Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

ДЕРЕВЬЯ

Отчет по лабораторной работе №4

по дисциплине «Технологии и методы программирования»

Студент гр. 726-2

\_\_\_\_\_\_\_ Грохотова Е.А.

Принял

Кандидат технических наук

\_\_\_\_\_\_\_ Романов А. С.

\_\_\_\_\_\_\_\_\_\_

2017

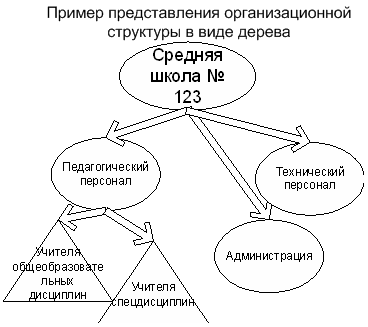
1. Введение

Цель работы: изучение теории и практическая работа с деревьями.

# Ход работы

1. Краткая теория

Дерево – это АТД (абстрактный тип данных) для хранения информационных элементов имеющих нелинейные отношения. За исключением элемента, который находится во главе дерева, каждый элемент имеет родителя, а также ноль или более дочерних элементов. Также говорят что дерево – это множество элементов, среди которых есть один выделенный который называется корнем, а остальные содержатся в N непересекающихся подмножествах, которые называются поддеревьями. Очевидно что дерево относится к классу динамических структур данных наряду с нам уже знакомыми стеком, очередями. С помощью дерева можно представить, например, следующую организационную структуру:

[](http://study-and-dev.com/blog/wp-content/uploads/mediawiki/images/28/sda_book_21.png)

Наиболее известным классическим примером деревьев является генеалогическое дерево.

Также в качестве примера можно привести принятую в большинстве файловых систем организацию с помощью иерархически вложенных директорий и файлов. Наиболее ярко это проявляется в О.С. linux и unix, где файловую систему представляют в виде единого дерева, которое начинается с корневой директории обозначаемой “/”.

При дальнейшем обсуждении деревьев следует условиться о терминологии. Корень дерева (root) – первый его элемент (в приведенном выше изображении это “Средняя школа # 123”). Каждый элемент данных называют узлом (node), иногда листом (leaf). Фрагмент дерева называется поддеревом (subtree) или ветвью. Узел, не имеющий отходящих от него ветвей ли поддеревьев, называются терминальным или конечным узлом (terminal node). Высота дерева (height) – определяется количеством уровней, на которых располагаются его узлы. При работе с деревьями полезно изображать их на бумаге, однако при этом всегда следует помнить, что деревья – это метод логической организации информации в памяти компьютера, не физической.

Следует помнить, что большинство операций, которые выполняются над деревьями, являются рекурсивными, т.к. деревья имеют рекурсивную структуру – каждое дерево содержит множество поддеревьев.

1. Задание

Задание:

1. Изучить теоретический материал по теме «Деревья».
2. При выполнение задания активно использоваться git
3. Реализовать программу, соответствующую Вашему варианту задания. Дерево реализовать в виде класса, необходимый функционал – в виде методов. Структура данных «дерево» должна быть реализована с помощью указателей.
4. В процессе реализации не забывать делать коммиты в github.
5. Написать отчет, залить его в MOODLE. Защитить отчет у преподавателя.

Индивидуальное задание (вариант 13)

Разработать программу, формирующую динамическую структуру данных для хранения генеалогического дерева. Каждая вершина дерева должна содержать следующую информацию: имя и год рождения.

1. Программа

#include "stdafx.h"

#include "fstream"

#include "string"

#include "queue"

#include "iostream"

using namespace std;

class Treee

{

public:

struct Node //узел

{

string Data;

Node \*left;

Node \*right;

int ID;

};

Node \*parent = nullptr;

Node \*root = nullptr;

int N = 0;

public:

Treee() {};

~Treee()

{

queue <Node\*> q;

q.push(root);

while (!q.empty())

{

Node \*fq = q.front();

if (fq->left != nullptr)

q.push(fq->left);

if (fq->right != nullptr)

q.push(fq->right);

delete fq;

q.pop();

}

parent = nullptr;

root = nullptr;

N = 0;

};

void Push(string D,bool gend)

{

N++;

Node \*node = new Node;

node->Data = D;

node->ID = N;

node->left = nullptr;

node->right = nullptr;

if (N == 1) // все кроме корня

{

root = node;

parent = root;

}

else

{

if (gend == 0)

{

parent->left = node;

if (N <= 3)

root->left = node;

}

else

{

parent->right = node;

if (N <= 3)

root->right = node;

}

}

};

bool Parent(string men)//ищет родителя

{

queue <Node\*> myQueue; //очередь, в ней будут храниться адреса ячеек дерева

if (root != nullptr)

{

Node ro = \*root;

myQueue.push(&ro); //первый элемент для сравнения - корень дерева

while (true)

{

if (myQueue.empty() == true) //если очередь пуста => человек не найден

return 0;

Node \*first = myQueue.front(); //работаем с первым элементом

if (first->Data == men)//человек найден

{

parent = first;

return 1;

}

else //первый элемент - не тот человек, которого мы ищем => удаляем его из очереди

myQueue.pop();

if (first->left != nullptr) //если есть левый, узел добавляем в очередь

myQueue.push(first->left);

if (first->right != nullptr) // то же, что и с левым

myQueue.push(first->right);

}

}

};

void Print(Node \*q, int level)

{

if (q)

{

Print(q->left, level + 1);

for (int i = 0; i < level; i++) cout << " ";

cout << q->ID << " " << q->Data << endl;

Print(q->right, level + 1);

}

};

};

int main()

{

setlocale(0, "");

Treee tr;

ifstream file("Family\_tree.txt");

if (!file)

{

cout << "Первая запись являеся корнем дерева. Напишите генеологическое дерево как в примере \nФИО и год рождения: Иванов Иван Иванович 2000\nМама: Иванова Лариса Олеговна 1990\nПапа: Иванов Иван Витальевич 1987\n";

system("Family\_tree.txt");

}

string men, mama, papa;

while (!file.eof())

{

getline(file, men);

men = men.substr(20);

getline(file, mama);

mama = mama.substr(6);

getline(file, papa);

papa = papa.substr(6);

if (tr.N == 0) //если не было записей, создаем корень

tr.Push(men, 1);

bool b = tr.Parent(men);

if (b == 1) //ищем человека

{

tr.Push(mama, 0); //записывам в дерево маму и папу

tr.Push(papa, 1);

}

else

cout << "Человек не найден, возможно вы не написали год\n";

}

file.close();

tr.Print(tr.root, 0);

tr.~Treee();

system("Pause");

return 0;

};

# Вывод

В процессе выполнения лабораторной работы были изучены деревья, а так же велась активная работа с системой контроля версий git.