**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Бинарные деревья

|  |  |  |
| --- | --- | --- |
| Студент гр. 6383 |  | Михайлов Ю.А. |
| Преподаватель |  | Шолохова О.М. |

Санкт-Петербург

2017

**Цель работы.**

Для заданного бинарного дерева b типа BT с произвольным типом элементов:

- подсчитать число узлов на заданном уровне n дерева b (корень считать узлом 1-го уровня)

**Содержательная постановка задачи.**

На входе программа получает последовательность символов из файла “KLP.txt”. Для решения поставленных задач используются рекурсивные процедуры.

Бинарное дерево - конечное множество узлов, которое либо пусто, либо состоит из корня и двух непересекающихся бинарных деревьев, называемых правым поддеревом и левым поддеревом. Так определенное бинарное дерево не является частным случаем дерева.

Определим скобочное представление бинарного дерева (БД):

* < БД > ::= < пусто > | < непустое БД >,
* < пусто > ::= /,
* < непустое БД > ::= ( < корень > < БД > < БД > ).
* Здесь пустое дерево имеет специальное обозначение - /.

Исходные данные представляют собой строку, содержащую элементы бинарного дерева. Результат работы программы представляет собой число – количество узлов на заданном уровне.

**Спецификация программы**

**Место и форма представления входных данных**: файл

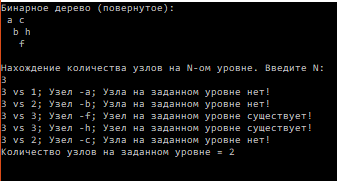
**Входные данные** : строка

**Выходные данные:** На экран выводится информация о количестве узлов на заданном уровне бинарного дерева

**Описание функций**

|  |  |
| --- | --- |
| Имя функции | Назначение |
| binTree enterBT(); | Считывает бинарное дерево из файла . |
| void displayBT(binTree b, int n | Функция вывода бинарного дерева в консоль. |
| unInt NumOfLvl(binTree b, int a, int c = 1) | Функция проверяющая существует ли узел на заданном уровне |

ПРИМЕР ДИАЛОГА С ПОЛЬЗОВАТЕЛЕМ



Тестирование

|  |  |  |
| --- | --- | --- |
| № опыта | Входные данные | Выходные данные |
| 1 | (a(b(f//)(h//))(c//))  2 | Количество узлов на заданном уровне = 2 |
| 2 | (//) | Пустое БД |
| 3 | (a//)  1 | Количество узлов на заданном уровне = 1 |
| 4 | (a(b//)/)  2 | Количество узлов на заданном уровне = 1 |
| 5 | (a(b//)(c//))  2 | Количество узлов на заданном уровне = 2 |

Код программы

**main.cpp**

#include <iostream>

#include <fstream>

#include <cstdlib>

#include "BTree.h"

#include <windows.h>

#include <conio.h>

#include <algorithm>

using namespace std;

using namespace binTree\_modul;

typedef unsigned int unInt;

binTree enterBT();

void displayBT(binTree b, int n);

unInt NumOfLvl(binTree b, int a, int c = 1);

ifstream infile("KLP.txt");

int main()

{

binTree b;

SetConsoleCP(1251); // для вывода кирилицы

SetConsoleOutputCP(1251); // для вывода кирилицы

b = enterBT();

if (isNull(b)) cout << "Пустое БД" << endl;

else

{

cout << "Бинарное дерево (повернутое): " << endl;

displayBT(b, 1);

cout << "\nНахождение количества узлов на N-ом уровне. Введите N:" << endl;

int num;

cin >> num;

cout << "Количество узлов на заданном уровне = " << NumOfLvl(b, num);

destroy(b);

}

\_getch();

}

binTree enterBT()

{

char ch;

binTree p, q;

infile >> ch;//считывание из файла

while (ch == '(' || ch == ')') infile >> ch;//проверка на скобки

if (ch == '/') { return NULL; }//проверка на пустой элемент

else { //иначе рекурсивная запись элементов

p = enterBT();

q = enterBT();

return ConsBT(ch, p, q);

}

}

void displayBT(binTree b, int n) //Для печати повернутого дерева

{ // n - уровень узла

if (b != NULL)

{

cout << ' ' << RootBT(b); //Проверка элемента

if (!isNull(Right(b)))

displayBT(Right(b), n + 1); //Если правая ветка не пустая, то печатаем элементы

else cout << endl; // спускаемся вниз

if (!isNull(Left(b))) //Если левая ветка не пустая, то печатаем элементы

{

for (int i = 1; i <= n; i++)

cout << " "; // вправо

displayBT(Left(b), n + 1);

}

}

else {};

}

unInt NumOfLvl(binTree b, int lvl, int lvl\_now)

{

if (b != NULL && lvl >= lvl\_now) //Если элемент не равен нулю и уровень больше текущего

cout << lvl << " vs " << lvl\_now << "; Узел -" << b->info << ";"; //то этот элемент - узел

{

if (lvl == lvl\_now) cout << " Узел на заданном уровне существует!\n"; //есл уровень равен текущему уровню

else cout <<" Узла на заданном уровне нет!"<< endl;

}

if (lvl == lvl\_now) return 1;

else if (lvl >= lvl\_now)

return(((b->lt) ? NumOfLvl(b->lt, lvl, lvl\_now + 1) : 0) + ((b->rt) ? NumOfLvl(b->rt, lvl, lvl\_now + 1) : 0)); //Подсчет кол-ва узлов

}

**BTree.h**

namespace binTree\_modul

{

typedef char base;

struct node {

base info;

node \*lt;

node \*rt;

// constructor

node() { lt = NULL; rt = NULL; }

};

typedef node \*binTree; // "представитель" бинарного дерева

binTree Create(void);

bool isNull(binTree);

bool RooBT(binTree);

base RootBT(binTree); // для непустого бин.дерева

binTree Left(binTree);// для непустого бин.дерева

binTree Right(binTree);// для непустого бин.дерева

binTree ConsBT(const base &x, binTree &lst, binTree &rst);

void destroy(binTree&);

}

**bt\_func.cpp**

// Implementation - Реализация АТД "Бинарное дерево" (в процедурно-модульной парадигме)

#include <iostream>

#include <cstdlib>

#include "Btree.h"

using namespace std;

namespace binTree\_modul

{

binTree Create() { return NULL; }

bool isNull(binTree b) { return (b == NULL); } //Проверяет не лустое ли дерево

base RootBT(binTree b) // для непустого бин.дерева

{

if (b == NULL) { cerr << "Error: RootBT(null) \n"; exit(1); } //Если пусто

else return b->info; //Если нет, возврат данных элемента

}

bool RooBT(binTree b)//Проверка элемента

{

if (b == NULL) return false; //если пустой

else return true; //Если не пуст

}

binTree Left(binTree b) // для непустого бин.дерева

{

if (b == NULL) { cerr << "Error: Left(null) \n"; exit(1); }

else return b->lt;

}

binTree Right(binTree b) // для непустого бин.дерева

{

if (b == NULL) { cerr << "Error: Right(null) \n"; exit(1); }

else return b->rt;

}

binTree ConsBT(const base &x, binTree &lst, binTree &rst)

{

binTree p;

p = new node;

if (p != NULL)

{

p->info = x;

p->lt = lst;

p->rt = rst;

return p;

}

else { cerr << "Memory not enough\n"; exit(1); }

}

void destroy(binTree &b)

{

if (b != NULL)

{

destroy(b->lt);

destroy(b->rt);

delete b;

b = NULL;

}

}

}