**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. 9382 _____ Павлов Р.В.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

**Цель**

Изучение алгоритмов сортировки, реализация одного из них.

**Основные теоретические сведения**

Сортировка кучей, пирамидальная сортировка (англ. Heapsort) — алгоритм сортировки, использующий структуру данных двоичная куча. Это неустойчивый алгоритм сортировки с временем работы O(nlogn), где n — количество элементов для сортировки, и использующий O(1) дополнительной памяти.

**Задание**

Вариант 13: Пирамидальная сортировка

**Алгоритм**

1. На основе массива, начиная с листьев, строится куча, в которой предки всегда больше потомков, т. е. массив становится отсортированным по убыванию. Начиная с элемента массива под номером **i = n/2 – 1**, где n – длина массива, рассматриваются его дочерние элементы под номерами **2*i + 1** и **2*i + 2** (так в массиве хранится бинарная куча, поэтому нет смысла проверять элементы массива дальше **n/2 – 1** - ого). Рассмотрение заканчивается на нулевом элементе массива (корне). При этом, если какой-то дочерний элемент больше корня и больше другого дочернего элемента, он помещается в корень, и корень становится на его место, а затем у старого корня проверяются дочерние элементы (может возникнуть необходимость сделать такую же перестановку уже в дочерних элементах и их потомках).
2. Поскольку теперь максимальный элемент в корне, меняем его с последним доступным, а затем заново строим кучу, отсортированную по убыванию, из всех элементов, кроме того, который поставили на место последнего доступного. Уменьшаем номер последнего доступного элемента на

1. Так делаем с **n-1** по **1** элемент. Постепенно массив заполняется убывающими значениями с конца, таким образом получается, что он сортируется по возрастанию.

**Функции и СД**

- **template <typename T> struct Content{**
  **T content;**
  **}** - шаблонный тип данных
- **inline void indent(int n)** – вывод отступа, соответствующего глубине рекурсии; n – глубина рекурсии
- **template <typename T> void maxHeap(T* arr, int n, int i, int& depth)** – формирование кучи, отсортированной по убыванию (корень — самый большой элемент); arr – указатель на корень дерева (начало массива), n – длина сортируемого массива, i — номер текущего корня, depth – глубина рекурсии
- **template <typename T> void heapSort(T* arr, int n, int& depth)** — сортировка массива; arr – указатель на корень дерева (начало массива), n – длина сортируемого массива, depth – глубина рекурсии
- **int main()** :
  - Файловые ввод/вывод
  - Консольные ввод/вывод
  - Сортировка и вывод промежуточных/итоговых результатов

## Тестирование

Результаты тестирования представлены в таблице 1.

Таблица 1

| **Входные данные** | **Выходные данные** |
|---|---|
| 7 -15 9 123 0 14 -35 72 | -35, -15, 0, 9, 14, 72, 123 |
| 2 1 -12 | -12, 1 |
| 1 4 | 4 |
| 0 4 1 6 3 8 | NA (Not Array) |
| -5 0 14 | NA (Not Array) |
| 3 -46 88 14 523 | -46, 14, 88 |

# ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

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

using namespace std;

//Шаблонный тип данных
template <typename T>
struct Content {
	T content;
};

//Отступ, соответствующий глубине рекурсии
inline void indent(int n) {
	for (int i = 0; i < n; i++) {
		cout << "\t";
	}
}


//Создание кучи, отсортированной по убыванию
template <typename T>
void maxHeap(T* arr, int n, int i, int& depth){
	int largest = i;
	int l = 2 * i + 1;
	int r = 2 * i + 2;
	if (l < n && arr[l].content > arr[largest].content) { //Если левое
поддерево больше корня
		indent(depth);
		cout << "The left child " << arr[l].content << " is greater than "
<< arr[largest].content << "\n";
		largest = l;
	}
	if (r < n && arr[r].content > arr[largest].content) { //Если правое
поддерево больше корня/левого поддерева
		indent(depth);
		cout << "The right child " << arr[r].content << " is greater than "
<< arr[largest].content << "\n";
		largest = r;
	}
	if (largest != i){			//Если корень меньше хотя бы одного из своих
поддеревьев
		indent(depth);
		cout << "Swapping " << arr[largest].content << " and " <<
arr[i].content << ", " << arr[i].content << " is now the root\n";
		swap(arr[i], arr[largest]);		//Меняем местами наибольший
элемент с корнем
		indent(depth);
		cout << "Checking the lower levels\n";
		maxHeap(arr, n, largest, ++depth);  //Проверяем то, что было корнем
	}
	else {					//Если корень больше или равен обоим своим
поддеревьям
		indent(depth);
		cout << "The root " << arr[largest].content << " is the largest\n";
		indent(depth);
		cout << "Finished creating maxHeap\n";
		if (!depth) {
```

```
                cout << "\n";
            }
        }
        if (depth) {
            depth--;
        }
}

//Сортировка по возрастанию
template <typename T>
void heapSort(T* arr, int n, int& depth){
        cout << "\nSort started\n\nPre-sort (descending sort)\n";
        for (int i = n / 2 - 1; i >= 0; i--) {                                    //
Проходим по куче с листьев, строим убывающую кучу
            cout << "Creating maxHeap from root " << arr[0].content << " to " <<
(n - 1) << " element " << arr[n - 1].content << " starting from " << i << "
element " << arr[i].content << "\n";
            maxHeap(arr, n, i, depth);
        }
        cout << "Final sort :\n";
        for (int i = n - 1; i > 0; i--){                                          //
Меняем первый и последний доступный элементы, строим убывающую кучу для первых i
- 1 элементов (пока i > 0)
            cout << "Swapping the root " << arr[0].content << " and the last
element of array " << arr[i].content << "\n";
            swap(arr[0], arr[i]);
            cout << "Creating maxHeap from root " << arr[0].content << " to " <<
(i - 1) << " element " << arr[i - 1].content << " starting from 0 element " <<
arr[0].content << "\n";
            maxHeap(arr, i, 0, depth);
        }
        cout << "Sort finished\n\n";
}

int main() {

        int command = 0;
        cout << "1 - console input, 2 - file input : \n\n";
        cin >> command;
        if (command != 1 && command != 2) {
            cout << "Unknown command\n";
        }
        else{
            int* arra;
            Content<int>* arr;
            int x, size, depth = 0;

            //Работа с консолью
            if (command == 1) {
                while (true) {
                    cout << "Enter the array size (0 to quit): ";
                    cin >> size;
                    if (size < 0) {
                        cout << "\nInvalid value\n\n";
                        continue;
                    }
                    if (!size) {
                        break;
                    }

                    //Создание, заполнение и сортировка массивов
                    arr = new Content<int>[size];
                    arra = new int[size];
```

```
                        for (int i = 0; i < size; i++) {
                              cout << i + 1 << " element : ";
                              cin >> x;
                              arr[i].content = x;
                              arra[i] = x;
                        }
                        if (size == 1) {
                              cout << "Sort isn't needed\n\n";
                              continue;
                        }
                        heapSort(arr, size, depth);
                        cout << "Initial array :\n";
                        for (int i = 0; i < size; ++i) {
                              cout << arra[i] << " ";
                        }
                        cout << "\n";
                        sort(arra, arra + size);
                        cout << "Heap sort result :\n";
                        for (int i = 0; i < size; i++) {
                              cout << arr[i].content << " ";
                        }
                        cout << "\n";
                        cout << "std::sort result :\n";
                        for (int i = 0; i < size; ++i) {
                              cout << arra[i] << " ";
                        }
                        cout << "\n";
                  }
            }

            //Работа с файлом
            else {
                  ifstream input;
                  ofstream output;
                  const string outputFilename = "output.txt";
                  string inputFilename = "";
                  output.open(outputFilename);
                  output << "";
                  output.close();
                  while (inputFilename != "!") {
                        cout << "Enter the input file name (or \'!\' to quit) :
";
                        cin >> inputFilename;
                        cout << "\n";
                        bool valid = true;
                        input.open(inputFilename);
                        if (input.is_open()) {
                              input >> size;
                              if (size < 1) {
                                    output.open(outputFilename, ios::app);
                                    output << "NA (Not Array)\n";
                                    output.close();
                                    cout << "Invalid size\n\n";
                                    input.close();
                                    continue;
                              }

                              //Создание, заполнение и сортировка массивов
                              arr = new Content<int>[size];
                              arra = new int[size];
                              for (int i = 0; i < size; i++) {
                                    if (input >> x) {
                                          arr[i].content = x;
```

```
                                    arra[i] = x;
                              }
                              else {
                                    valid = false;
                                    break;
                              }
                        }
                        input.close();
                        if (size == 1) {
                              output.open(outputFilename, ios::app);
                              output << arr[0].content << "  ->  " <<
arr[0].content << "\n";
                              output.close();
                              cout << "Initial array : " << arr[0].content
<< "\n";
                              cout << "Sort isn't needed\n\n";
                              continue;
                        }
                        if (!valid) {
                              cout << "\nNot enough elements to fill the
array\n\n";
                              continue;
                        }
                        output.open(outputFilename, ios::app);
                        for (int i = 0; i < size; i++) {
                              output << arr[i].content;
                              if (i != size - 1) {
                                    output << ", ";
                              }
                        }
                        output << "  ->  ";
                        heapSort(arr, size, depth);
                        cout << "Initial array :\n";
                        for (int i = 0; i < size; ++i) {
                              cout << arra[i] << " ";
                        }
                        for (int i = 0; i < size; i++) {
                              output << arr[i].content;
                              if (i != size - 1) {
                                    output << ", ";
                              }
                        }
                        output << "\n";
                        output.close();
                        cout << "\n";
                        sort(arra, arra + size);
                        cout << "Heap sort result :\n";
                        for (int i = 0; i < size; i++) {
                              cout << arr[i].content << " ";
                        }
                        cout << "\n";
                        cout << "std::sort result :\n";
                        for (int i = 0; i < size; ++i) {
                              cout << arra[i] << " ";
                        }
                        cout << "\n\n";
                  }
                  else if (inputFilename == "!") {
                        break;
                  }
                  else if (inputFilename != "") {
                        cout << inputFilename << " doesn't exist\n\n";
                  }
```

```
                    }
                }

        }
        return 0;
}
```