**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 |  | Медведев Г.О. |
| Преподаватель |  | Шолохова О.М. |

Санкт-Петербург

2017

**Цель работы.**

Познакомиться с такой часто используемой на практике, особенно при решении задач кодирования и поиска, нелинейной структурой данных, как бинарное дерево, способами её представления и реализации, получить навыки решения задач обработки бинарных деревьев.

**Содержательная постановка задачи.**

На входе программа получает последовательность символов из файла “DEREVO.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 выдает элементы из всех листьев дерева.

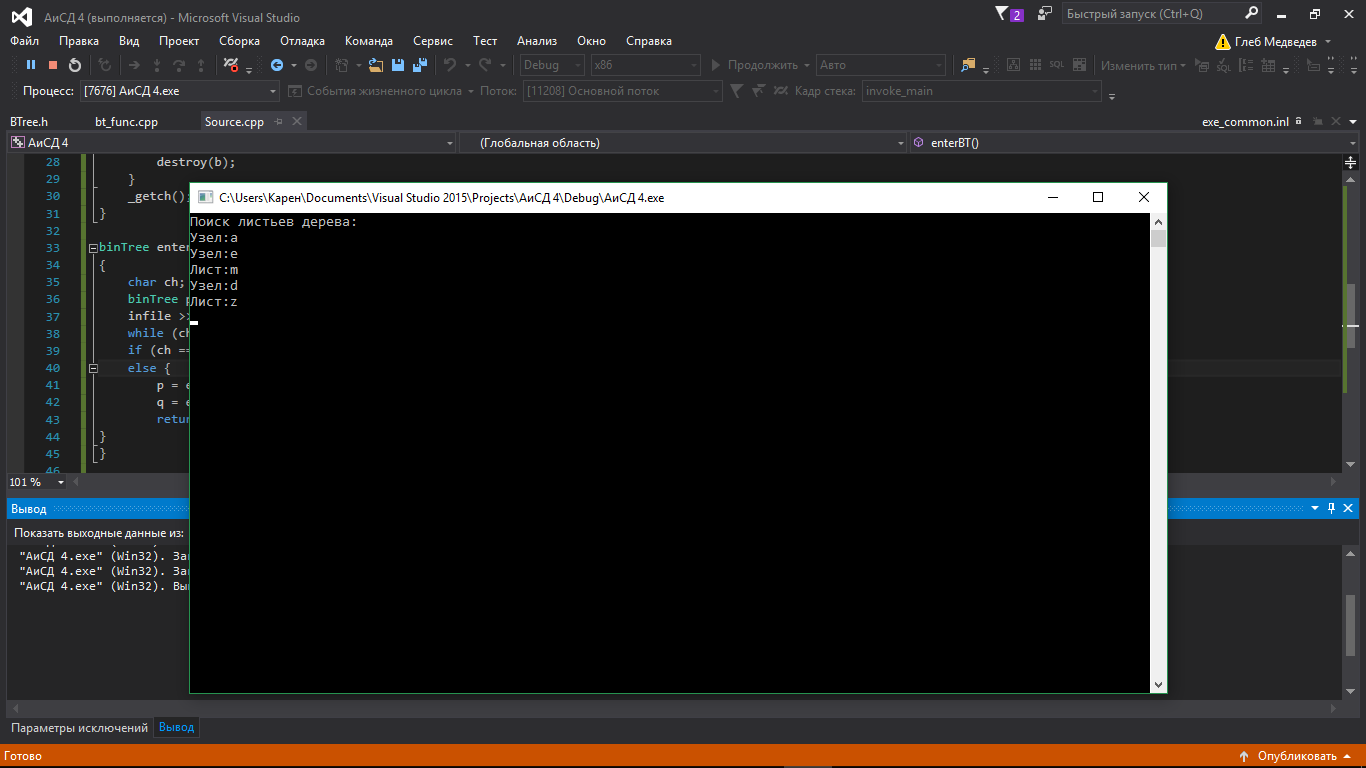
**Спецификация программы.**

Программа получает на вход строку из файла, производится обработка входной последовательности символов, в результате на экран выводятся листья введённого дерева.

**Описание функций.**

|  |  |
| --- | --- |
| Имя функции | Назначение |
| binTree enterBT(); | Считывает бинарное дерево из файла |
| void Findleaf(binTree b) | Функция вывода листьев дерева |

Пример диалога с пользователем.



Тесты.

|  |  |  |
| --- | --- | --- |
| № опыта | Входные данные | Выходные данные |
| 1 | (//) | Пустое БД |
| 2 | (a(e(m//())(/))(d()/(z//))) | Лист: m  Лист: z |
| 3 | (a(b//)(n(k(x//)(v//))(c//))) | Лист: b  Лист: x  Лист: v  Лист: c |
| 4 | (a(b//)(c//)) | Лист: b  Лист: c |

Вывод.

В ходе выполнения лабораторной работы были получены навыки представления и реализации бинарного дерева, а также удалось решить поставленную задачу: определить листья бинарного дерева.

Приложение A. Код программы

**Source.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 Findleaf(binTree b);

ifstream infile("DEREVO.txt");

int main()

{

binTree b;

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

b = enterBT();

int depth = 1;

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

else

{

cout << "Поиск листьев дерева:" << endl; Findleaf(b);

destroy(b);

}

\_getch();

}

binTree enterBT()

{

char ch;

binTree p, q;

infile >> ch;

while (ch == '(' || ch == ')') infile >> ch;

if (ch == '/') { return NULL; }

else {

p = enterBT();

q = enterBT();

return ConsBT(ch, p, q);

}

}

void Findleaf(binTree b)

{

if (!isNull(b))

{

if (!(isNull(b->lt) && isNull(b->rt)))

cout << "Узел:" << b->info << endl;

Findleaf(b->lt);

if (isNull(b->lt) && isNull(b->rt))

cout << "Лист:" << b->info << endl;

Findleaf(b->rt);

}

}

**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&);

} // end of namespace binTree\_modul

**bt\_func.cpp**

#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;

}

}

}