**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: Сортировка

|  |  |  |
| --- | --- | --- |
| Студент гр. 9304 |  | Аксёнова Е.А. |
| Преподаватель |  | Филатов Ар.Ю. |

Санкт-Петербург

2020

## Цель работы.

## Изучить алгоритмы сортировок. Реализовать программу на языке С++, сортирующую массив данных, с помощью одного из алгоритмов сортировки.

## Задание.

Вариант 19.

## Реализовать алгоритм сортировки расчёской.

## Описание алгоритм работы.

Сначала производится считывание массива данных в строку. Затем вызывается функция checkStr(), которая проверяет строку на валидность, если попадаются элементы не подходящие под условие, то она их просто удаляет. Объявляем проверенную строку потоком и считываем из нее элементы в вектор, пропуская все знаки пробела. После считывания вектора, мы записываем его значение в ещё один вектор, который предназначен для проверки и вызываем функцию, для сортировки sortc().

В функции сортировки создаются три целочисленные переменные: size – для определения размера массива, count – для определения номера итерации, step – переменная для шага сортировки. Затем, пока шаг больше единицы, записываем в переменную step максимальное из чисел 1, step/= 1.247f, и пробегаемся по массиву, сравнивая элементы находящиеся в шаге друг от друга. Если левый больше правого, то мы их меняем.

Сортируем этот же массив с помощью std::sort и сравниваем полученные значения.

Разработанный программный код см. в приложении А.

**Формат входных и выходных данных.**

На вход программе подается строка, в которой через пробелы указаны элементы сортируемого массива.

Программа выводит отсортированный массив чисел.

**Описание основных структур данных и функций.**

* void sortc() – функция сортирующая массив алгоритмом расчёска.
* void checkStr() – функция проверяющая валидность строки
* operator << - перегруженный оператор для вывода вектора

**Тестирование.**

Запуск программы начинается с ввода команды “make”, что приведёт к компиляции и линковки программы, после чего будет создан исполняемый файл lab4. Тестирование производится посредством bash-скрипта. Далее прописываем команду chmod +x test\_scr. А затем прописываем “./test\_scr”, что приводит к запуску тестирования.

## Выводы.

Реализована программа на языке С++, сортирующая массив поданных данных с помощью алгоритма сортировки расчёской. Сложность данной сортировки в лучшем случае – n\*log(n), а в худшем – n2. Имеет преимущества над сортировкой пузырьком, ввиду того, что выполняется быстрее в большинстве случаев.

# Приложение А Исходный код программы

Название файла: sort.cpp

#include <iostream>

#include <algorithm>

#include <string>

#include <sstream>

#include <vector>

std::ostream& operator << (std::ostream& out, const std::vector<int>& vec) {

out << '[';

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

if (i == vec.size() - 1) {

out << vec[i];

break;

}

out << vec[i] << ", ";

}

out << ']';

return out;

}

void sortc(std::vector<int>& vec, int size){

int tmp;

int count = 0;

int step = size;

while (step > 1) {

count += 1;

std::cout << "This is " << count << " iteration of sorting" << '\n';

step = std::max(1, step/= 1.247f);

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

if (vec[i] > vec[i + step] ) {

std::cout << "We will change this elements: " << vec[i] << "(" << i << ")" << " and " << vec[i + step] << "(" << i + step << ")" << '\n';

std::swap(vec[i], vec[i + step]);

}

}

std::cout << vec << '\n';

}

}

void checkStr(std::string& str) {

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

if (!isdigit(str[i])&&str[i]!=' ') {

str.erase(i, 1);

i -= 1;

}

}

}

int main(int argc, char\*\* argv) {

std::string str1;

std::vector<int> vec;

std::vector<int> vecToCheck;

int value;

if (argc < 2)

std::getline(std::cin, str1);

else

str1 = argv[1];

checkStr(str1);

std::stringstream ss(str1);

while (ss >> value) {

vec.push\_back(value);

if (ss.peek() == ' ') {

ss.ignore();

}

}

vecToCheck = vec;

sortc(vec, vec.size());

std::cout << "This is sorted array" << '\n';

std::cout << vec;

std::cout << '\n';

std::sort(vecToCheck.begin(), vecToCheck.end());

if (vec == vecToCheck) {

std::cout << "They are identical" << std::endl;

}

else {

std::cout << "Incorrect!" << std::endl;

}

return 0;

}

# Приложение Б Тестирование

Результаты тестирования представлены в таблице Б.1

Таблица Б.1 — Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Результат проверки |
| 1. | 12 1 30 24 | This is 1 iteration of sorting  [12, 1, 30, 24]  This is 2 iteration of sorting  [12, 1, 30, 24]  This is 3 iteration of sorting  We will change this elements: 12(0) and 1(1)  We will change this elements: 30(2) and 24(3)  [1, 12, 24, 30]  This is sorted array  [1, 12, 24, 30] | They are identical |
| 2. | 6 32 24 20 54 8 44 26 1 10 22 34 12 16 18 45 | This is 1 iteration of sorting  We will change this elements: 32(1) and 16(13)  We will change this elements: 24(2) and 18(14)  [6, 16, 18, 20, 54, 8, 44, 26, 1, 10, 22, 34, 12, 32, 24, 45]  This is 2 iteration of sorting  We will change this elements: 20(3) and 12(12)  We will change this elements: 54(4) and 32(13)  [6, 16, 18, 12, 32, 8, 44, 26, 1, 10, 22, 34, 20, 54, 24, 45]  This is 3 iteration of sorting  We will change this elements: 16(1) and 1(8)  We will change this elements: 18(2) and 10(9)  We will change this elements: 26(7) and 24(14)  [6, 1, 10, 12, 32, 8, 44, 24, 16, 18, 22, 34, 20, 54, 26, 45]  This is 4 iteration of sorting  We will change this elements: 32(4) and 18(9)  We will change this elements: 44(6) and 34(11)  We will change this elements: 24(7) and 20(12)  We will change this elements: 32(9) and 26(14)  [6, 1, 10, 12, 18, 8, 34, 20, 16, 26, 22, 44, 24, 54, 32, 45]  This is 5 iteration of sorting  We will change this elements: 18(4) and 16(8)  We will change this elements: 34(6) and 22(10)  We will change this elements: 34(10) and 32(14)  [6, 1, 10, 12, 16, 8, 22, 20, 18, 26, 32, 44, 24, 54, 34, 45]  This is 6 iteration of sorting  We will change this elements: 10(2) and 8(5)  We will change this elements: 26(9) and 24(12)  We will change this elements: 44(11) and 34(14)  [6, 1, 8, 12, 16, 10, 22, 20, 18, 24, 32, 34, 26, 54, 44, 45]  This is 7 iteration of sorting  We will change this elements: 12(3) and 10(5)  We will change this elements: 22(6) and 18(8)  We will change this elements: 32(10) and 26(12)  We will change this elements: 54(13) and 45(15)  [6, 1, 8, 10, 16, 12, 18, 20, 22, 24, 26, 34, 32, 45, 44, 54]  This is 8 iteration of sorting  We will change this elements: 6(0) and 1(1)  We will change this elements: 16(4) and 12(5)  We will change this elements: 34(11) and 32(12)  We will change this elements: 45(13) and 44(14)  [1, 6, 8, 10, 12, 16, 18, 20, 22, 24, 26, 32, 34, 44, 45, 54]  This is sorted array  [1, 6, 8, 10, 12, 16, 18, 20, 22, 24, 26, 32, 34, 44, 45, 54] | They are identical |
| 3. | -45 3 18 11 2 -7 ! | This is 1 iteration of sorting  We will change this elements: 45(0) and 2(4)  [2, 3, 18, 11, 45, 7]  This is 2 iteration of sorting  We will change this elements: 18(2) and 7(5)  [2, 3, 7, 11, 45, 18]  This is 3 iteration of sorting  [2, 3, 7, 11, 45, 18]  This is 4 iteration of sorting  We will change this elements: 45(4) and 18(5)  [2, 3, 7, 11, 18, 45]  This is sorted array  [2, 3, 7, 11, 18, 45] | They are identical |
| 4. | 23 a 56 89 12 3 | This is 1 iteration of sorting  We will change this elements: 23(0) and 3(4)  [3, 56, 89, 12, 23]  This is 2 iteration of sorting  We will change this elements: 56(1) and 23(4)  [3, 23, 89, 12, 56]  This is 3 iteration of sorting  We will change this elements: 23(1) and 12(3)  We will change this elements: 89(2) and 56(4)  [3, 12, 56, 23, 89]  This is 4 iteration of sorting  We will change this elements: 56(2) and 23(3)  [3, 12, 23, 56, 89]  This is sorted array  [3, 12, 23, 56, 89] | They are identical |
| 5. |  | This is sorted array  [] | They are identical |
| 6. | 12 90 4 25 8 3 1 666 453 | This is 1 iteration of sorting  [12, 90, 4, 25, 8, 3, 1, 666, 453]  This is 2 iteration of sorting  We will change this elements: 12(0) and 3(5)  We will change this elements: 90(1) and 1(6)  [3, 1, 4, 25, 8, 12, 90, 666, 453]  This is 3 iteration of sorting  [3, 1, 4, 25, 8, 12, 90, 666, 453]  This is 4 iteration of sorting  [3, 1, 4, 25, 8, 12, 90, 666, 453]  This is 5 iteration of sorting  We will change this elements: 25(3) and 12(5)  [3, 1, 4, 12, 8, 25, 90, 666, 453]  This is 6 iteration of sorting  We will change this elements: 3(0) and 1(1)  We will change this elements: 12(3) and 8(4)  We will change this elements: 666(7) and 453(8)  [1, 3, 4, 8, 12, 25, 90, 453, 666]  This is sorted array  [1, 3, 4, 8, 12, 25, 90, 453, 666] | They are identical |