|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | |  | | | | | |
|  | | |
|  | Институт информационных технологий (ИТ) |
|  | Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ) |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4** | |
| **по дисциплине** | |
| **«Структуры и алгоритмы обработки данных»**  **По теме: «Двоичное бинарное дерево, бинарное дерево поиска, сбалансированное дерево»**  **Вариант 9(23)** | |
|  | |
| Выполнил студент группы ИВБО-06-17 | Ушакова А.С. |
| Принял преподаватель | Скворцова Л.А. |

Москва 2019

**Оглавление**

[1. Задание 1. 10](#_Toc26286335)

[1.1. Структура узла 10](#_Toc26286336)

[1.2. Таблица тестов 10](#_Toc26286337)

[1.3. Текст исходного кода (листинг) программы 10](#_Toc26286338)

[Листинг программы представлен в приложении 1. 10](#_Toc26286339)

[2. Задание 2. 10](#_Toc26286340)

[2.1. Вариант задания. 11](#_Toc26286346)

[2.2. Структура класса 11](#_Toc26286347)

[2.3. Таблица тестов 13](#_Toc26286395)

[2.4. Текст исходного кода (листинг) программы 13](#_Toc26286396)

[3. Контрольные прогоны программы 13](#_Toc26286398)

[Приложение 1 14](#_Toc26286401)

# **1. Задание 1.**

# Разработать класс бинарное дерево, включив в него методы: конструктор по умолчанию и вывод бинарного дерева, используя прямой обход дерева, разворачивая дерево горизонтально.

* 1. Структура узла

|  |
| --- |
| struct TreeEl |
|  | { |
|  | TreeEl \*left; |
|  | TreeEl \*right, \*parent; |
|  | int num; |
|  | string text; |
|  | TreeEl(int n = 0, string s = "", TreeEl\* l = 0, TreeEl\* r = 0):num(n), left(l), right(r), text(s){} |
|  | }; |

* 1. Таблица тестов

|  |  |  |
| --- | --- | --- |
| **Номер теста** | **Входные данные** | **Эталон результата** |
| **1** | Введите кол-во узлов в дереве  5  Введите узлы дерева:  Текст:1111  Кол-во цифра в тексте:4  Текст:текст  Кол-во цифра в тексте:0  Текст:55478  Кол-во цифра в тексте:5  Текст:55Анна  Кол-во цифра в тексте:2  Текст:8поросят  Кол-во цифра в тексте:1 | 5-55478  4-1111  2-55Анна  1-8поросят  0-текст |

* 1. Текст исходного кода (листинг) программы

Листинг программы представлен в приложении 1.

1. Задание 2.

Разработать производный класс для дерева определенного вариантом.

В класс включить методы:

* Конструктор с параметром (для сбалансированного дерева) – число узлов.
* Копирующий конструктор
* Деструктор
* Методы для операций, определенных в варианте.
  1. Вариант задания.

Информационная часть узла содержит текст и количество в нем цифр:

* Определите количество узлов, текст которых содержит максимальное число цифр.
* Удалить узел, не содержащий в тексте цифр.
* Обойти дерево в обратном порядке и вывести значения его узлов.
  1. Структура класса

class Tree

{

TreeEl \*root;

public:

//конструктор

Tree(){

root = nullptr;

}

//конструктор копирования

Tree(Tree& tree){

root = copyTree(tree.root);

}

//добавление узла - основа

void add(TreeEl \*&t, int n, string s);

//добавление узла

void addEl(int n, string s){

add(root, n, s);

}

//вывод дерева - основа

void inorder(TreeEl\* &t, int level);

//вывод дерева

void showtree(int level){

inorder(root, level);

}

//удаление узла - основа

TreeEl\* delNode(TreeEl\*& node);

//удаление узла

TreeEl\* del();

//проход в обратном порядке - основа

void revers(TreeEl\* &t);

//вывод дерева

void showtree1(){

revers(root);

}

//нахождение максимума

int maxEl(TreeEl\*& node);

//кол-во максимумов - основа

int maxNum(TreeEl\*& node, int key);

//кол-во максимумов

int numOfMax();

//удаление дерева

void Destroy(TreeEl\*&t);

//копирование

TreeEl\* copyTree(TreeEl\*& t);

//деструктор

~Tree();

};

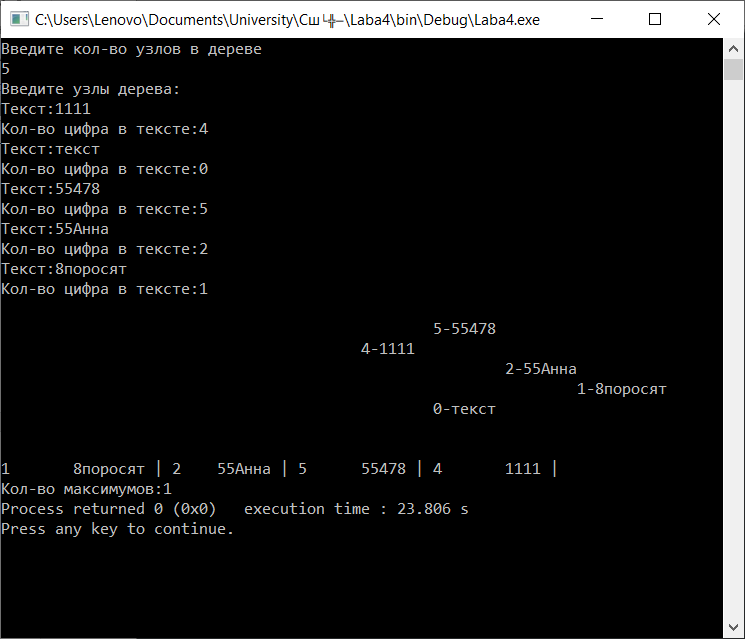
* 1. Таблица тестов

|  |  |  |
| --- | --- | --- |
| **Номер теста** | **Входные данные** | **Эталон результата** |
| **1** | Введите кол-во узлов в дереве  5  Введите узлы дерева:  Текст:1111  Кол-во цифра в тексте:4  Текст:текст  Кол-во цифра в тексте:0  Текст:55478  Кол-во цифра в тексте:5  Текст:55Анна  Кол-во цифра в тексте:2  Текст:8поросят  Кол-во цифра в тексте:1 | Определите количество узлов, текст которых содержит максимальное число цифр.  Кол-во максимумов:1  Удалить узел, не содержащий в тексте цифр.  Обойти дерево в обратном порядке и вывести значения его узлов.  1 8поросят | 2 55Анна | 5 55478 | 4 1111 | |

* 1. Текст исходного кода (листинг) программы

Листинг программы представлен в приложении 1.

1. Контрольные прогоны программы



Вывод

В результате выполнения лабораторной работы были получены навыки по созданию и работе с бинарными деревьями поиска.

Приложение 1

Tree.h

#ifndef TREE\_H

#define TREE\_H

#include <iostream>

#include <string>

#include <cmath>

using namespace std;

struct TreeEl

{

TreeEl \*left;

TreeEl \*right, \*parent;

int num;

string text;

TreeEl(int n = 0, string s = "", TreeEl\* l = 0, TreeEl\* r = 0):num(n), left(l), right(r), text(s){}

};

class Tree

{

TreeEl \*root;

public:

//конструктор

Tree(){

root = nullptr;

}

//конструктор копирования

Tree(Tree& tree){

root = copyTree(tree.root);

}

//добавление узла - основа

void add(TreeEl \*&t, int n, string s);

//добавление узла

void addEl(int n, string s){

add(root, n, s);

}

//вывод дерева - основа

void inorder(TreeEl\* &t, int level);

//вывод дерева

void showtree(int level){

inorder(root, level);

}

//удаление узла - основа

TreeEl\* delNode(TreeEl\*& node);

//удаление узла

TreeEl\* del();

//проход в обратном порядке - основа

void revers(TreeEl\* &t);

//вывод дерева

void showtree1(){

revers(root);

}

//нахождение максимума

int maxEl(TreeEl\*& node);

//кол-во максимумов - основа

int maxNum(TreeEl\*& node, int key);

//кол-во максимумов

int numOfMax();

//удаление дерева

void Destroy(TreeEl\*&t);

//копирование

TreeEl\* copyTree(TreeEl\*& t);

//деструктор

~Tree();

};

#endif // TREE\_H

Tree.cpp

#include "Tree.h"

//Прямой обход

void Tree::inorder(TreeEl\* &t, int level)

{

if ( t ) {

//движение по правой ветке

inorder( t->right, level+1);

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

//посещение узла

cout << t->num <<"-"<<t->text<<endl;

//движение по левой ветке

inorder( t->left, level+1);

}

else return;

}

//Добавление

void Tree::add(TreeEl \*&t, int n, string s){

//если корня дерева не существует

if(t == NULL){

//создать корень

t = new TreeEl(n,s);

}

else{

if(n < t->num){

//левый потомок

add(t->left, n,s);

}

else{

//правый потомок

add(t->right, n,s);

}

}

}

//Удаление элемента

TreeEl\* Tree::delNode(TreeEl\*& node){

//если нет корня

if(node == NULL)

return NULL;

//если значение в корне совпадает с искомым

if(node->num == 0){

//если у корня нет потомков

if(node->left == NULL && node->right == NULL) {

free(node);

return NULL;

}

//если нет левого потомка

if(node->right == NULL && node->left!=NULL){

TreeEl\* temp = node->left; //ставим на место корня правого потомка

free(node);

return temp;

}

//аналогично при отсутствии правого потомка

if(node->left == NULL && node->right!=NULL){

TreeEl\* temp = node->right;

free(node);

return temp;

}

//движемся по правой ветке

node->right = delNode(node->right);

return node;

}

//если значение в корне больше искомого

if(0<node->num){

node->left = delNode(node->left); //идем по левой ветке

}

//если значение в корне меньше искомого

if(0>node->num){

node->right = delNode(node->right); //идем по правой ветке

return node;

}

return node;

}

TreeEl\* Tree::del(){

int k = maxNum(root, 0);

while (k){

delNode(root);

k--;

}

}

//Обратный обход

void Tree::revers(TreeEl\* &t)

{

if ( t != NULL ) {

//движение по левой ветке

revers( t->left );

//движение по правой ветке

revers( t->right );

//посещение узла

cout << t->num <<"\t"<<t->text<< " | ";

}

else return;

}

//нахождение максимального узла

int Tree::maxEl(TreeEl\*& node) {

if (node == NULL) {

return 0;

}

int max\_l = max(maxEl(node->left), node->num);

int max\_r = max(maxEl(node->right), node->num);

return max(max\_l, max\_r);

}

//кол-во максимальных узлов

int Tree::maxNum(TreeEl\*& node, int key) {

if (node == NULL)

return 0;

if (key < node->num) {

return maxNum(node->left, key);

} else if (key > node->num) {

return maxNum(node->right, key);

} else {

return 1 + maxNum(node->right, key);

}

}

int Tree::numOfMax(){

return maxNum(root, maxEl(root));

}

//деструктор

Tree::~Tree(){

Destroy(root);

}

//удаление дерева

void Tree::Destroy(TreeEl\*& t){

delete t;

}

//копирование

TreeEl\* Tree::copyTree(TreeEl\*& t){

TreeEl \*newlptr, \*newrptr, \*newnode;

if(t==NULL)

return NULL;

if(t->left!=NULL)

newlptr=copyTree(t->left);

else

newlptr=NULL;

if(t->right!=NULL)

newrptr=copyTree(t->right);

else

newrptr=NULL;

newnode=new TreeEl(t->num, t->text, newlptr, newrptr);

return newnode;

}

main.cpp

#include <iostream>

#include "Tree.h"

#include <windows.h>

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Tree bt;

int n;

string str;

int lvl;

cout<<"Введите кол-во узлов в дереве\n";

cin>>lvl;

cout<<"Введите узлы дерева:\n";

for(int i = 0; i < lvl; i++){

cout<<"Текст:";

cin>>str;

cout<<"Кол-во цифра в тексте:";

cin>>n;

// int n = rand() % 20-10;

// string str = "текст";

bt.addEl(n, str);

}

cout << endl;

//вывод дерева

bt.showtree(lvl);

//удаление дерева

bt.del();

cout << endl;

cout << endl;

//обратный обход

bt.showtree1();

cout << endl;

cout<<"Кол-во максимумов:";

cout<<bt.numOfMax();

return 0;

}