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: Алгоритмы сортировки

Студент гр. 9381 _____ Аухадиев А.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Познакомиться с алгоритмами сортировки, реализовать алгоритм двухпутевой сортировки бинарными вставками на языке C++.

Задание.

Вариант 3.

Двухпутевая сортировка бинарными вставками.

Описание алгоритма.

Двухпутевая сортировка бинарными вставками предполагает создание из исходного массива нового массива путём добавления в него элементов из исходного. Сначала в новый массив добавляется первый элемент исходного, затем все остальные по очереди.

При очередной вставке новый элемент сравнивается с крайними элементами (первым и последним) и добавляется в начало, если он меньше либо равен первому или в конец, если он больше либо равен последнему. Если ни одно из этих условий не выполняется, элемент сравнивается с элементом, находящимся в середине отсортированного массива.

Если элемент меньше среднего элемента, то элемент добавляется в начало массива, и массив разбивается на две части с фиксацией левой и правой границ, и элемент сравнивается со средним элементом левой половины. В зависимости от результатов сравнений со средними элементами, левая и правая границы сдвигаются, пока правая не оказывается левее. Затем вся часть массива, находящаяся левее левой границы сдвигается влево, и на освободившееся место ставится новый элемент.

Если элемент больше среднего, то происходят аналогичные операции в правой части массива с требуемым сдвигом элементов вправо.

Таким образом, при каждой вставке происходит сдвиг не более половины элементов массива.

Данный алгоритм прост в реализации, эффективен на небольших наборах данных, эффективен на частично отсортированных данных (если в исходном

массиве находятся подмассивы с числами по возрастанию или убыванию, быстро добавляемые в начало или конец), устойчив (не меняется порядок отсортированной части), может сортировать список по мере его получения. Недостатком является обработка каждого символа отдельно и то, что по мере увеличения кол-ва элементов в массиве, увеличивается и время выполнения одной итерации.

Всего происходит не более $N*\log_2 N$ сравнений, причём значение N увеличивается с каждым шагом на 1. Кол-во пересылок не превышает $N^2 / 8$.

Выполнение работы.

Была написана шаблонная функция binDoubleWaySort, принимающая в качестве шаблона тип данных, которые нужно отсортировать. В данной функции из исходного массива создаётся новый массив путём переноса символа, пока исходный массив не станет пустым. Функция реализует сортировку, алгоритм которой описан выше.

Разработанный программный код см. в приложении А.

Результаты тестирования см. в приложении Б.

Выводы.

Были изучены алгоритмы сортировки, реализован алгоритм сортировки двухпутевыми бинарными вставками на языке программирования C++.

Разработана функция binDoubleWay*Sort*, при помощи которой можно отсортировать массив данных.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <string>
#include <vector>
#include <sstream>
#include <fstream>
#include <algorithm>

int stepNumber = 0;

template <typename T>
void printArray(std::vector<T> arr){    //Вывод массива на экран
    for(auto i : arr)
        std::cout << i << ' ';
    std::cout << '\n';
}

template <typename T>
std::vector<T> binDoubleWaySort(std::vector<T> arr){
    std::vector<T> arr1;        //Новый массив для заполнения
отсортированными элементами
    int x = arr[0];
    arr1.push_back(x);     //Добавление в новый массив первого элемента
    arr.erase(arr.cbegin());    //Удаление из старого массива первого
элемента

    while(!arr.empty()){
        std::cout << "Шаг № " << ++stepNumber << " (элемент " << x << "):\
n";
        printArray(arr1);
        x = arr[0];                 //Рассматриваемый элемент
        arr.erase(arr.cbegin());    //Удаление рассматриваемого элемента
из старого массива
        if(x >= arr1.back()) {      //Если элемент больше или равен
последнему элементу,
            arr1.push_back(x);      //добавляем его в конец
отсортированного массива
            continue;               //и переходим к следующей итерации
        }
         if(x <= arr1.front()){     //Если элемент меньше или равен
первому элементу,
             arr1.insert(arr1.cbegin(), x); //Добавляем его в начало
отсортированного массива
              continue;                     //и переходим к следующей
итерации
         }
        int midIndex = arr1.size()/2;   //индекс среднего элемента
отсортированного массива
        int midElem = arr1[midIndex];  //средний элемент отсортированного
массива
        if(x < midElem){                //Если элемент меньше среднего,
            arr1.insert(arr1.cbegin(), x); //добавляем его в начало
массива
            int left = 1;                   //Индекс крайнего левого
элемента (не считая добавленного)
            int right = midIndex + 1;       //Индекс крайнего правого
элемента (среднего)

            do{
```

```
                    int mid = (left + right)/2; //Индекс среднего элемента
левой части массива
                    if(arr1[mid] < x)           //Если средний элемент
подмассива меньше рассматриваемого,
                        left = mid + 1;         //Левая граница смещается до
середины + 1
                    else                        //Если средний элемент
подмассива больше рассматриваемого
                        right = mid - 1;        //Сдвигаем правую границу до
середины - 1
                }while(left <= right);          //Сдвиги происходят, пока
левая граница не превзойдёт правую,
                                                //так образуется место для
добавления элемента
                for(auto i = 0; i < left - 1; i++)  //Сдвиг всех элементов
                    arr1[i] = arr1[i + 1];          //для освобождения места
для добавления
                arr1[left - 1] = x;                 //Добавление элемента в
отведённое место
            }else{                              //Если рассматриваемый элемент
больше среднего,
                arr1.push_back(x);              //он добавляется в конец
массива, и в правой части производятся
                int left = midIndex -1;         //операции, аналогичные
операциям левой
                int right = arr1.size() - 1;

                do{
                    int mid = (left + right) / 2;
                    if(arr1[mid] < x)
                        left = mid + 1;
                    else
                        right = mid - 1;
                }while(left <= right);

                for(auto i = arr1.size()-1; i > left; i--) {
                    arr1[i] = arr1[i - 1];
                }
                arr1[left] = x;
            }
        }

        std::cout << "Шаг № " << ++stepNumber << " (элемент " << x << "):\n";
        printArray(arr1);
        std::cout << "Конец сортировки\n";

        return arr1;
    }

    int main() {
        std::vector<int> arr;
        std::string str;

        int inputFlag;
        std::cout << "Выберите, откуда будет производиться ввод (0 - консоль,
1 - файл)\n";
        std::cin >> inputFlag;

        switch(inputFlag){
            case 0:
                std::cout << "Введите числа\n";
                std::cin.ignore();
                std::getline(std::cin, str);
                break;
```

```
        case 1:
            std::string fileName;
            std::cout << "Введите имя файла\n";
            std::cin >> fileName;
            std::ifstream fin(fileName);

            if(!fin.is_open()){
                std::cout << "Файл не найден\n";
                return 0;
            }

            std::getline(fin, str);
            fin.close();
            std::cout << "Сортируемый массив\n" << str << '\n';
    }

    std::istringstream stream (str);
    int x = 0;

    while(stream >> x)
        arr.push_back(x);

    if(arr.empty())
        return 0;

    std::vector<int> arr1 = arr;
    arr = binDoubleWaySort(arr);

    std::sort(arr1.begin(), arr1.end());

    std::cout << "Результат работы библиотечнй сортировки:\n";
    printArray(arr1);

    return 0;
}
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Таблица Б.1 - Примеры тестовых случаев на некорректных данных

№ п/п	Входные данные	Выходные данные
1.	-121 54 345 76 0 32 -76 45 0 65 23 11 33 65 3 5 -5 -10	Шаг № 1 (элемент -121): -121 Шаг № 2 (элемент 54): -121 54 Шаг № 3 (элемент 345): -121 54 345 Шаг № 4 (элемент 76): -121 54 76 345 Шаг № 5 (элемент 0): -121 0 54 76 345 Шаг № 6 (элемент 32): -121 0 32 54 76 345 Шаг № 7 (элемент -76): -121 -76 0 32 54 76 345 Шаг № 8 (элемент 45): -121 -76 0 32 45 54 76 345 Шаг № 9 (элемент 0): -121 -76 0 0 32 45 54 76 345 Шаг № 10 (элемент 65): -121 -76 0 0 32 45 54 65 76 345 Шаг № 11 (элемент 23): -121 -76 0 0 23 32 45 54 65 76 345 Шаг № 12 (элемент 11): -121 -76 0 0 11 23 32 45 54 65 76 345 Шаг № 13 (элемент 33): -121 -76 0 0 11 23 32 33 45 54 65 76 345 Шаг № 14 (элемент 65): -121 -76 0 0 11 23 32 33 45 54 65 65 76 345 Шаг № 15 (элемент 3): -121 -76 0 0 3 11 23 32 33 45 54 65 65 76 345 Шаг № 16 (элемент 5): -121 -76 0 0 3 5 11 23 32 33 45 54 65 65 76 345 Шаг № 17 (элемент -5): -121 -76 -5 0 0 3 5 11 23 32 33 45 54 65 65 76 345 Шаг № 18 (элемент -10): -121 -76 -10 -5 0 0 3 5 11 23 32 33 45 54 65 65 76 345

		Конец сортировки Результат работы библиотечнй сортировки: -121 -76 -10 -5 0 0 3 5 11 23 32 33 45 54 65 65 76 345
2.	1 5 7 -3 5 12 65 -40	Шаг № 1 (элемент 1): 1 Шаг № 2 (элемент 5): 1 5 Шаг № 3 (элемент 7): 1 5 7 Шаг № 4 (элемент -3): -3 1 5 7 Шаг № 5 (элемент 5): -3 1 5 5 7 Шаг № 6 (элемент 12): -3 1 5 5 7 12 Шаг № 7 (элемент 65): -3 1 5 5 7 12 65 Шаг № 8 (элемент -40): -40 -3 1 5 5 7 12 65 Конец сортировки Результат работы библиотечнй сортировки: -40 -3 1 5 5 7 12 65
3.	<пустая строка>	Массив пуст
4.	9 8 7 6 5 4 3 2 1 0	Шаг № 1 (элемент 9): 9 Шаг № 2 (элемент 8): 8 9 Шаг № 3 (элемент 7): 7 8 9 Шаг № 4 (элемент 6): 6 7 8 9 Шаг № 5 (элемент 5): 5 6 7 8 9 Шаг № 6 (элемент 4): 4 5 6 7 8 9 Шаг № 7 (элемент 3): 3 4 5 6 7 8 9 Шаг № 8 (элемент 2): 2 3 4 5 6 7 8 9 Шаг № 9 (элемент 1): 1 2 3 4 5 6 7 8 9

		Шаг № 10 (элемент 0): 0 1 2 3 4 5 6 7 8 9 Конец сортировки Результат работы библиотечнй сортировки: 0 1 2 3 4 5 6 7 8 9
5.	-10 -4 -1 0 4 8 2 -2	Шаг № 1 (элемент -10): -10 Шаг № 2 (элемент -4): -10 -4 Шаг № 3 (элемент -1): -10 -4 -1 Шаг № 4 (элемент 0): -10 -4 -1 0 Шаг № 5 (элемент 4): -10 -4 -1 0 4 Шаг № 6 (элемент 8): -10 -4 -1 0 4 8 Шаг № 7 (элемент 2): -10 -4 -1 0 2 4 8 Шаг № 8 (элемент -2): -10 -4 -2 -1 0 2 4 8 Конец сортировки Результат работы библиотечнй сортировки: -10 -4 -2 -1 0 2 4 8
6.	0	Шаг № 1 (элемент 0): 0 Конец сортировки Результат работы библиотечнй сортировки: 0