МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Сортировка n-арной кучей.

Демонстрация

Студентка гр. 9382 _____ Пя С.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студентка Пя С.

Группа 9382

Тема работы: Сортировка n-арной кучей (n=1, 2, 3, …), 2 варианта просеивания (сверху-вниз и снизу вверх). Демонстрация (вариант 29)

Исходные данные:
"Демонстрация" - визуализация структур данных, алгоритмов, действий. Демонстрация должна быть подробной и понятной (в том числе сопровождаться пояснениями), чтобы программу можно было использовать в обучении для объяснения используемой структуры данных и выполняемых с нею действий.
Содержание пояснительной записки:

«Содержание», «Введение», «Структура программы», «Тестирование», «Выводы», «Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 31.10.2020

Дата сдачи реферата: 14.12.2020

Дата защиты реферата: 14.12.2020

Студентка гр. 9382 _____ Пя С.

Преподаватель _____ Фирсов М.А.

АННОТАЦИЯ

В курсовой работе реализована n-нарная куча с возможностью выбора просейки. Программа демонстрирует сортировку элементов массива с указанием максимального количества сыновей. Каждый шаг сопровожден пояснениями, которые выводятся в терминал.

Код программы приведен в приложении А.

SUMMARY

In the course work, an n-Nar heap is implemented with the ability to select a sifting. The program demonstrates how to sort elements of array with instructions of the maximum number of sons. Each step is accompanied by explanations that are output to the terminal.

The program code is given in Appendix A.

СОДЕРЖАНИЕ

1

ВВЕДЕНИЕ

Целью работы: реализация программы, демонстрирующей сортировку N-нарными кучами с подробным пояснением шагов.

Задачи:

- реализация двух вариантов просеивания и сортировки N-нарной кучей
- написание функций демонстрации шагов алгоритма
- написание пояснений для каждого шага в целях обучения

Куча – специальная древовидная структура, использующаяся для организации данных. Это дерево, в котором все родительские узлы не меньше, чем узлы-потомки.

Просейка – упорядочивание элементов в дереве для формирования и поддержания состояния сортирующего дерева. Просеивание для элемента состоит в том, что если он меньше по размеру чем потомки, объединённых в неразрывную цепочку, то этот элемент нужно переместить как можно ниже, а бóльших потомков по ветке поднять наверх на 1 уровень.

N-нарная куча – это куча с n количеством потомков. Для i-го элемента массива индексы (если отсчитывать их с нуля) его N потомков вычисляются:

1-й потомок: $N \times i + 1$

2-й потомок: $N \times i + 2$

3-й потомок: $N \times i + 3$

…

N-й потомок: $N \times i + N$

Разработка велась в IDE CLion 2020.2.1 на ОС Windows 10.

В результате была написана программа, которая демонстрирует сортировку N-нарной кучи.

1. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2 . Описание алгоритма

В начале работы программы в массив заносятся переданные вводом с консоли/из файла данные и максимальное количество сыновей. Дерево из массива формируется за счёт индексов. Под нулевым индексом соответственно будет находится корень дерева (у кучи есть друге название – сортирующее дерево). Этот корень будет являться узлом нулевого уровня. Следующие n элементов (в n-нарной куче) будут составлять 1-ый уровень. В следующем этапе будут идти уже их потомки на одном уровне - элементы под индексом от n*i+ 1 до n*i + n. Если индексы потомков выходят за пределы массива, значит у i-го элемента потомков нет или присутствуют не все потомки. (пример вывода кучи – рис.1)

Из массива путем просейки (на выбор дается два варианта: сверху-вниз и снизу-вверх) образуется куча, в корне которой лежит максимальное значение.

В первом случае просейки родитель сравнивается с сыновьями. Элемент-родитель сравнивается с максимальным по значению сыном из остальных и меняется с ним местами в случае своего уступающего значения. И если произошла замена, то сын принимает роль родителя и сравнивается со своими сыновьями. Так происходит с каждым элементом по очереди, начиная с конца.

Во втором случае просейки сравнивается уже сын с родителем. Если он превышает по значению родителя, то они меняются местами. Так происходит с каждым элементом по очереди, начиная с начала.

Чтобы начать сортировку, необходимо превратить обычное дерево в сортирующее с помощью просейки. Затем максимум, который находится на вершине, мы меняем местами с последним элементом, откуда он больше не перемещается. А затем для оставшихся элементов массива происходит

просейка для нахождения следующего максимума до того момента, пока весь массив не будет отсортирован по возрастанию.

Предусмотрен механизм простейшего взаимодействия с пользователем, позволяющий понять алгоритм исполнения программы, с помощью вывода сообщений. Также был предусмотрен ввод данных с клавиатуры. (Примеры работы алгоритма ниже в разделе Тестирование – таблицы 1)

3 1.2. Описание функций и структур данных

Использованы функции:

1. printHeap

Сигнатура: void printHeap(int n, int u).

Назначение: выводит массив в виде n-нарной кучи.

Описание аргументов: n : int – количество значений в массиве, u : int – максимальное количество сыновей.

2. printArray

Сигнатура: void printArray(int n, string message).

Назначение: выводит текущее состояние массива.

Описание аргументов: n: int – количество значений массива, message : string – сообщение, которое выводится на экран.

3. heapifyDown

Сигнатура: void heapifyDown (int pos, int n, int u, int m, string message).

Назначение: осуществляет просейку, сравнивая значение родителя с максимальным сыном.

Описание аргументов: pos : int – индекс текущего элемента, n: int – рассматриваемое количество значений массива, u : int – максимальное

количество сыновей, m : int – общее количество значений массива, message : string – сообщение, которое выводится на экран.

4. heapifyUp

Сигнатура: void heapifyUp (int pos, int u, int m, string message).

Назначение: осуществляет просейку, сравнивая значение сына с родителем.

Описание аргументов: pos : int – индекс текущего элемента, u : int – максимальное количество сыновей, m : int – общее количество значений массива, message : string – сообщение, которое выводится на экран.

5. heapMakeDown

Сигнатура: void heapMakeDown(int n, int u, int m, string message).

Назначение: создает из массива кучу путем применения просеивания ко всем элементам по очереди, начиная с конца.

Описание аргументов: n: int –рассматриваемое количество значений массива, u : int – максимальное количество сыновей, m : int – общее количество значений массива, message : string – сообщение, которое выводится на экран.

6. heapMakeUp

Сигнатура: void heapMakeUp(int n, int u, int m, string message).

Назначение: создает из массива кучу путем применения просеивания ко всем элементам по очереди, начиная с начала.

Описание аргументов: n: int –рассматриваемое количество значений массива, u : int – максимальное количество сыновей, m : int – общее количество значений массива, message : string – сообщение, которое выводится на экран.

7. heapSortDown

Сигнатура: void heapSortDown(int n, int u).

Назначение: сортирует массив элементов по возрастанию, используя просейку снизу-вверх.

Описание аргументов: n: int –рассматриваемое количество значений массива, u : int – максимальное количество сыновей.

8. heapSortUp

Сигнатура: void heapSortUp(int n, int u).

Назначение: сортирует массив элементов по возрастанию, используя просейку сверху-вниз.

Описание аргументов: n: int –рассматриваемое количество значений массива, u : int – максимальное количество сыновей.

9. fcn

Сигнатура: void fcn(std::istream &fin).

Назначение: предназначена для универсальной работы с потоками. Также в ней реализуются все остальные функции.

Описание аргументов: fin : istream – поток ввод.

В главной функции main() происходит приветствие и зацикливание выполнения программы. Также здесь выбирается поток ввода значений и максимального количества сыновей.

4 1.3. Описание интерфейса пользователя

В начале работы на экран выводится приветствие, информация о программе, ученике и краткая инструкция по использованию демонстрации. У пользователя есть возможность выбрать поток ввода данных, варианта просейки. Также у него есть возможность выбора пропуска демонстрации для вывода конечного отсортированного массива, и он может остановиться на любом доступном шаге демонстрации. Также есть возможность продолжить выполнение программы после вывода конечного результата.

Пример работы программы показан на рис. 1.

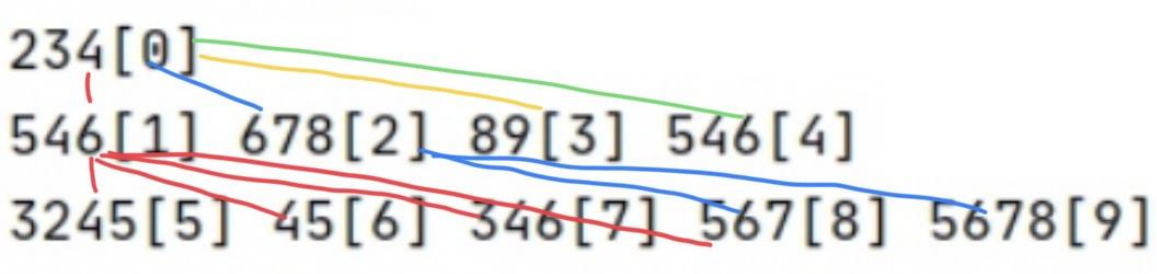


Рис. 1 – демонстрация n-нарной кучи

2. ТЕСТИРОВАНИЕ

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 87 56 34 23 23 2	Current state of array is: 23[0] 23[1] 34[2] 56[3] 87[4]	Проверка на корректность работы программы с терминалом
2.	1 9 8 7 6 5 4 3 2 1 3	Current state of array is: 1[0] 2[1] 3[2] 4[3] 5[4] 6[5] 7[6] 8[7] 9[8]	Проверка на корректность работы с файлом
3.	2 2	2 Error data You entered the wrong data!	Проверка на корректность работы с неверными данными
4.	3 1 1	Current state of array is: 1[0]	Проверка на корректность работы с пограничными значениями

5.	5 987 654 543 234 211 87 56 45 34 23 6789 6544 3456 2345 4321 1234 4	Current state of array is: 23[0] 34[1] 45[2] 56[3] 87[4] 211[5] 234[6] 543[7] 654[8] 987[9] 1234[10] 2345[11] 3456[12] 4321[13] 6544[14] 6789[15]	Проверка на корректность работы с файлом
6	4 987 654 543 234 211 87 56 45 34 23 6789 6544 3456 2345 4321 1234 4	Current state of array is: 23[0] 34[1] 45[2] 56[3] 87[4] 211[5] 234[6] 543[7] 654[8] 987[9] 1234[10] 2345[11] 3456[12] 4321[13] 6544[14] 6789[15]	Проверка на корректность работы с файлом

ЗАКЛЮЧЕНИЕ

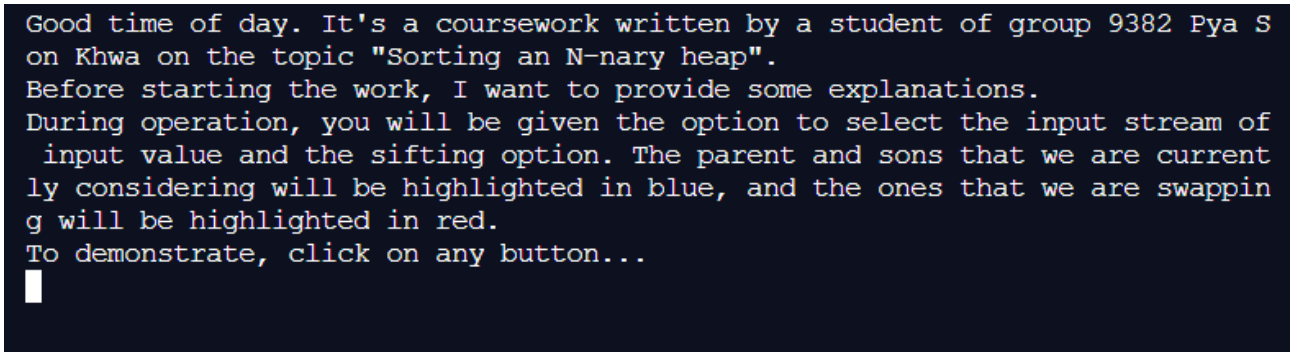
В результате выполнения курсовой работы была создана программа, демонстрирующая сортировку N-нарной кучей. Также были реализованы два метода просейки.

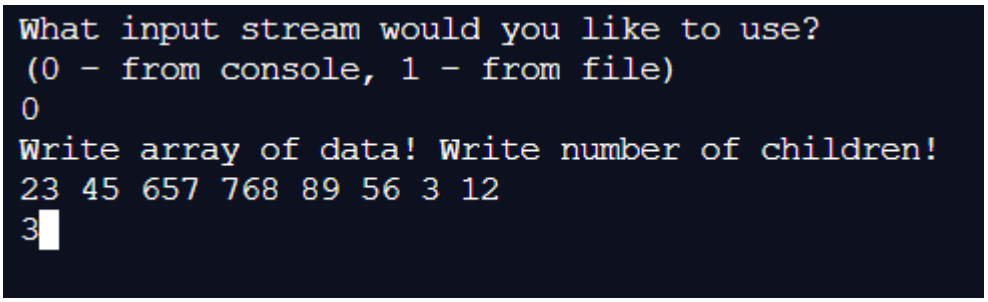
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

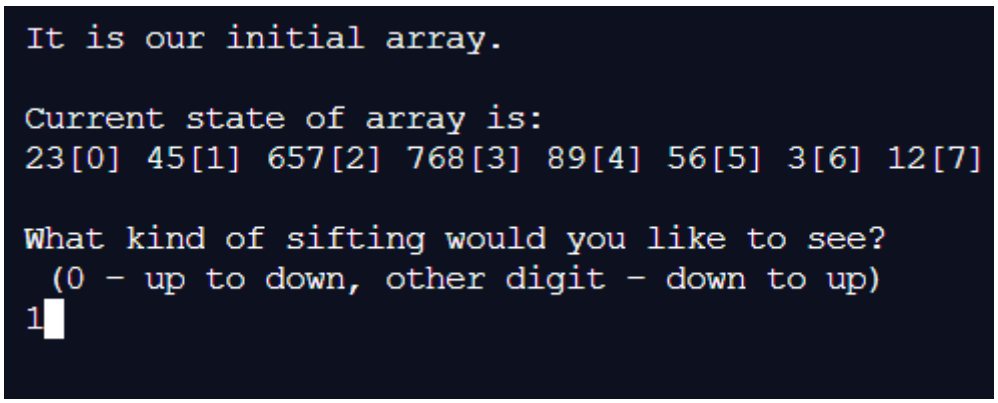
1. Neerc ITMO: https://neerc.ifmo.ru/wiki/index.php?title=%C4%E2%EE%E8%F7%ED%E0%FF_%EA%F3%F7%E0
2. Web.Archive: https://web.archive.org/web/20090315200203/http://iproc.ru/parallel-programming/lection-5/
3. Cybern: https://away.vk.com/away.php
4. Хабр: http://cybern.ru/heapsortcpp.html

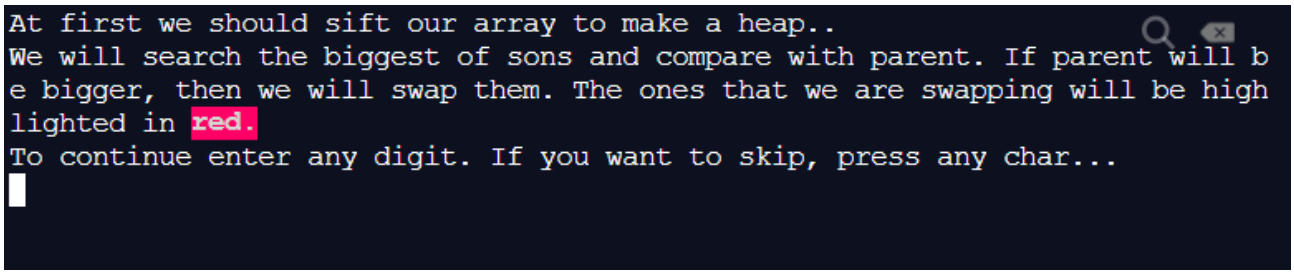
ПРИЛОЖЕНИЕ А

ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ









```
It's sifting down to up...

Current state of array is:
23[0] 45[1] 657[2] 768[3] 89[4] 56[5] 3[6] 12[7]

Current state of 3-nary heap is:
23[0]
45[1] 657[2] 768[3]
89[4] 56[5] 3[6] 12[7]
To continue enter any digit. If you want to skip, press any char...
```

```
It's sifting down to up...

Current state of array is:
23[0] 45[1] 657[2] 768[3] 89[4] 56[5] 3[6] 12[7]

Current state of 3-nary heap is:
23[0]
45[1] 657[2] 768[3]
89[4] 56[5] 3[6] 12[7]
To continue enter any digit. If you want to skip, press any char...
```

Затем происходит проход по всем элементам.

Демонстрация обмена значениями в случае наибольшего значения у сына:

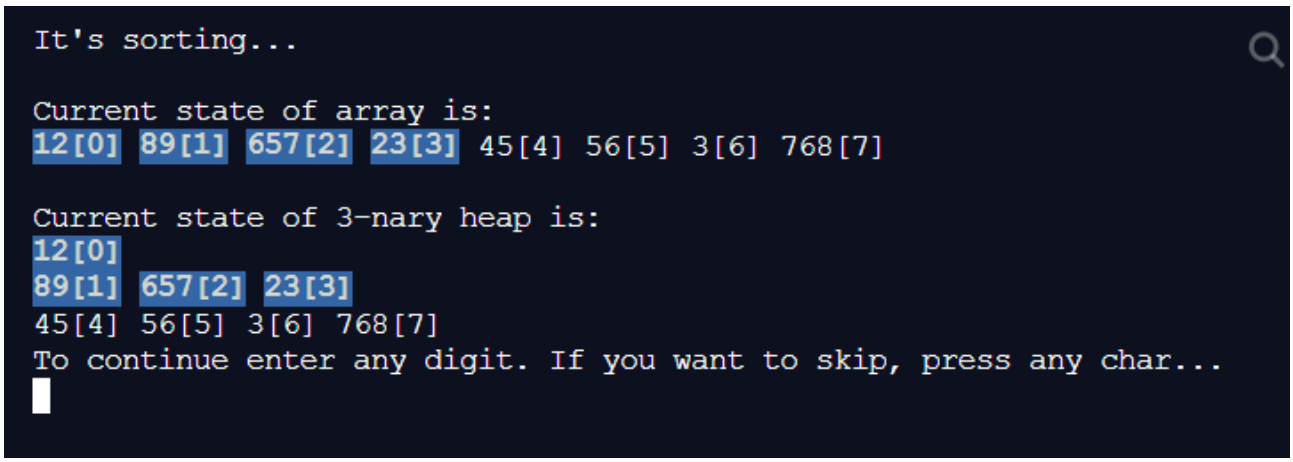
```
It's sifting down to up...

Current state of array is:
23[0] 89[1] 657[2] 768[3] 45[4] 56[5] 3[6] 12[7]

Current state of 3-nary heap is:
23[0]
89[1] 657[2] 768[3]
45[4] 56[5] 3[6] 12[7]
To continue enter any digit. If you want to skip, press any char...
2
```

```
It's sifting down to up...

Current state of array is:
768[0] 89[1] 657[2] 23[3] 45[4] 56[5] 3[6] 12[7]

Current state of 3-nary heap is:
768[0]
89[1] 657[2] 23[3]
45[4] 56[5] 3[6] 12[7]
To continue enter any digit. If you want to skip, press any char...
```

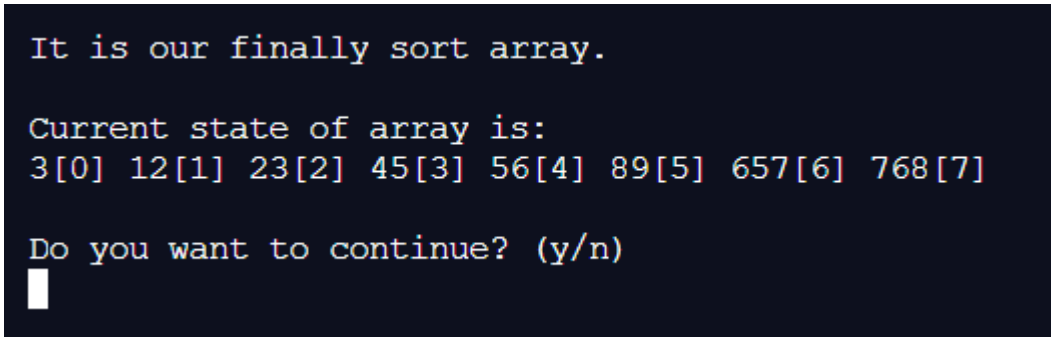
```
Now we are going to sort.
Max array member will sent to the end, and array will be sift again.
To continue enter any digit. If you want to skip, press any char...
```

```
Forming an array of maximums.

Current state of array is:
768[0] 89[1] 657[2] 23[3] 45[4] 56[5] 3[6] 12[7]

Current state of 3-nary heap is:
768[0]
89[1] 657[2] 23[3]
45[4] 56[5] 3[6] 12[7]
To continue enter any digit. If you want to skip, press any char...
```

```
Forming an array of maximums.

Current state of array is:
12[0] 89[1] 657[2] 23[3] 45[4] 56[5] 3[6] 768[7]

Current state of 3-nary heap is:
12[0]
89[1] 657[2] 23[3]
45[4] 56[5] 3[6] 768[7]
To continue enter any digit. If you want to skip, press any char...
```



Далее просеивание повторяется..



ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <cstring>
#include <string>
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <limits>
#include <vector>

#define NMAX 500
using namespace std;

int arr[NMAX+1];//массив значений
vector <int> iVec;//вектор для хранения индексов, которые нужно выделить
bool isFirst = true;//для переключения цвета фона при выводе

void printHeap(int n, int u) {//предназначена для вывода в форме кучи
    std::cout << "Current state of " << u <<"-nary heap is:\n";
    int k = 1, i = 0, y = 0;
    while (true) {
        while (i < k + y) {
            bool flag = false;
            if (!iVec.empty()) {
                for (auto v : iVec) {
                    if (isFirst && v == i) {//вывод рассматриваемых сыновей и
родителя
                        cout << "\033[37;41m\033[1m\033[5m" << arr[i] << "[" <<
i << "]\033[0m ";
                        flag = true;
                    } else
                    if (v == i) {
                        cout << "\033[37;44m\033[1m\033[5m" << arr[i] << "[" <<
i << "]\033[0m ";
                        flag = true;
                    }
                }
            }
            if (!flag)
                cout << arr[i] << "[" << i << "] ";//вывод сыновей и родителя
            i++;
            if (i == n) {
                y = -1;
                break;
            }
        }
        cout << "\n";
        k *= u;
        if (y == -1) {//выход из цикла
            break;
        }
        y = i;
    }
    std::cout << "To continue enter any digit. If you want to skip, press any
char...\n";
    iVec.clear();//отчищение вектора индексов
    cin >> y;
}

void printArray(int n, string message) {//предназначена для вывода массива
```

```
    system("clear");
    std::cout << message << "\nCurrent state of array is:\n";
    for (int q = 0;q < n;q++) {
        bool flag = false;
        if (!iVec.empty()) {
            for (auto v : iVec)
                if (isFirst && v == q) {//вывод рассматриваемых элементов
                    cout << "\033[37;41m\033[1m\033[5m" << arr[q] << "[" << q <<
"]\033[0m ";
                    flag = true;
                } else
                if (v == q) {
                    cout << "\033[37;44m\033[1m\033[5m" << arr[q] << "[" << q <<
"]\033[0m ";
                    flag = true;
                }
        }
        if (!flag)
            cout << arr[q] << "[" << q << "] ";//вывод элементов
    }
    std::cout << "\n\n";
}

void heapifyDown (int pos, int n, int u, int m, string message) {//предназначена
для просейки снизу-вверх
    while (u * pos + 1 < n) {
        int max = u * pos + 1;
        isFirst = false;
        iVec.push_back(pos);
        iVec.push_back(max);
        for (int k = 2;k < u + 1;k++) {//поиск максимального сына
            if (u * pos + k < n) {
                iVec.push_back(u * pos + k);
                if (arr[u * pos + k] >= arr[max]) {
                    max = u * pos + k;
                }
            }
        }
        printArray(m, message);
        printHeap(m, u);
        isFirst = true;
        iVec.push_back(max);
        iVec.push_back(pos);
        printArray(m, message);
        printHeap(m, u);
        if (arr[pos] < arr[max]) {//обмен значениями, если родитель меньше сына
            swap(arr[pos], arr[max]);
            iVec.push_back(max);
            iVec.push_back(pos);
            printArray(m, message);
            printHeap(m, u);
            isFirst = false;
            pos = max; //сын становится рассматриваемым родителем
        } else
            break;
    }
}

void heapifyUp (int pos, int u, int m, string message) {//преназначена для
просейки сверху-вниз
    if (pos > 0 && !(arr[(pos -1)/ u] < arr[pos])) {
        iVec.push_back(pos);
        iVec.push_back((pos - 1)/u);
        isFirst = true;
```

```
        printArray(m, "It's sifting up to down...\nSearching child, that is
bigger than parent...\n");
        printHeap(m, u);
    } else
        while (pos > 0 && arr[(pos -1)/ u] < arr[pos]) {//обмен в случае, если
сын больше родителя
            iVec.push_back(pos);
            iVec.push_back((pos - 1)/u);
            isFirst = true;
            printArray(m, message);
            printHeap(m, u);
            swap(arr[pos], arr[(pos - 1)/ u]);
            iVec.push_back(pos);
            iVec.push_back((pos - 1)/u);
            printArray(m, message);
            printHeap(m, u);
            pos = (pos - 1)/ u;//родитель становится рассматриваемым сыном
        }
}

void heapMakeDown(int n, int u, int m, string message) {//преназначена для
формирвания кучи
    for (int i = (n - 1); i >= 0; i--) {
        heapifyDown(i, n, u, m, message);
    }
}

void heapMakeUp(int n, int u, int m, string message) {//преназначена для
формирвания кучи
    for (int i = 1; i <= (n - 1); i++) {
        heapifyUp(i, u, m, message);
    }
}
void heapSortDown(int n, int u) {//преназначена для сортировки кучей
    system ("clear");
    cout << "At first we should sift our array to make a heap..\nWe will search
the biggest of sons and compare with parent. If parent will be bigger, then we
will swap them. The ones that we are swapping will be highlighted in
\033[37;41m\033[1m\033[5mred.\033[0m\nTo continue enter any digit. If you want
to skip, press any char...\n";
    int t;
    cin >> t;
    system ("clear");
    heapMakeDown(n, u, n, "It's sifting down to up...\n");//формирование кучи
    system ("clear");
    cout << "Now we are going to sort.\nMax array member will sent to the end,
and array will be sift again.\nTo continue enter any digit. If you want to skip,
press any char...\n";
    cin >> t;
    int m = n;
    while(n > 1) {
        isFirst = true;
        iVec.push_back(0);
        iVec.push_back(n - 1);
        printArray(m, "Forming an array of maximums.\n");
        printHeap(m, u);
        swap(arr[0], arr[n - 1]);//формирование массива по возрастанию
        iVec.push_back(0);
        iVec.push_back(n - 1);
        printArray(m, "Forming an array of maximums.\n");
        printHeap(m, u);
        n--;
        heapifyDown(0, n, u, m, "It's sorting...\n");
    }
```

```
}

void heapSortUp(int n, int u) {//преназначена для сортировки кучей
    system ("clear");
    cout << "At first we should sift our array to make a heap..\nIf child will
be bigger than parent, we will swap them.The ones that we are swapping will be
highlighted in \033[37;41m\033[1m\033[5mred.\033[0m\nTo continue enter any
digit. If you want to skip, press any char...\n";
    int t;
    cin >> t;
    system ("clear");
    heapMakeUp(n, u, n, "It's sifting up to down...\n");//формирование кучи
    system ("clear");
    cout << "Now we are going to sort.\nMax array member will sent to the end,
and array will be sift again.\nTo continue enter any digit. If you want to skip,
press any char...\n";
    cin >> t;
    int m = n;
    while(n > 1) {
        isFirst = true;
        iVec.push_back(0);
        iVec.push_back(n - 1);
        printArray(m, "Forming an array of maximums.\n");
        printHeap(m, u);
        swap(arr[0], arr[n - 1]);//формирование массива по возрастанию
        iVec.push_back(0);
        iVec.push_back(n - 1);
        printArray(m, "Forming an array of maximums.\n");
        printHeap(m, u);
        n--;
        heapMakeUp(n, u, m, "It's sorting...\n");
    }
}

void fcn(std::istream &fin) {//функция для универсальной работы с потоком ввода
    int u = 0, i = 0, value;
    char* str = new char[30]();
    while ((str[u] = fin.get()) != '\n') {//считывание значений и запись в
массив
        if (str[u] == ' ') {
            str[u] = '\0';
            u = -1;
            value = stoi(str, nullptr, 10);
            arr[i++] = value;
            delete[] str;
            str = new char[20]();
        } else if (!isdigit(str[u])) {
            break;
        }
        u++;
    }
    str[u] = '\0';
    try {
        value = stoi(str, nullptr, 10);//считывание последнего элемента
        arr[i++] = value;
    }
    catch (std::invalid_argument) {
        cout << "Error data\nYou entered the wrong data!\n";

        delete[] str;
        return;
    }
    int r;
    if (!(fin >> r)) {
```

```
        cout << "Error data\nYou entered the wrong data!\n";
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        delete[] str;
        return;
    }
    printArray(i, "It is our initial array.\n");
    std::cout << "What kind of sifting would you like to see?\n (0 - up to down,
other digit - down to up)\n";
    int answer;
    while (!(std::cin >> answer)) {
        std::cout << "Error number of sifting!" << std::endl;
    }
    if (answer == 0)
        heapSortUp(i, r);
    else
        heapSortDown(i, r);
    printArray(i, "It is our finally sort array.\n");
    memset(arr, 0, sizeof(int)*i);
    delete[] str;
}


int main() {
    cout << "Good time of day. It's a coursework written by a student of group
9382 Pya Son Khwa on the topic \"Sorting an N-nary heap\".\n";
    cout << "Before starting the work, I want to provide some explana-
tions.\nDuring operation, you will be given the option to select the input
stream of input value and the sifting option. The parent and sons that we are
currently considering will be highlighted in blue, and the ones that we are
swapping will be highlighted in red.\nTo demonstrate, click on any button...\n";
    char c;
    cin >> c;
    system("clear");
    do {
        char n, *name = new char[100]();
        cout << "What input stream would you like to use?\n(0 - from console, 1
- from file)\n";
        cin >> n;
        if (n == '0') {
            cout << "Write array of data! Write number of children!\n";
            while (cin.get() != '\n');
            fcn(std::cin);
        } else if (n == '1'){
            cout << "Write the name of file:\n";
            cin >> name;
            char *filename = new char[30]();//ввходные данные из файла
            cin.sync();
            strcpy(filename, "Tests//");
            strcat(filename, name);
            strcat(filename, ".txt");
            std::ifstream in(filename);
            if (!in.is_open()) {
                std::cout << "File wasn't opened!";
                delete[] name;
                delete[] filename;
                break;
            }
            fcn(in);
            delete[] name;
            delete[] filename;
        } else {
            cout << "Wrong digit of stream!\n";
        }
```

```
        cout << "Do you want to continue? (y/n)\n";
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        while (true) {
            cin >> c;
            if (c == 'y' || c == 'Y' || c == 'n' || c == 'N')
                break;
            std::cin.clear();
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        }
    } while (c == 'y' || c == 'Y');
    return 0;
}
```