# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №3**

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

**Тема: Бинарные деревья**

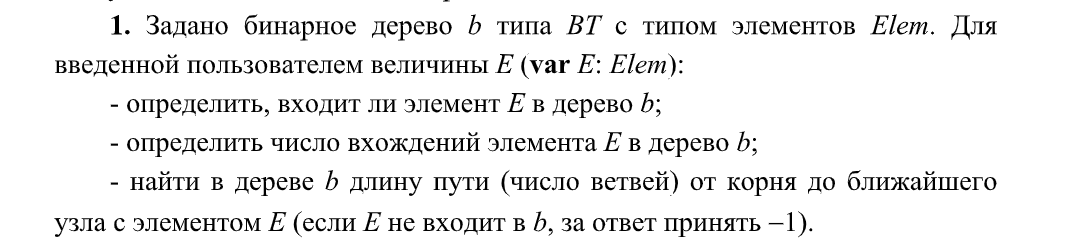
|  |  |  |
| --- | --- | --- |
| Студент гр. 9381 |  | Авдеев И. |
| Преподаватель |  | Фирсов М.А. |

# Цель работы.

Изучить структуру — бинарное дерево и работу с ним.

# Задание.

# Варинат 1



# Выполнение задания

1. Был создан тип данных, описывающий элемент бинарного дерева

**Класс elem**

Поля класса elem

**Int num public** - номер узла

**Int weight public** - вес узла

**struct elem\* left public** - указатель в левое плечо

**struct elem\* right public** - указатель в правое плечо

2. Для создания списка была создана функция push, создающая узел и соединяющая его к другим.

**void push(int a, elem\*\* t);**

3. Функции для вывода

**void printTree(elem\* t, int u);**

4. Функции инициализации из файла или из консоли

**void console(elem\*\* t);**

**void file(elem\*\* t);**

5. Функция удаления структуры

**void deleteCor(elem\* cor);**

6. Для проверки наличия элемента E в дерево создана функция

**bool isContains(elem\* t, int E);**

7. Для поиска количества элементов E в дереве создана функция

**int searchElem(elem\* t, int E);**

8. Для поиска длины пути до ближайшего узла содержащего элемент E создана функция

**int path(elem\* t, int E);**

**Описание алгоритма функции isContains(elem\* t, int E)**

Сначала проводится проверка наличия элемента структуры. Если ее нет, то функция возвращает 0.

Если содержимым узла является элемент E возвращаем 1.

Создается возвращаемое значение r типа bool равное 0;

Далее проводится сравнение содержимого узла с элементом E, если оно больше или равно элементу r будет равна запуску функции на левом плече, иначе r будет равна запуску функции на правом плече

После чего функция возвращает результат.

**Описание алгоритма функции int searchElem(elem\* t, int E)**

Сначала проводится проверка наличия элемента структуры. Если ее нет, то функция возвращает 0.

Создается возвращаемое значение res типа int равное 0;

Проводится сравнение содержимого узла с элементом Е, если оно больше Е к res прибавляется значение функции в левом плече, иначе к res прибавляется значение функции в правом плече

# После чего функция возвращает результат.

**Описание алгоритма функции int path(elem\* t, int E)**

Создается возвращаемое значение res типа int равное 0;

Проводится проверка наличия элемента структуры. Если ее нет, то функция возвращает -1.

# Проводится сравнение содержимого узла с элементом Е. Если оно равно Е возращает 0. Если он больше содержимого результат будет равен сумме res увеличенного на один и запуску функции в правом плече, иначе результат будет равен сумме res увеличенного на один и запуску функции в левом плече.

# **Тестирование функции int path(elem\* t, int E)**

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемый результат | Результат |
| 5  3  1  2  4  5 | 2 |  |
| 1  2  2 | 0 |  |
| 1  2  f  2 | 0 |  |
| 6  5  4  7  6  8  9  10 | -1 |  |

**Тестирование функции searchElem(elem\* t, int E)**

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемый результат | Результат |
| 15  6  4  8  3  5  7  9  1  2  10  4  6  2  1  3  1 | 2 |  |
| 1  f  1  2 | 0 |  |
| 0  2 | 0 |  |

**Тестирование функции isContains(elem\* t, int E)**

|  |  |  |
| --- | --- | --- |
| Входные данные | Ожидаемый результат | Результат |
| 15  6  4  8  3  5  7  9  1  2  10  4  6  2  1  3  1 | 1 |  |
| 2  1  f  4  1 | 1 |  |
| 2  4  6  1 | 0 |  |

**Выводы**

Были изучены бинарные деревья, в ходе были написаны функции проверки наличия, поиска количества вхождений и поиска длинны пути до ближайшего узла содержащего элемент Е.

**ПРИЛОЖЕНИЕ А**

**СОДЕЛЖАНИЕ ФАЙЛА Source.cpp**

#include <iostream>

#include <fstream>

#include <conio.h>

using namespace std;

int fnum = 0;

class elem

{

public:

int num;

int E;

elem\* left = NULL;

elem\* right = NULL;

};

void push(int a, elem\*\* t)

{

if ((\*t) == NULL)

{

(\*t) = new elem;

(\*t)->num = ++fnum;

(\*t)->E = a;

(\*t)->left = (\*t)->right = NULL;

return;

}

if (a > (\*t)->E) push(a, &(\*t)->right);

else push(a, &(\*t)->left);

}

void printTree(elem\* t, int u)

{

if (t == NULL) return;

else

{

printTree(t->right, ++u);

for (int i = 0; i < u - 1; ++i) cout << "\t";

cout << t->num << "(" << t->E << ")\n";

u--;

}

printTree(t->left, ++u);

}

bool isContains(elem\* t, int E)

{

if (t == NULL)

return 0;

if (t->E == E)

return 1;

bool r;

if (t->E > E)

r = isContains(t->left, E);

else

r = isContains(t->right, E);

return r;

}

bool isContainslog(elem\* t, int E)

{

int f = ++fnum;

cout << f << "-й запуск функции\n";

if (t == NULL)

{

cout << "плечо отсутствует возвращается назад\n";

cout << f << "-й запуск функции завершился\n";

\_getch();

return 0;

}

cout << "находится в узле " << t->num << "\n";

if (t->E == E)

{

cout << "узел найден, возвращается назад\n";

cout << f << "-й запуск функции завершился\n";

\_getch();

return 1;

}

else

\_getch();

bool r;

if (t->E >= E)

{

cout << "перемещается в левое плечо\n";

\_getch();

r = isContainslog(t->left, E);

cout << "вернулся в узел " << t->num << "\n";

}

else

{

cout << "перемещается в правое плечо\n";

\_getch();

r = isContainslog(t->right, E);

cout << "вернулся в узел " << t->num << "\n";

}

cout << f << "-й запуск функции завершился\n";

\_getch();

return r;

}

int searchElem(elem\* t, int E)

{

if (!t)

return 0;

int res = 0;

if (t->E == E)

res = 1;

if (t->E >= E)

res += searchElem(t->left, E);

else

res += searchElem(t->right, E);

return res;

}

int searchElemlog(elem\* t, int E)

{

int f = ++fnum;

cout << f << "-й запуск функции\n";

if (!t)

{

cout << "плечо отсутствует возвращается назад\n";

cout << f << "-й запуск функции завершился\n";

\_getch();

return 0;

}

cout << "находится в узле " << t->num << "\n";

int res = 0;

if (t->E == E)

{

cout << "узел содержит элемент\n";

\_getch();

res = 1;

}

else

\_getch();

if (t->E >= E)

{

cout << "перемещается в левое плечо\n";

\_getch();

res += searchElemlog(t->left, E);

cout << "вернулся в узел " << t->num << "\n";

}

else

{

cout << "перемещается в правое плечо\n";

\_getch();

res += searchElemlog(t->right, E);

cout << "вернулся в узел " << t->num << "\n";

}

cout << f << "-й запуск функции завершился\n";

\_getch();

return res;

}

int path(elem\* t, int E)

{

int res = 0;

if (!t)

return -1;

if (t->E == E)

return 0;

if (t->E < E)

{

int r = path(t->right, E);

return r == -1 ? -1 : ++res + r;

}

else

{

int r = path(t->left, E);

return r == -1 ? -1 : ++res + r;

}

}

int pathlog(elem\* t, int E)

{

int f = ++fnum;

cout << f << "-й запуск функции\n";

int res = 0;

if (!t)

{

cout << "плечо отсутствует -> элемент отсутствует, возвращается назад\n";

cout << f << "-й запуск функции завершился\n";

\_getch();

return -1;

}

else

\_getch();

cout << "находится в узле " << t->num << "\n";

if (t->E == E)

{

cout << "узел содержит элемент\n";

\_getch();

return 0;

}

else

\_getch();

if (t->E < E)

{

cout << "перемещается в правое плечо\n";

\_getch();

int r = pathlog(t->right, E);

cout << "вернулся в узел " << t->num << "\n";

cout << f << "-й запуск функции завершился\n";

\_getch();

return r == -1 ? -1 : ++res + r;

}

else

{

cout << "перемещается в левое плечо\n";

\_getch();

int r = pathlog(t->left, E);

cout << "вернулся в узел " << t->num << "\n";

cout << f << "-й запуск функции завершился\n";

\_getch();

return r == -1 ? -1 : ++res + r ;

}

}

void file(elem\*\* t)

{

int n;

int s;

ifstream lin("lin.txt");

lin >> n;

for (int i = 0; i < n; ++i)

{

lin >> s;

push(s, &(\*t));

}

fnum = 0;

lin.close();

}

void console(elem\*\* t)

{

while(1)

{

char nin, sin;

int n;

int s;

cout << "Введите количество элементов\n";

cin >> nin;

if (isdigit(nin))

{

n = atoi(&nin);

for (int i = 0; i < n; ++i)

{

cout << "Введите число\n";

cin >> sin;

if (isdigit(sin))

s = atoi(&sin);

else

{

cout << "неправильный ввод, попробуйте еще\n";

i--;

continue;

}

push(s, &(\*t));

}

return;

}

else

cout << "неправильный ввод, попробуйте еще\n";

}

fnum = 0;

}

void deleteCor(elem\* cor)

{

if (cor != NULL) {

deleteCor(cor->left);

deleteCor(cor->right);

delete cor;

}

}

int main()

{

setlocale(LC\_ALL, "ru");

elem\* t = NULL;

char k;

cout << "Ввод из консоли на кнопку С, из файла на любую другую\n";

k = \_getch();

switch(k)

{

case 'c':

console(&t);

break;

default:

file(&t);

}

fnum = 0;

printTree(t, 0);

int E;

while (1)

{

cout << "Введите элемент \n";

cin >> k;

if (isdigit(k))

E = atoi(&k);

else

{

cout << "Нужно ввести число\n";

continue;

}

cout << "Выяснить есть ли он в дереве на кнопку 1, Выяснить количество элементов в дереве 2, найти путь до близжаешего узла с элементов любая другая кнопка\n";

k = \_getch();

switch (k)

{

case '1':

cout << "пошаговый метод на кнопку 1, обычный на любую другую\n";

k = \_getch();

switch (k)

{

case '1':

cout << isContainslog(t, E);

break;

default:  
 cout << isContains(t, E);  
 break;

}

break;

case '2':

cout << "пошаговый метод на кнопку 1, обычный на любую другую\n";

k = \_getch();

switch (k)

{

case '1':

cout << searchElemlog(t, E);

break;

default:  
 cout << searchElem(t, E);  
 break;

}

break;

default:

cout << "пошаговый метод на кнопку 1, обычный на любую другую\n";

k = \_getch();

switch (k)

{

case '1':

cout << pathlog(t, E);

break;

default:  
 cout << path(t, E);  
 break;

}

break;

}

break;

}

deleteCor(t);

return 0;

}