# Лабораторна робота № 14. Використання дерев

***Мета:*** набути досвід практичної роботи вирішення задач з використанням бінарних дерев.

**1 Вимоги**

**1.1 Розробник**

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

**1.2 Загальне завдання**

Розробити програму, що створює бінарне дерево та вирішує індивідуальне завдання. Видати вміст дерева та результати індивідуального завдання на екран.

**1.2 Загальне завдання**

Розробити програму для побудови АВЛ-дерева. Визначити значення мінімального та максимального ключів у дереві.

**2 Описи програм**

**Код програми**

#include "cstdlib"  
#include <iostream>  
  
using namespace std;  
  
struct node // структура для представления узлов дерева  
{  
 int key;  
 unsigned char height;  
 node\* left;  
 node\* right;  
};  
  
unsigned char Height(node\* *p*)  
{  
 return *p*? *p*->height : 0;  
}  
  
int Bfactor(node\* *p*)  
{  
 return Height(*p*->right) - Height(*p*->left);  
}  
  
void FixHeight(node\* *p*)  
{  
 unsigned char hl = Height(*p*->left);  
 unsigned char hr = Height(*p*->right);  
 *p*->height = (hl > hr? hl : hr)+1;  
}  
  
node\* RotateRight(node\* *p*) // правый поворот вокруг p  
{  
 node\* q = *p*->left;  
 *p*->left = q->right;  
 q->right = *p*;  
 FixHeight(*p*);  
 FixHeight(q);  
 return q;  
}  
node\* RotateLeft(node\* *q*) // левый поворот вокруг q  
{  
 node\* p = *q*->right;  
 *q*->right = p->left;  
 p->left = *q*;  
 FixHeight(*q*);  
 FixHeight(p);  
 return p;  
}  
node\* Balance(node\* *p*) // балансировка узла p  
{  
 FixHeight(*p*);  
 if( Bfactor(*p*)==2 )  
 { if( Bfactor(*p*->right) < 0 )  
 *p*->right = RotateRight(*p*->right);  
 return RotateLeft(*p*);  
 }  
 if( Bfactor(*p*)==-2 )  
 { if( Bfactor(*p*->left) > 0 )  
 *p*->left = RotateLeft(*p*->left);  
 return RotateRight(*p*);  
 }  
 return *p*; // балансировка не нужна  
}  
  
node\* Insert(node\* *p*, int *k*) // вставка ключа k в дерево с корнем p  
{  
 if( !*p* ){  
 *p* = new node;  
 *p*->key = *k*; *p*->left = *p*->right = 0; *p*->height = 1;  
 return *p*;  
 }  
 if( *k*<*p*->key )  
 *p*->left = Insert(*p*->left,*k*);  
 else  
 *p*->right = Insert(*p*->right,*k*);  
 return Balance(*p*);  
}  
  
node\* FindMin(node\* *p*){ // поиск узла с минимальным ключом в дереве p  
 return *p*->left? FindMin(*p*->left):*p*;  
}  
  
node\* FindMax(node\* *p*){ // поиск узла с максимальным ключом в дереве p  
 return *p*->right? FindMax(*p*->right):*p*;  
}  
  
  
void Print(node \**tree*, int *level*)  
{ int i;  
 if (*tree* == NULL) return;  
 Print(*tree*->left, *level* - 2);  
 for (i = 0; i < *level*; i++)  
 printf(" ");  
 printf("%d\n", *tree*->key);  
 Print(*tree*->right, *level* - 2);  
}  
  
void Delete(node \**tree*) //удаление дерева  
{ if (*tree* == NULL) return;  
 Delete(*tree*->left);  
 Delete(*tree*->right);  
 free(*tree*);  
}  
  
  
int main(){  
 node \*tree = NULL;  
  
 setlocale(LC\_ALL, "Rus"); // для работи з кирилицей  
 int count;  
 cout << " Сколько элементов добавлять? ";  
 cin >> count;  
 int num;  
  
 for(int i = 0; i < count; i++)  
 {  
 num = rand() % 44 + 1; //случайные значения  
 tree = Insert(tree, num);  
 }  
  
 Print(tree, 30);  
  
 clock\_t Start1 = clock();  
 node \*MinNode = FindMin(tree);  
 clock\_t End1 = clock();  
 cout << endl << " Минимальный ключ: " << MinNode->key << endl;  
  
 clock\_t Start2 = clock();  
 node \*MaxNode = FindMax(tree);  
 clock\_t End2 = clock();  
 cout << endl << " Максимальный ключ: " << MaxNode->key << endl;  
  
 double Res1 = (double)(End1 - Start1)/ CLOCKS\_PER\_SEC;  
 printf("Время поиска минимального узла: %.10f сек\n\n", Res1);  
  
 double Res2 = (double)(End2 - Start2)/ CLOCKS\_PER\_SEC;  
 printf("Время поиска максимального узла: %.10f сек\n\n", Res2);  
  
 Delete(tree);  
 return 0;  
}

**Результати виконання програми**

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

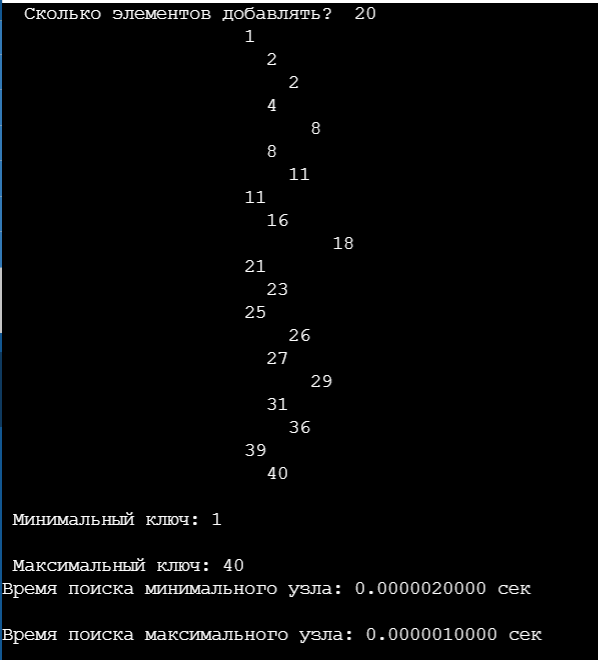


Рисунок 1 – Результати роботи першого кроку

**Висновок:** на цій лабораторній роботі ми набули досвід практичної роботи вирішення задач з використанням бінарних дерев.