Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

**РЕФЕРАТ**

на тему:

“Очередь с приоритетом”

Выполнил:

Халалеенко Андрей Николаевич

Проверил:

Белодед Николай Иванович

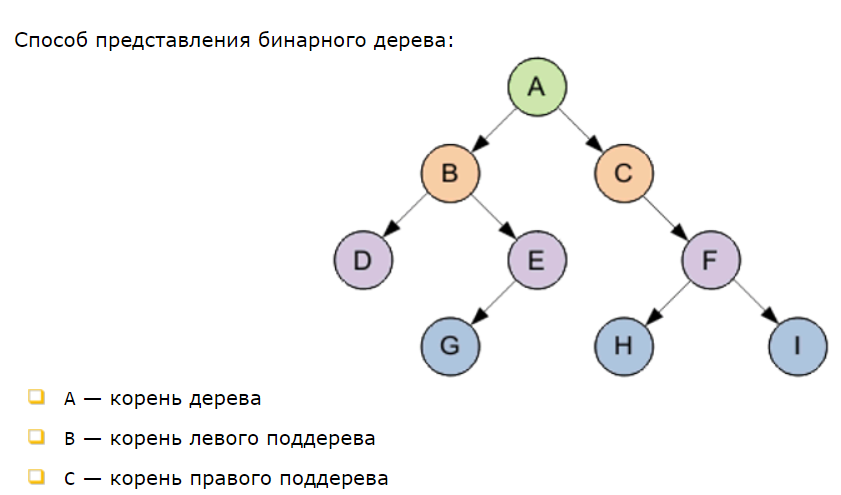
Минск 2022

Оглавление

1. Понятие бинарного дерева
2. Реализация бинарного дерева
3. Пример работы с бинарным деревом и рекурсией
4. Вывод
5. **Понятие Бинарного дерева**

Бинарное дерево — это упорядоченное дерево, каждая вершина которого имеет не более двух поддеревьев, причем для каждого узла выполняется правило: в левом поддереве содержатся только ключи, имеющие значения, меньшие, чем значение данного узла, а в правом поддереве содержатся только ключи, имеющие значения, большие, чем значение данного узла.

При всем этом, если элемент, который мы хотим поместить в дерево, больше, чем корневой, то, с помощью рекурсивного вызова функции происходит последовательное перемещение элемента в правую часть. Наш элемент прыгает через один как через скакалку, пока не найдет своё место. А его место - это то место, где справа элемент или больше чем он сам или если прыгать больше не через кого.  
Если записываемый элемент меньше чем корневой, то выполняется то же самое, только в левую сторону.



Узел D, который находится непосредственно под узлом B, называется потомком B. Если D находится на уровне i, то B – на уровне i-1. Узел B называется предком D.  
  
Максимальный уровень какого-либо элемента дерева называется его глубиной или высотой.  
  
Если элемент не имеет потомков, он называется листом или терминальным узлом дерева.  
  
Остальные элементы – внутренние узлы (узлы ветвления).

Число потомков внутреннего узла называется его степенью. Максимальная степень всех узлов есть степень дерева.

Если элемент не имеет потомков, он называется листом или терминальным узлом дерева.  
  
Остальные элементы – внутренние узлы (узлы ветвления).  
  
Число потомков внутреннего узла называется его степенью. Максимальная степень всех узлов есть степень дерева.  
  
Число ветвей, которое нужно пройти от корня к узлу x, называется длиной пути к x. Корень имеет длину пути равную 0; узел на уровне i имеет длину пути равную i.

1. **Реализация бинарного дерева**

Для того, чтобы отобразить наше бинарное дерево на экране сначала проверяем вообще есть ли оно. Если мы не заполним дерево и не создадим корень, то дерева в принципе существовать не может и поэтому отображать будет нечего, но если создали, то используя рекурсивный вызов функции последовательно отображаем все элементы. Кроме того, эта первая проверка будет и сигналом для завершения рекурсивного вызова функции. Ведь, если встретится нулевое звено, то происходит **return**, а мы после создания каждого звена очищали память для следующего роста.

#include <iostream>

#include <conio.h>

using namespace std;

//Наша структура

struct node

{

int info; //Информационное поле

node \*l, \*r;//Левая и Правая часть дерева

};

node \* tree=NULL; //Объявляем переменную, тип которой структура Дерево

/\*ФУНКЦИЯ ЗАПИСИ ЭЛЕМЕНТА В БИНАРНОЕ ДЕРЕВО\*/

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

{

if ((\*t)==NULL) //Если дерева не существует

{

(\*t)=new node; //Выделяем память

(\*t)->info=a; //Кладем в выделенное место аргумент a

(\*t)->l=(\*t)->r=NULL; //Очищаем память для следующего роста

return; //Заложили семечко, выходим

}

//Дерево есть

if (a>(\*t)->info) push(a,&(\*t)->r); //Если аргумент а больше чем текущий элемент, кладем его вправо

else push(a,&(\*t)->l); //Иначе кладем его влево

}

/\*ФУНКЦИЯ ОТОБРАЖЕНИЯ ДЕРЕВА НА ЭКРАНЕ\*/

void print (node \*t,int u)

{

if (t==NULL) return; //Если дерево пустое, то отображать нечего, выходим

else //Иначе

{

print(t->l,++u);//С помощью рекурсивного посещаем левое поддерево

for (int i=0;i<u;++i) cout<<"|";

cout<<t->info<<endl; //И показываем элемент

u--;

}

print(t->r,++u); //С помощью рекурсии посещаем правое поддерево

}

void main ()

{

int n; //Количество элементов

int s; //Число, передаваемое в дерево

setlocale(LC\_ALL, "Rus");

cout<<"введите количество элементов ";

cin>>n; //Вводим количество элементов

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

{

cout<<"ведите число ";

cin>>s; //Считываем элемент за элементом

push(s,&tree); //И каждый кладем в дерево

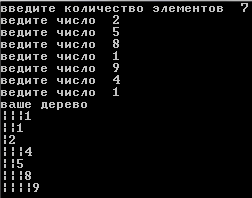
}

cout<<"ваше дерево\n";

print(tree,0);

getch();

}



1. **Пример работы с бинарным деревом и рекурсией**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <locale.h>

#include <windows.h>

#include <stdio.h>

#include <iostream>

#include <string.h>

using namespace std;

typedef struct btree {

int val;

btree \*left,\*right;

};

//----- Рекурсивный поиск в двоичном дереве---------------

// Возвращается указатель на найденную вершину

btree \*Search(btree \*p, int v) {

if (p==NULL) return(NULL); // Ветка пустая

if (p->val == v) return(p); // Вершина найдена

if (p->val > v) // Сравнение с текущим

return(Search(p->left,v)); // Левое поддерево

else

return(Search(p->right,v)); // Правое поддерево

}

//----- Включение значения в двоичное дерево--------------

// функция возвращает указатель на созданную вершину,

// либо на существующее поддерево

btree \*Insert(btree \*pp, btree \*v) {

if (pp == NULL) { // Найдена свободная ветка

// Создать вершину дерева

btree \*q = new btree; // и вернуть указатель

q->val = v->val;

q->left = q->right = NULL;

return q;

}

if (pp->val == v->val) return pp;

if (pp->val > v->val) // Перейти в левое или

pp->left=Insert(pp->left,v); // правое поддерево

else

pp->right=Insert(pp->right,v);

return pp;

}

// Рекурсивный обход двоичного дерева с выводом

// значений вершин в порядке возрастания

void Scan(btree \*p) {

if (p==NULL) return;

Scan(p->left);

cout << p->val << endl;

Scan(p->right);

}

// Рекурсивный обход двоичного дерева с нумерацией вершин

// снизу-вверх слева-направо, n - текущий номер вершины

int Scan2 (btree \* p, int n) {

if (p==NULL) return n;

Scan2 (p->left,n);

n++;

cout << n << ") " << p->val << endl;

n=Scan2(p->right,n) ;

return n;

}

// Рекурсивный обход двоичного дерева с последовательной

// нумерацией вершин в возвратом указателя на вершину с заданным номером

// Глобальный счетчик вершин передается через указатель

btree \*ScanNum(btree \*p, int \*n) {

btree \*q;

if (p==NULL) return NULL;

q=ScanNum(p->left,n);

if (q!=NULL) return q;

if ((\*n)-- ==0) return p;

return ScanNum(p->right,n);

}

int main (void) {

int v=1;

btree \*root=NULL;

int depth=1;

setlocale(LC\_ALL,"Rus"); //SetConsoleCP(1251); SetConsoleOutputCP(1251);

// ввод узлов

while (1) {

printf ("\n Введите целое число (узел дерева) или 0 для выхода:");

fflush (stdin);

scanf ("%d",&v);

if (!v) break;

btree node;

node.val=v;

root=Insert (root,&node);

}

// поиск значения

printf ("\n введите узел для поиска:");

fflush (stdin);

scanf ("%d",&v);

btree \*found=Search (root,v);

if (found) printf ("\n found %d\n",found->val);

else printf ("\n not found %d\n",v);

// нумерация вершин

printf ("\nScan:\n");

Scan (root);

printf ("\n введите начальный номер вершины:");

fflush (stdin);

scanf ("%d",&v);

// поиск по номеру вершины

printf ("\nScan2:\n");

Scan2 (root,v);

printf ("\nScan&Num:\n");

found=ScanNum (root,&v);

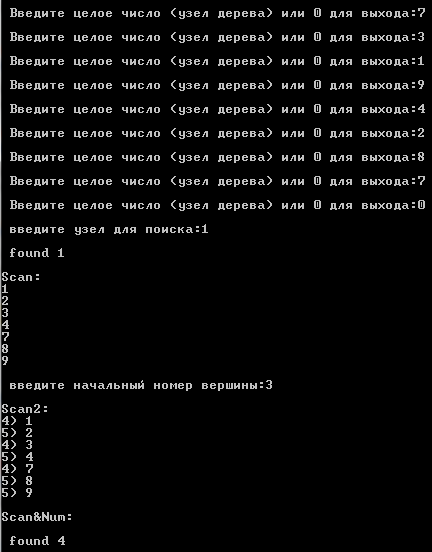
if (found) printf ("\n found %d\n",found->val);

else printf ("\n not found %d\n",v);

fflush (stdin); getchar ();

return 0;

}



1. **Вывод**

Бинарное дерево применяется в тех случаях, когда в каждой точке вычислительного процесса должно быть принято одно из двух возможных решений.  
Имеется много задач, которые можно выполнять на дереве.

Распространенная задача — выполнение заданной операции p с каждым элементом дерева. Здесь p рассматривается как параметр более общей задачи посещения всех узлов или задачи обхода дерева.  
  
Если рассматривать задачу как единый последовательный процесс, то отдельные узлы посещаются в определенном порядке и могут считаться расположенными линейно.