Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

«ДЕРЕВЬЯ»

Отчет по практической работе №4

по дисциплине «Методы программирования»

Выполнил:

Студент гр. 745

\_\_\_\_\_\_\_\_ Загреба М.А.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2017 г.

Проверил:

Доцент кафедры БИС

\_\_\_\_\_\_\_\_\_ Романов А.С.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_2017 г.

Томск 2017

1 Введение

Цель работы: изучить деревья – абстрактный тип данных, выполнить задание на практическую работу.

ЗАДАНИЕ

1. Изучить теоретический материал по теме «Деревья».

2. Создать проект в github

3. Реализовать программу, соответствующую Вашему варианту задания. Дерево реализовать в виде класса, необходимый функционал – в виде методов.

4. В процессе реализации не забывать делать коммиты в github.

5. Написать отчет, залить его в MOODLE.

6. Защитить отчет у преподавателя.

2 Ход работы

2.1 Дерево

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.

2.1.1 Реализация дерева

#pragma once

#include <iostream>

namespace program

{

template <class T>

class Tree

{

struct Node

{

T val;

Node \*left;

Node \*right;

Node() : left(0), right(0){};

};

Node \*root;

int n = 0;

public:

Tree()

{

root = NULL;

};

~Tree()

{

}

void add\_node()

{

T x;

cout << "Введите элемент дерева: ";

cin >> x;

add\_node(root, x);

}

void show()

{

if (root == 0)

{

cout << "Дерево не содержит элементов\n";

}

else

{

cout << "Элементы дерева: " << endl;

show(root);

cout << endl;

}

}

void del()

{

del(root);

cout << "Дерево удалено\n";

root = NULL;

}

void search()

{

T data;

cout << "\nВведите нужную строку: ";

cin >> data;

cout << endl;

if (search(root, data) == 0)

cout << "Такой строки нет \n" << endl;

else

cout << "Строка найдена \n" << endl;

}

private:

void add\_node(Node \*&node, const T &x)

{

if (node == NULL)

{

node = new Node;

node->val = x;

}

else

{

if (x.size() < node->val.size())

{

add\_node(node->left, x);

}

else

{

add\_node(node->right, x);

}

}

}

void del(Node \*node)

{

if (node)

{

del(node->left);

del(node->right);

delete node;

}

}

void show(Node \*&node)

{

if (node != NULL)

{

cout << node->val << " ";

show(node->right);

show(node->left);

}

}

int search(Node \*node, T data)

{

if (node->val == data) {

n++;

}

else

{

if (node->right != NULL)

search(node->right, data);

if (node->left != NULL)

search(node->left, data);

}

return n;

}

};

}

2.2 Индивидуальный вариант №6

Разработать рекурсивную функцию, которая определяет, помещена ли строка в бинарное дерево поиска.

#include <iostream>

#include <conio.h>

#include <string>

#include "Tree.h"

using namespace std;

using program::Tree;

int main()

{

setlocale(LC\_ALL, "Rus");

int height;

Tree <string> tree;

cout << "Введите высоту дерева: ";

cin >> height;

while (height)

{

tree.add\_node();

height--;

}

cout << endl;

tree.show();

tree.search();

system("pause");

return 0;

}

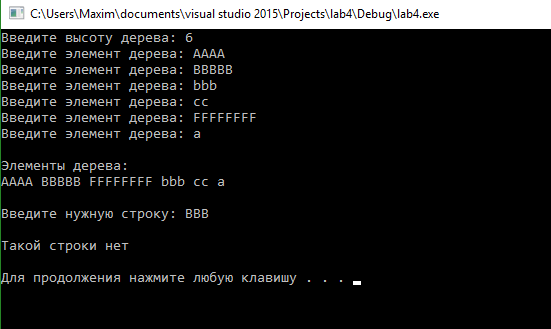


Рисунок 1 – Пример работы без нужной строки

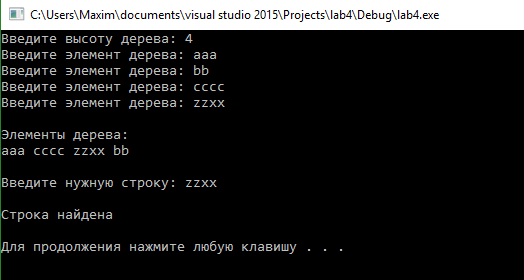


Рисунок 2 – Пример работы с найденной строкой

2.3 Работа с github