**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: Программирование алгоритмов с бинарными деревьями**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6383 |  | Азаревич А.Д. |
| 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.

**Постановка задачи.**

Задание:

Задано бинарное дерево *t* типа *BT* с типом элементов *Elem*. Для введенной пользователем величины *x* (**var** *x*: *Elem*) определить, входит ли элемент *x* в дерево *t*.

**Основные теоретические положения.**

Бинарное дерево － это конечное множество узлов, которое либо пусто, либо состоит из корня и двух непересекающихся бинарных деревьев, называемых правым поддеревом и левым поддеревом.

Примером может служить структура на рисунке 1.

*a*

*c*

*b*

*f*

*d*

*h*

*e*

*j*

*g*

*i*

*k*

*l*

Рис. 1 - Пример бинарного дерева

Скобочное представление бинарного дерева (БД):

< БД > ::= < пусто > | < непустое БД >,

< пусто > ::= Λ,

< непустое БД > ::= ( < корень > < БД > < БД > ).

Скобочное представление дерева на рисунке 4.2 выглядит следующим образом:

(a (b (d Λ(h Λ Λ))(e Λ Λ))(c (f (i Λ Λ)(j Λ Λ))(g Λ(k (l Λ Λ) Λ))))

Применив правила упрощения скобочной записи:

1.(<корень> <непустое БД> Λ) ≡ (<корень> <непустое БД>),

2. (< корень > Λ Λ) ≡ (< корень >),

получим (a (b (d Λ (h)) (e)) (c (f (i) (j)) (g Λ (k (l )))))

Во многих алгоритмах работы с деревьями используется понятие обхода (посещения узлов) бинарного дерева, дерева и леса. В основном используются 3 порядка обхода: прямой (сверху вниз), обратный (слева направо) и концевой (снизу вверх).

Рекурсивные процедуры обхода бинарного дерева в каждом из этих порядков приведены ниже.

void обходКЛП (BTree b)

{ // прямой порядок обхода

if ( ! Null (b) )

{

посетить (RootBT (b));

обходКЛП (Left (b));

обходКЛП (Right (b));

}

} // обходКЛП

Буква К обозначает корень, Л － левое поддерево, П － правое поддерево.

**Спецификация программы.**

*Назначение программы*.

Программа предназначена для определения наличия заданного элемента в заданном бинарном дереве.

*Описание программы*.

Программа написана на языке С++ с использованием компилятора ml64 9.0.30729.1. Входными данными для программы является строка из файла test.txt, представляющая собой прямую запись бинарного дерева («пустой» элемент обозначается символом ‘/’), а также элемент, который ищется в этом дереве. Проверка входных данных из файла выражается в следующем виде: если работа функции по постройке бин. дерева не завершила свою работу, а уже встретился конец файла, то выдаётся сообщение об ошибке и программа завершает свою работу (так же есть проверка на наличие самого файла). Выходными данными является информация, есть ли данный элемент в списке или нет.

*Пример диалога с пользователем*.

Задано бинарное дерево:

-a

-b

-d

-e

-c

-f

Элемент a в заданном бинарном дереве

присутствует

Элемент z в заданном бинарном дереве

отсутствует

*Реализация*.

Классы:

1. BT – класс, представляющий собой дерево.

Переменные класса:

1. BT \*left – указатель на левое бинарное поддерево;
2. BT \*right – указатель на правое бинарное поддерево;
3. Elem cell – значение, содержащееся в данном элементе бинарноо дерева (Elem = char).

Функции класса:

1. BT(std::ifstream &in);

Конструктор класса. Вызывает функцию Enter(in), где in – поток ввода.

1. ~Queue();

Деструктор класса. Вызывает деструктор для левого и правого поддерева.

1. void Enter(std::ifstream &in);

«Считывает» новый элемент в поддерево;

Входные параметры:

Поток ввода in.

Возвращаемое значение:

Отсутствует;

Предусловие:

Если встречен конец файла программа завершается;

Постусловие:

Если следующий элемент не равен ‘/’ создаёт новый элемент класса BT и присваивает его адрес left (аналогично строится правое поддерево).

1. bool isNull(BT \*t);;

Определяет, пусто ли бинарное поддерево t.

Входные параметры:

Указатель на бинарное поддерево;

Возвращаемое значение:

True, если t==NULL, или false, если t!=NULL

1. void Display(int n)

Печатает n пробелов и содержимое cell «действующего» элемента;

Входные параметры:

Номер «высоты» элемента;

Постусловие:

Если left!=NULL, вызывается Display для левого поддерева (аналогично для правого).

1. bool Found(Elem const &x)

Проверяет равенство cell==x (ищет х в бинарном дереве);

Входные параметры:

Искомый элемент x;

Выходное значение:

True, если x является элементом бинарного дерева, false – н является;

Постусловие:

Если в корне или в одном из поддеревьев выполняется cell==x возвращает true; если оба поддерева пусты или ни в одном из них нет x возвращает false.

**Тестирование.**

В файле test.txt:

abdh/o///e/l//cfmp///n//g/o//

|  |  |
| --- | --- |
| x | Результат |
| a | присутствует |
| o | присутствует |
| z | отсутствует |
| 5 | отсутствует |

**Выводы.**

В ходе выполнения лабораторной работы по программированию алгоритмов с бинарными деревьями мы познакомились с такой нелинейной структурой данных, как бинарное дерево.

Так же мы освоили прямой обход бинарного дерева (КЛП), получили навыки по поиску элементов в бинарном дереве.

**Приложение А.**

**Исходный код.**

BT.h

#include <iostream>

#include <fstream>

typedef char Elem;

void GLOBAL\_ERRORS(int n = -1);

class BT

{

public:

BT(std::ifstream &in);

~BT();

void Display(int n);

bool Found(Elem const &x);

private:

BT \*left;

BT \*right;

Elem cell;

bool isNull(BT \*t);

void Enter(std::ifstream &in);

};

BT.cpp

#include "BT.h"

BT::BT(std::ifstream &in)

{

Enter(in);

}

void BT::Enter(std::ifstream &in)

{

Elem x;

if(in.eof())

GLOBAL\_ERRORS(0);

in >> x;

cell = x;

if (in.peek() != '/' )

left = new BT(in);

else

{

left = NULL;

in.get();

}

if (in.peek() != '/' )

right= new BT(in);

else

{

right = NULL;

in.get();

}

}

BT::~BT()

{

if( !isNull(left) )

left->~BT();

if( !isNull(right) )

right->~BT();

}

bool BT::isNull(BT \*t)

{

return (t == NULL);

}

void BT::Display(int n)

{

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

std::cout <<" ";

std::cout << "-" << cell << std::endl;

if( !isNull(left) )

left->Display(n+1);

if( !isNull(right) )

right->Display(n+1);

}

bool BT::Found(Elem const &x)

{

if( cell == x ) return true;

if( !isNull(left) )

if( left->Found(x) ) return true;

if( !isNull(right) )

if( right->Found(x) ) return true;

return false;

}

main.cpp

//

// Лабораторная работа №4

// main.cpp

// Азаревич артём, группа 6383

// 30.10.2017

//

// Программа для поиска элемента х в бинарном дереве b

#include <windows.h>

#include "BT.h"

using namespace std;

int main ()

{

SetConsoleCP(1251); // для вывода кирилицы

SetConsoleOutputCP(1251); // для вывода кирилицы

std::ifstream fin("test.txt");

if(!fin)

GLOBAL\_ERRORS(2); //cerr: тестовый файл не найден

BT b(fin);

std::cout <<"Задано бинарное дерево:\n";

b.Display(1);

Elem x;

while( true )

{

std::cout << "Элемент ";

std::cin >> x;

cout <<"в заданном бинарном дереве\n";

if ( b.Found(x) )

std::cout << "присутствует\n\n";

else

std::cout << "отсутствует\n\n";

}

}

void GLOBAL\_ERRORS(int n)

{

switch(n)

{

case 0: cerr << "Ошибка: Бин. дерево не закончно\n"; break;

case 1: cerr << "Ошибка: Дерево пусто\n"; break;

case 2: cerr << "Ошибка: Тестовый файл не найден\n"; break;

default: return;

}

exit(1);

}