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

Студент гр. 726-2

\_\_\_\_\_\_\_ Арестов М.О.

Принял

Кандидат технических наук

\_\_\_\_\_\_ \_\_\_\_\_\_\_ Романов А. С.

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

2017

# 1 Цель работы

Научиться работать со структурами данных – деверья.

1.2 Задание

Вариант 16. В непустом дереве Т найти длину пути (число ветвей) от корня до ближайшей вершины с элементом Е. Если элемент Е не входит в Т, то за ответ принять "-1". Дерево ЛЮБОЕ, не бинарное.

# 2 Ход работы

Код программы:

2.1 Main

#include "stdafx.h"

#include <iostream>

#include "Tree.h"

void main()

{

int n(5), leaf(2), i(1), j(0), data(0), number(0);

int a[] = {0,1,1,1,3,4,4,5,5,5};

int b[] = {0,1,3,4,5,9};

tree \*t = new tree(data);

data++;

while (i <= n)

{

leaf = b[i];

std::cout<<"\nLeaf(Level\_"<<i<<"): ";

//std::cin>>leaf;

std::cout<<leaf<<"\n";

while (j < leaf)

{

number = a[j+1];

std::cout<<"\nNumber(Level\_"<<i<<"): ";

//std::cin>>number;

std::cout<<number<<"\nData(Level\_"<<i<<"): ";

//std::cin>>data;

std::cout<<data;

t->add(i, number, data);

j++;

data++;

}

std::cout<<"\n";

j = 0;

i++;

}

t->print();

std::cout<<"\nWays to (10): "<<t->find(10)<<std::endl;

delete t;

system("pause");

}

2.2 Tree

#ifndef TREE\_H

#define TREE\_H

#include <iostream>

#include <iomanip>

#include <vector>

#include <vector>

#include <cstdlib>

class tree

{

int size, maxlevel;

struct derevo;

derevo \*root;

derevo \*leaf(derevo \*);

derevo \*findlevel(int, int \*, derevo \*);

derevo \*m\_print(int &, derevo \*, int &);

derevo \*\_find(const int &, int &, derevo \*d);

public:

tree(int);

~tree();

void add(int, int, int);

void print();

int find(const int &);

};

struct tree::derevo

{

int data;

int level;

derevo \*parent;

std::vector<derevo \*> child;

};

tree::tree(int data)

{

root = new derevo;

root->data = data;

root->level = 0;

root->parent = nullptr;

}

tree::~tree()

{

derevo \*n;

while (!root->child.empty())

{

n = leaf(root);

n->parent->child.pop\_back();

delete n;

}

delete root;

}

tree::derevo \*tree::findlevel(int level, int \*number, derevo \*node)

{

if (node->level == level - 1)

return node;

else

{

if (node->child.empty())

return nullptr;

for (auto it = node->child.begin(); it != node->child.end(); ++it)

{

derevo \*buff = findlevel(level, number, \*it);

if (buff != nullptr)

{

if ((\*it)->level == level - 1)

(\*number)--;

if (\*number == 0)

return buff;

}

}

}

}

void tree::add(int level, int number, int data)

{

int num(number);

derevo \*d = findlevel(level, &num, root);

if (!d)

{

std::cout<<"\n\nWrong number!\n";

system("pause");

exit(1);

}

derevo \*n = new derevo;

n->data = data;

n->level = level;

n->parent = d;

d->child.push\_back(n);

if (level>maxlevel) maxlevel = level;

}

tree::derevo \*tree::leaf(derevo \*d)

{

if (d->child.empty())

return d;

else

{

for (auto it = d->child.rbegin(); it != d->child.rend(); ++it)

{

derevo \*buff = leaf(\*it);

if (buff != nullptr)

return buff;

}

}

}

int tree::find(const int &k)

{

int l = maxlevel + 1;

auto n = \_find(k, l, root);

if (l > maxlevel) return -1;

return l;

}

tree::derevo \*tree::\_find(const int &k, int &l, derevo \*d)

{

if (d->data == k && l>d->level) l = d->level;

if (d->child.empty() && d == leaf (root)) return d;

if (d->child.empty())

return nullptr;

for (auto it = d->child.begin(); it != d->child.end(); ++it)

derevo \*buff = \_find(k, l, \*it);

}

tree::derevo \*tree::m\_print(int &k, derevo \*d, int &j)

{

if (d->level == k)

{

std::cout<<d->data<<"|";

if (k !=0 && \*--d->parent->child.end() == d) std::cout<<"|";

j++;

return d;

}

else

{

if (d->child.empty())

return nullptr;

std::cout<<"|";

for (auto it = d->child.begin(); it != d->child.end(); ++it)

derevo \*buff = m\_print(k, \*it, j);

}

}

void tree::print()

{

derevo \*buff;

int k(0), j(1);

while (k <= maxlevel)

{

std::cout<<std::setw(30-2\*j)<<"";

buff = m\_print(k, root, j);

std::cout<<"\n";

k++;

}

}

#endif

# 3 Вывод

В процессе выполнения лабораторной работы была получены навыки работы с деревьями, выполнено задание 16.