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. 9382 _____ Иерусалимов.Н

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Разобраться в понятии и применении сортировки данных. Составить оценку её достоинств и недостатков.

Задание.

Вариант 4.

4. Пузырьковая сортировка оптимизированная; сортировка чёт-нечет.

Описание основных функций.

void printDep(int depth)

Назначение: Выводит глубину работы алгоритма.

Описание аргументов: Глубина погружения сортировки.

void CheckTheInput(string arr, bool &Exit, bool choise)

Назначение: Проверяет массив на соответствие сортировке.

Описание аргументов: Массив хронящий элементы, ссылку на выход из цикла, и выбор который определяет из файла данные или из консоли.

template<typename T> void oddEvenSorting(T *array, size_t N)

Назначение: Производит сортировку элементов массива.

Описание аргументов: Указатель на массив, хранящий элементы произвольного типа, количество элементов в массиве.

Возвращаемое значение: Отсортированный вектор с элементами произвольного типа.

Описание алгоритма.

После считывания массива и отлова ошибок на подобии пустого массива или одного элемента в нем запускается алгоритм сортировки,

Выглядит это так:

1. Первым делом заводим переменные, отвечающие за глубину “рекурсии”.
2. Запускаем цикл который идет до N. Внутри этого цикла прибавляем 1 глубине рекурсии и заводим переменную size_t j, которая отвечает за разбиение массива на части которые будем сравнивать.

   “j” определяется по формуле “(i%2) ? 0 : 1” возвращает 1, если i четное, 0, если i не четное. Где i счетчик первого цикла. После чего запускаем второй цикл где счетчиком является “j” и он идет до j + 1<N; и на каждом шаге прибавляет +2 к j. Благодаря этому цикл, мы разбиваем массив на группы элементов под четными/нечетными индексами где проверяется элемент под четным индексом, больше ли он следующего элемента(нечетного), если больше то элементы меняются местами. После отработки с четными/нечетными цикл завершается, делает сдвиг путем переопределением “j” и начинает работать с нечетными/ четными индексами, проделывая тоже самое что было описано выше. После того как массив отсортирован, управление возвращается той функции от куда вызвали сортировку.

Алгоритм представляет вариацию алгоритма пузырьковой сортировки. В отличие от "пузырька", где на i -м проходе первые i элементов занимают свои места, в алгоритме "чет - нечет" элементы гарантировано занимают свои места после выполнения всех n проходов.
Для самого легкого элемента достаточно n - 1 проход для "всплытия" в вершину массива, так как на каждом проходе элемент поднимается вверх на одну позицию. Для следующего за ним элемента может понадобиться в самом неблагоприятном случае ровно N проходов.

Достоинство этого алгоритма в том, что итерации внутренних циклов независимы и потому допускают естественное распараллеливание.

Пример работы программы.

Таблица 1 – Пример работы

Входные данные	Выходные данные
961	Input data or '!' to quit: 961 Amount of elements: 3 What indexes can we go by: 1) even/odd 2) odd/even Sorting start! _____ /\Split array into *odd/even* subgroups by index /\array[1] > array[2]: 6 > 1 True /\array{9, 6, 1, } /\swap: array[1] -> array[2] /\new array{9, 1, 6, } /\/\Split array into *even/odd* subgroups by index /\/\array[0] > array[1]: 9 > 1 True /\/\array{9, 1, 6, } /\/\swap: array[0] -> array[1] /\/\new array{1, 9, 6, } /\/\/\Split array into *odd/even* subgroups by index /\/\/\array[1] > array[2]: 9 > 6 True /\/\/\array{1, 9, 6, } /\/\/\swap: array[1] -> array[2] /\/\/\new array{1, 6, 9, } Sorted array is: 169 _____

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 — Результаты тестирования

№	Входные данные	Выходные данные	Доп.Сообщения
1.	1	1	In array one element : 1
2.	fg35df	35dffg	Amount of elements: 6
3.	954v19563	13455699v	Amount of elements: 9
4.	“Enter”		Empty array!
5.	qwertyuiopasdfghjklzxcvbnm	abcdefghijklmnopqrstuvwxyz	Amount of elements: 26
6.	n\	\n	Amount of elements: 2
7.	стрпо	опрст	Amount of elements: 5

Сортировка библиотекой <std::sort>

№	Входные данные	Выходные данные
1.	1	1
2.	fg35df	35dffg
3.	954v19563	13455699v
4.	“”	empty
5.	qwertyuiopasdfghjklzxcvbnm	abcdefghijklmnopqrstuvwxyz
6.	n\	\n
7.	стрпо	опрст

Выводы.

Получены знания в области работы с сортировкой чет/нечет. Написана работающая программа на языке C++, способная сортировать элементы, поданные на вход. Оценена оптимальность работы с этим видом совтировки.

ПРИЛОЖЕНИЕ С КОДОМ

main.cpp :

```
#include <iostream>
#include<string>
#include <fstream>
#include <conio.h>

using namespace std;
void printDep(int depth){
    for(int i = 0; i < depth; ++i){
        cout<<"/\\";
    }
}
/*
В этой сортировке мы делим данные на четные и нечетные индексы, а затем
сравниваем их.
 Четные строго со следующим числом, т.е нечетным.
```

```
 После чего делаем сдвиг и сравниваем уже нечетные/четные.
*/
template < typename T>
void oddEvenSorting(T &array, size_t N) {
    cout<<"\nSorting
start!\n_____________________________________________________________\n";

    int depth = 0;
    for (size_t i = 0; i < N; i++) {
        ++depth;
        // (i % 2) ? 0 : 1 возвращает 1, если i четное, 0, если i не четное
        size_t j = (i % 2) ? 0 : 1;
        bool split = (i % 2) ? 0 : 1;
        for (; j + 1 < N; j += 2) {

            if (array[j] > array[j + 1]) {
                printDep(depth);
                if(!split){
                    cout<< "Split array into *even/odd* subgroups by index\n";
                }else {
                    cout<< "Split array into *odd/even* subgroups by index\n";
                }

                printDep(depth);
                cout<<"array["<<j<<"] > array["<<j+1<<"]: "<< array[j] <<" >
"<<array[j + 1]<<" True\n";

                printDep(depth);
                cout<<"array{";
                for(int out =0; out<N;++out){
                    cout<<array[out]<<", ";
                }
                cout<<"}";
                cout<<'\n';
                std::swap(array[j], array[j + 1]);
                printDep(depth);
                cout<<"swap: array["<<j<<"] -> array["<<j+1<<"]\n";
                printDep(depth);
                cout<<"new array{";
                for(int out =0; out<N;++out){
                    cout<<array[out]<<", ";
                }
                cout<<"}";
                cout<<'\n';
                cout<<'\n';
            }else{
                printDep(depth);
                if(!split){
                    cout<< "Split array into *even/odd* subgroups by index\n";
                }else {
                    cout<< "Split array into *odd/even* subgroups by index\n";
                }
                printDep(depth);
                cout<<"array["<<j<<"] > array["<<j+1<<"]: "<< array[j] <<" >
"<<array[j + 1]<<" Lie\n";
                printDep(depth);
                cout<<"move on!\n\n";
            }
        }
    }
}

void writeToFile(const string filename, const string arg) {
    ofstream output;
    output.open(filename, ios::app);
    output << arg;
    output.close();
}
```

```
bool CheckTheInput(string arr, bool &Exit, bool choise){
    if (arr.length() == 1 && arr[0] == '!') {
        Exit = false;
    } else if (arr.length() == 1) {
        cout << "In array one element : " << arr << "\n\n\n";
    } else if (arr.length() == 0) {
        cout << "\nAmount of elements: " << arr.length() << "\n";
        cout << "Empty array!\n\n\n";
    } else {
        cout << "\nAmount of elements: " << arr.length() << '\n';
        cout << "What indexes can we go by:\n1) even/odd\n2) odd/even\n";
        oddEvenSorting(arr, arr.length());
        if(!choise) {
            writeToFile("output.txt", "\nSorted array is: ");
            writeToFile("output.txt", arr);
        }
        cout << "Sorted array is: " << arr <<
"\n_____________________________________________\n\n";
    }
}

int main() {
    bool Exit = true;
    bool choise = true;
    cout << "Enter : 0 - File Input, 1 - Console input: ";
    cin>>choise;
    cin.ignore();
    cout<<'\n';
    if(!choise){
         //-----Ввод с файла-----//
        string input_filename;
        const string output_filename = "output.txt";
        ifstream in;
        ofstream out;

        out.open(output_filename);
        out << "";
        out.close();

        cout << "Enter the input file name: \n\n";
        cin >> input_filename;
        in.open(input_filename);

        if (in.is_open()) {
            string s = "";

            getline(in, s);
            writeToFile("output.txt", "Initial array: ");
            writeToFile("output.txt", s);
            CheckTheInput(s,Exit,choise);

        }
        else {
            cout << input_filename << " doesn't exist!\n";
        }

    }else {
        string arr = "";
        while (Exit) {
            cout << "Input data or \'!\' to quit: ";
            getline(cin, arr);
            CheckTheInput(arr, Exit,choise);
        }
    }
    getch();
    return 0;
```

```
}
```