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: Методы сортировок

Студент гр. 9303 _____ Дюков В. А.

Преподаватель _____ Филатов А. Ю.

Санкт-Петербург

2020

Цель работы.

Написанить функцию оптимизированной сортировки пузырьком – чёт-нечёт.

Задание.

Вариант 4.

Пузырьковая сортировка оптимизированная; сортировка чёт-нечет.

Основные теоретические положения.

Сортировка чёт-нечёт в основном разновидность пузырьковой сортировки. Этот алгоритм делится на две фазы: нечетная и четная. Алгоритм работает до тех пор, пока элементы массива не будут отсортированы, и в каждой итерации происходит две фазы: нечетная и четная.

В нечетной фазе мы выполняем пузырьковую сортировку по нечетным индексированным элементам, а в четной фазе мы выполняем пузырьковую сортировку по четным индексированным элементам.

Выполнение работы.

Для выполнения работы была реализована шаблонная функция *even_odd_sort* осуществляющая сортировку чёт-нечёт элементов вектора в порядке возрастания. Данная функция принимает вектор, который надо отсортировать. В цикле *while* сначала идёт сравнение пар чётных – нечётных элементов вектора, затем сравнение пар нечётных – чётных элементов. Если элемент с индексом « i » оказывается больше чем « i + 1 » , то производится их замена местами. Цикл завершается, если в его теле не происходит ниодной замены элементов вектора. Функция возвращает отсортированый вектор элементов.

В теле функции вызываются такие шаблонные функции как:

- *even_print* – выводит в поток вывода вектор с обозначениями элементов которые нужно заменить при чётной замене.
- *odd_print* – выводит в поток вывода вектор с обозначениями элементов которые нужно заменить при нечётной замене.

Данные функции нужны для подробной демонстрации работы функции *even_odd_sort*.

Тестирование

№ т.	Входные данные	Результат	Вывод

1	`-1 3 5 2 77 8 0`	`Исходный массив 2:`
		`-1 3 5 2 77 8 0`
		
		`Нечётное 1: -1  3--5  2--77  8--0`
		`-1  3--5  2--77  0--8`
		
		`Чётное  1: -1--3  5--2  77--0  8`
		`-1--3  2--5  0--77  8`
		
		`Нечётное 2: -1  3--2  5--0  77--8`
		`-1  2--3  0--5  8--77`
		
		`Чётное  2: -1--2  3--0  5--8  77`
		`-1--2  0--3  5--8  77`
		
		`Нечётное 3: -1  2--0  3--5  8--77`
		`-1  0--2  3--5  8--77`
		
		`Чётное  3: -1--0  2--3  5--8  77`
		`-1--0  2--3  5--8  77`
		
		`Нечётное 4: -1  0--2  3--5  8--77`
		`-1  0--2  3--5  8--77`
		
		`Чётное  4: -1--0  2--3  5--8  77`
		`-1--0  2--3  5--8  77`
		
		`Результат:`
		`-1 0 2 3 5 8 77`
			Программа работает верно

2	`1 2 3 4`	`Исходный массив 3:` `1 2 3 4`	Программа работает верно

		Нечётное 1: 1 2--3 4 1 2--3 4 Чётное 1: 1--2 3--4 1--2 3--4 Результат: 1 2 3 4	
3	b a e g f c d	Исходный массив 2: b a e g f c d d Нечётное 1: b a--e g--f c--d d b a--e f--g c--d d Чётное 1: b--a e--f g--c d--d a--b e--f c--g d--d Нечётное 2: a b--e f--c g--d d a b--e c--f d--g d Чётное 2: a--b e--c f--d g--d a--b c--e d--f d--g Нечётное 3: a b--c e--d f--d g a b--c d--e d--f g Чётное 3: a--b c--d e--d f--g a--b c--d d--e f--g Нечётное 4: a b--c d--d e--f g a b--c d--d e--f g Чётное 4: a--b c--d d--e f--g a--b c--d d--e f--g Результат: a b c d d e f g	Программа работает верно
4	f e d c b a	Исходный массив 1: f e d c b a a Нечётное 1: f e--d c--b a--a f d--e b--c a--a Чётное 1: f--d e--b c--a a d--f b--e a--c a	Программа работает верно

		Нечётное 2: d f--b e--a c--a d b--f a--e a--c Чётное 2: d--b f--a e--a c b--d a--f a--e c Нечётное 3: b d--a f--a e--c b a--d a--f c--e Чётное 3: b--a d--a f--c e a--b a--d c--f e Нечётное 4: a b--a d--c f--e a a--b c--d e--f Чётное 4: a--a b--c d--e f a--a b--c d--e f Нечётное 5: a a--b c--d e--f a a--b c--d e--f Чётное 5: a--a b--c d--e f a--a b--c d--e f Результат: a a b c d e f	

Выводы

Были изучены теоретические материалы по методам сортировки данных, в частности методу чёт-нечёт. Тип, сортируемых сделан шаблонным, однако функция сможет работать только с тем типом, для которого определены операторы сравнения и вывода.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

main.cpp:

```
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include <vector>
#include <conio.h>
#include <Windows.h>

#define HEAD "Демонстрация работы функции сортировки пузырьком 'четно-нечетно'\nДюков
Владимир гр. 9303\n\n"
#define FICO "Выберете способ ввода массива:\n\n"
#define COIN "Введите массив, которые надо отсортировать:\n\n"
#define FIIN "Введите имя файла:\n\n"
#define FIER "Ошибка ввода файла\n\n"

template<typename T>
void even_print(std::vector<T> data, std::ostream& out) {

    int n = (int)data.size();
    for (int i = 0; i < n; i++) {

        if ((i + 1) % 2 && i != n - 1) out << data[i] << "--";
        else out << data[i] << "  ";
    }
}

template<typename T>
void odd_print(std::vector<T> data, std::ostream& out) {

    int n = (int)data.size();
    for (int i = 0; i < n; i++) {

        if (i % 2 && i != n - 1) out << data[i] << "--";
        else out << data[i] << "  ";
    }
}

template<typename T>
std::vector<T> even_odd_sort(std::vector<T> data, std::ostream& out) {

    int n = (int)data.size();
    bool isSorted = 0;
    int k = 1;

    while (!isSorted) {
        isSorted = 1;
```

```
        out << "\nНечётное " << k << ":\t";
        odd_print(data, out);

        for (int i = 1; i < n - 1; i += 2) {
            if (data[i] > data[i + 1]) {
                T temp = data[i];
                data[i] = data[i + 1];
                data[i + 1] = temp;
                isSorted = 0;
            }
        }

        out << "\n\t\t";
        odd_print(data, out);

        out << "\n\nЧётное   " << k << ":\t";
        even_print(data, out);

        for (int i = 0; i < n - 1; i += 2) {
            if (data[i] > data[i + 1]) {
                T temp = data[i];
                data[i] = data[i + 1];
                data[i + 1] = temp;
                isSorted = 0;
            }
        }

        out << "\n\t\t";
        even_print(data, out);
        out << '\n';
        k++;
    }
    return data;
}

template<typename T>
std::ostream& operator<<(std::ostream& out, std::vector<T> data) {

    int n = (int)data.size();
    for (int i = 0; i < n; i++) out << data[i] << " ";
    return out;
}

int button_get(std::string* buttons, int size) {

    int count = 0;

    while (1) {

        std::cout << '\r';

        for (int i = 0; i < size; i++) {

            if (i == count) std::cout << '<' << buttons[i] << '>' << '\t';
```

```
            else std::cout << ' ' << buttons[i] << ' ' << '\t';
        }

        unsigned char key = _getch();
        if (key == 224) key = _getch();

        switch (key) {
        case 75:
            count--;
            break;
        case 77:
            count++;
            break;
        case 13:
            return count + 1;
        case 27:
            return 0;
        }
        if (count > size - 1) count = 0;
        if (count < 0) count = size - 1;
    }
}

typedef int vect_type;

int main() {

    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    std::string ficoget[2] = { "Консоль", "Файл" };
    std::ofstream outfile("result.txt");

    while (1) {

        std::stringstream inputText("");
        int test_num = 0;

        system("cls");
        std::cout << HEAD;
        std::cout << FICO;

        int check_1 = button_get(ficoget, 2);
        system("cls");
        if (!check_1) break;
        if (check_1 == 1) {

            std::cout << HEAD;
            std::cout << COIN;
            std::string str;
            std::getline(std::cin, str);
            inputText << str;
            test_num++;
        }
```

```
        if (check_1 == 2) {

            std::cout << HEAD;
            std::cout << FIIN;
            std::string name;
            std::cin >> name;
            std::ifstream file(name);
            if (!file.fail()) {

                while (!file.eof()) {

                    std::getline(file, name);
                    inputText << name << '\n';
                    test_num++;
                }
                file.close();
            }
            else {

                std::cout << '\n' << FIER;
                system("pause");
                continue;
            }
        }

        inputText.seekg(0, inputText.beg);

        for (int i = 0; i < test_num; i++) {

            std::stringstream inputLine;
            std::vector<vect_type> arr;
            std::string str;
            std::getline(inputText, str);
            inputLine << str;

            while (!inputLine.eof()) {

                vect_type var;
                inputLine >> var;
                arr.push_back(var);
            }

            system("cls");
            std::cout << HEAD;
            std::cout << "Исходный массив " << i + 1 << ":\n" << arr << "\n\n\n";
            std::cout << "\nРезультат:\n" << even_odd_sort(arr, std::cout) << "\n\n";
            if (outfile.is_open()) outfile << "Исходный массив " << i + 1 << ":\n" <<
arr << "\n\n\n";
            if (outfile.is_open()) outfile << "\nРезультат:\n" << even_odd_sort(arr,
outfile) << "\n\n\n";
            system("pause");
        }
    }
```

```
    if (outfile.is_open()) outfile.close();

    return 0;
}
```