -99	SAME SAME	Обычный массив
4.	900 0 27 -94 114 35 -25 -41	SAME SAME	Обычный массив
5.	2000 -15 -15 89 -15 -15 64 48 000	SAME SAME	Проверка на сортировку одинаковых значений
6.	-0 0 124 523 -627 1000	SAME SAME	Проверка корректности ввода 0 и -0
7.	100 90 81 79 64 51 49 36 25 11 1 -4 -24 -29	SAME SAME	Массив реверсивно отсортированный изначально
8.	-921 a 34 321 -21	Error: incorrect value	Некорректный символ
9.		Error: incorrect value	Пустая строка
10.	BARNAUL, SAVE AMERICA KOSTROMA, SAVE RUSSIA	Error: incorrect value	Некорректная строка