# Лабораторна робота № 13. Алгоритми обробки дерев

***Мета:*** набути досвід практичної роботи з бінарними деревами.

**1 Вимоги**

**1.1 Розробник**

* Макаренко Владислав Олександрович
* Студент 1-го курсу
* Групи КІТ-120а

**1.2 Загальне завдання**

Розробити програму, що дозволяє створити бінарне дерево та вирішити індивідуальне завдання.

**1.2 Загальне завдання**

Визначити кількість правих вузлів

**2 Описи програм**

**Код програми**

#include <stdlib.h>  
#include <iostream>  
using namespace std;  
  
class TreeNode{  
 int key; //ключ  
 int counter = 0;  
 int info; //информационное поле  
 TreeNode \*LLink, \*RLink; //указатель на левый и правый узел  
public:  
 TreeNode(){ key=0;LLink= RLink=NULL;} //конструктор по умолчанию  
 TreeNode(int *k*, int *info\_c*) //конструктор с параметр  
 { key=*k*;  
 info=*info\_c*;  
 LLink=RLink=NULL;  
 }  
 ~TreeNode(){} //Деструктор  
  
 void Insert(int *number*, int *info*, TreeNode \**ptr*); //функция добавления элемента в дерево  
 void PrintTree(TreeNode \**ptr*, int *n*); //функция вывода дерева на экран  
 void DeleteTree(TreeNode \**ptr*); //функция удаления дерева  
 int getCounter(); //функция получения количества правых узлов  
};  
  
int TreeNode::getCounter() {  
 return this->counter;  
}  
  
void TreeNode::Insert(int *n*, int *inf*, TreeNode \**ptr*)  
{ cout<<"\n Добавляется элемент ("<<n<<";"<<inf<<")\n";  
 int flag=1;  
 TreeNode \*p, \*q;  
 p=*ptr*;  
 while(flag){  
 if(*n* < p->key){  
 q=p->LLink; //a3  
 cout<<"Найден узел "<<p->key<<endl<<"Переход влево";  
 if(q==NULL) cout<<"NULL"<<endl;  
 else cout<<q->key<<endl;  
 if(q == NULL){  
 flag=0;  
 cout<<"Создается новый элемент ("<<n<<","<<inf<<")\n";  
 q=new TreeNode(*n*, *inf*);  
 cout<<"Обновление связей: "<<p->key<<"->LLink = "<<q->key<<endl;  
 p->LLink=q;  
 } else {  
 p=q;  
 }  
 }  
 else if(*n* > p->key)  
 { q=p->RLink; //a4  
 cout<<"Найден узел "<<p->key<<endl<<"Переход вправо";  
 if(q == NULL)cout<<"NULL"<<endl;  
 else cout<<q->key<<endl;  
 if(q == NULL){  
 flag=0;  
 cout<<"Создается новый элемент ("<<n<<","<<inf<<")\n ";  
 q=new TreeNode(*n*, *inf*);  
 cout<<"Обновление связей: "<<p->key<<"->RLink = "<<q->key<<endl;  
 p->RLink=q;  
 this->counter++;  
 }  
 else {  
 p=q;  
 }  
 }  
 else if(*n* == p->key){  
 cout<<"Такой элемент уже существует "<<endl;  
 flag=0; //выход из цикла WHILE  
 }  
 }  
}  
  
void TreeNode:: PrintTree(TreeNode \**ptr*, int *n*)  
{ if (*ptr*)  
 { PrintTree(*ptr*->RLink, *n*+3);  
 for(int i=1;i<n;i++)  
 cout << " ";  
 cout<< ptr->key<<endl;  
 PrintTree(*ptr*->LLink, *n*+3);  
 }  
}  
  
void TreeNode::DeleteTree(TreeNode \**ptr*){  
 if (ptr != NULL){  
 DeleteTree(*ptr*->LLink);  
 DeleteTree(*ptr*->RLink);  
 delete(*ptr*);  
 }  
}  
  
int main() //главная программа  
{  
 setlocale(LC\_ALL, "Rus"); // для работи з кирилицей  
 int k\_elem;  
 cout << " Сколько элементов добавлять? ";  
 cin >> k\_elem;  
 int num,inf;  
  
 num=rand() % 100 + 1; //случайные значения  
 inf=rand() % 100 + 1;  
  
 TreeNode \*root=new TreeNode(num, inf); //указатель на корень  
 clock\_t Start = clock();  
 for(int i=1; i<k\_elem; i++)  
 {  
 num=rand() % 100 + 1; //случайные значения  
 inf=rand() % 100 + 1;  
 root->Insert(num,inf,root);  
 }  
 clock\_t End = clock();  
  
 root->PrintTree(root, 5);  
 cout << endl << "Количество правых узлов: " << root->getCounter() <<endl;  
  
 double Res1 = (double)(End - Start)/ CLOCKS\_PER\_SEC;  
 printf("Время обхода дерева: %.10f сек\n\n", Res1);  
  
 root->DeleteTree(root); //освобождение памяти  
}

**Результати виконання програми**

За алгоритмом коду демонструємо роботу програми для 10 елементів (див. рис. 1).

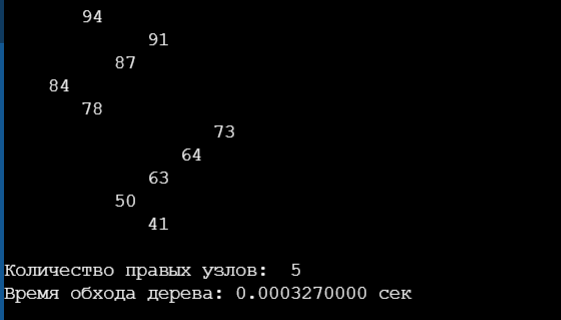


Рисунок 1 – Результати роботи першого кроку

**Висновок:** на цій лабораторній роботі ми набули досвід практичної роботи з бінарними деревами.