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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «АиСД»

ТЕМА: n-нарная куча

Студентка гр. 9382 _____ Пя С.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Познакомиться с n-нарными кучами. Научиться представлять массив в виде n-нарной кучи на примере языка C++. Выполнить работу в соответствии с заданием.

Основные теоретические положения.

Куча — это специализированная структура данных типа дерево, которая удовлетворяет свойству кучи: если B является узлом-потомком узла A, то ключ(A) ≥ ключ(B).

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 × i + 1

2-й потомок: N × i + 2

3-й потомок: N × i + 3

…

N-й потомок: N × i + N

Задание

Вариант №29.

Дан массив чисел и число n (n=1, 2, 3, …). Предполагая, что массив является n-арной кучей: - Вывести его в виде n-арной кучи. - Получить путь от корня до листа такой, что при каждом шаге вниз выбирается наибольший сын.

Ход работы.

1) Разработан алгоритм:

На вход подается массив, который с помощью индексов можно представить в виде n-нарной кучи, затем задается число n. Массив выводится в виде этой кучи.
Как это происходит:

Все происходит с помощью 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. Следом 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)

Затем выводится путь от корня до листа, спускаясь вниз и выбирая наибольшего сына. Алгоритм выбирает из сыновей большего и выводит его на экран.

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 и 2)

2) 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. printRoot

Сигнатура: int printRoot(int n, int u).

Назначение: вывод пути от корня до листа с выбором наибольшего сына.

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

Возвращаемое значение: i : int – количество значений в массиве, содержащем искомый путь.

3. 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 ввода.

4. PrintFinallyRoot

Сигнатура: void printFinallyRoot(int* &finallyArr, int count).

Назначение: предназначена для окончательного вывода искомого пути.

Описание аргументов: finallyArr : int* - массив, содержащий искомый путь, count : int – количество элементов в этом массиве.

Реализация вывода пути от корня до листа с выбором наибольшего сына:

На вход подаются количество значений массива и максимальное количество сыновей. Выводится корень, затем из сыновей выбирается наибольший, после сравниваются его сыновья и так далее.

Пример работы программы.

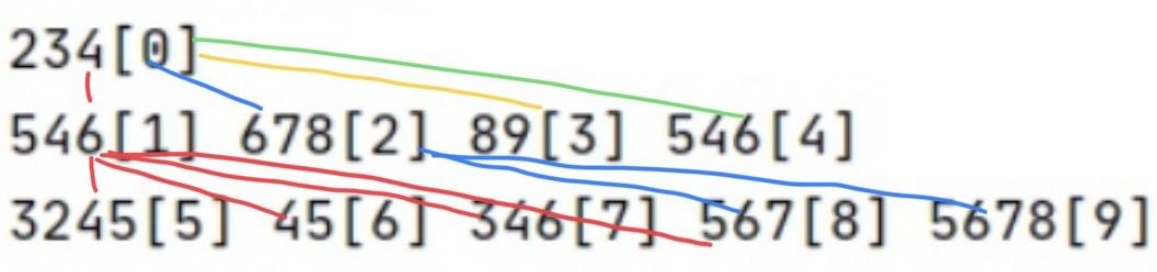

рис. 1

Таблица 1 – демонстрация работы программы.

Входные данные	Выходные данные
1 1 y 0 1 1 n	What input stream would you like to use? (0 - from console, 1 - from file) 1 Write the name of file: 1 N-narny heap: 9[0] 8[1] 7[2] 6[3] 5[4] 4[5] 3[6] 2[7] 1[8] Define max root: It is 1 level 9[0] Searching the biggest son from: 8 7 6 It is 2 level 8[1]

Входные данные	Выходные данные
	Searching the biggest son from: 5 4 3 It is 3 level 5[4] Max root is: 9 8 5 Do you want to countinue? (y/n) y What input stream would you like to use? (0 - from console, 1 - from file) 0 Write array of data! Write number of children! 1 1 N-narny heap: 1[0] Define max root: It is 1 level 1[0] Max root is: 1 Do you want to countinue? (y/n) n

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 2 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	0 87 56 34 23 23 2	N-narny heap: 87[0] 56[1] 34[2] 23[3] 23[4] Max root: 87 56 23	Проверка на корректность работы программы с терминалом
2.	1 6789 6544 3456 2345 4321 1234 987 654 543 234 211 87 56 45 34 23 4	N-narny heap: 6789[0] 6544[1] 3456[2] 2345[3] 4321[4] 1234[5] 987[6] 654[7] 543[8] 234[9] 211[10] 87[11] 56[12] 45[13] 34[14] 23[15] Max root: 6789 6544 1234	Проверка на корректность работы с файлом
4.	2 2	Error data You entered the wrong data!	Проверка на корректность работы с неверными данными
	3 1 1	N-narny heap: 1[0] Max root is: 1	Проверка на корректность работы с пограничными значениями

Выводы.

В ходе работы была освоена реализация n-нарной кучи, отработано понимание его применения, и отработаны навыки письма в C++.

Код программы можно найти в приложении А.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

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

#define NMAX 500
using namespace std;
int arr[NMAX+1];

void printHeap(int n, int u) {//предназначена для вывода в форме кучи
    int k = 1, i = 0, y = 0;
    while (true) {
    while (i < k + y) {
        cout << arr[i] << "[" << i << "] ";//вывод сыновей
        i++;
        if (i == n) {
            y = -1;
            break;
        }
    }
    cout << "\n";
        k *= u;
        if (y == -1)//выход из цикла
            break;
        y = i;
    }
}

int printRoot(int n, int u, int* &finallyArr) {//предназначена для вывода пути
от корня до листа с выбором наибольшего сына
    int main = 0, max = 1, l = 1, i = 0;
    cout << "It is " << l++ <<" level\n";
    cout << arr[0] << "[0]\n";
    finallyArr[i++] = arr[0];
    while (true) {
        max = main + 1;
        if (max >= n) {
            return i;
        }
        cout << "Searching the biggest son from:\n";
        for (int k = 1; k < u + 1; k++) {
            if (main + k < n)
                cout << arr[main + k] << " ";
            if (main + k < n && arr[main + k] >= arr[max]) {//выбор наибольшего
сына
                max = main + k;
            }
        }
        cout << "\nIt is " << l++ <<" level\n";
        cout << arr[max] << "[" << max << "]\n";//вывод сына
        finallyArr[i++] = arr[max];
        main = max * u;
    }
}
```

```
void printFinallyRoot(int* &finallyArr, int count) {//предназначена для окончательного вывода искомого пути
    for (int i = 0;i < count;i++) {
        std::cout << finallyArr[i] << " ";
    }
    cout << "\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;
    fin >> r;
    if (!r) {
        cout << "Error data\nYou entered the wrong data!\n";
        delete[] str;
        return;
    }
    cout << "N-narny heap:\n";
    printHeap(i, u);//вывод в форме кучи
    cout << "Define max root:\n";
    int* finallyArr = new int[i]();
    int count = printRoot(i, u, finallyArr);//вывод пути от корня до листа с выбором наибольшего сына
    cout << "Max root is:\n";
    printFinallyRoot(finallyArr, count);
    memset(arr, 0, sizeof(int)*i);
    delete[] str;
}


int main() {
    char c;
    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";
            n = cin.get();
            fcn(std::cin);
        } else {
            cout << "Write the name of file:\n";
            cin >> name;
            char *filename = new char[30]();//ввходные данные из файла
            strcpy(filename, "Tests//");
            strcat(filename, name);
            strcat(filename, ".txt");
            std::ifstream in(filename);
            if (!in.is_open()) {
                std::cout << "File wasn't opened!";
                return 0;
            }
            fcn(in);
            delete[] name;
            delete[] filename;
        }
        cout << "Do you want to countinue? (y/n)\n";
        c = getch();
    } while (c == 'y' || c == 'Y');
    return 0;
}
```