МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное учреждение

высшего профессионального образования

«Ижевский государственный технический университет имени

М.Т. Калашникова»

Факультет «Информационно Вычислительная Техника»

Кафедра «Программное обеспечение»

**Лабораторная работа №4**

**АиСД**

**Сбалансированные деревья**

Выполнил:

студент гр.\_Б04-191-1з\_\_\_\_\_\_\_\_\_ \_\_\_Байков Д. О.\_\_\_\_\_\_\_

Проверил: ст. преподаватель

кафедры «Программное обеспечение» \_\_\_Еланцев М. О.\_\_\_\_\_\_

Ижевск 2020

**1.**

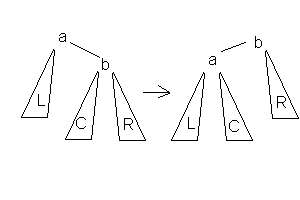
1. **Описание задания**

Осуществить программную реализацию АВЛ- или B-дерева заданного порядка, с указанным типом ключа.

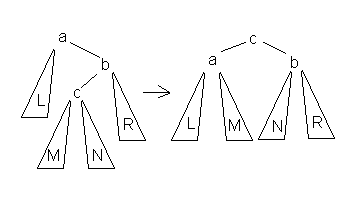
**Краткая теория**

АВЛ-дерево — сбалансированное по высоте двоичное дерево поиска: для каждой его вершины высота её двух поддеревьев различается не более чем на 1.

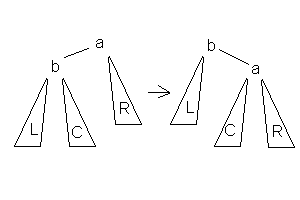
1.**Малое правое вращение**



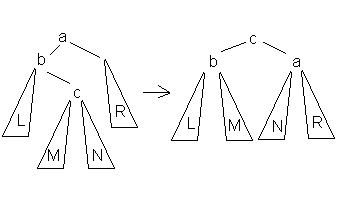
2.**Большое правое вращение**



3.**Малое левое вращение**



4.**Большое левое вращение**



# Код программы

#include <conio.h>

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <locale.h>

#include <windows.h>

struct tree

{

int key;

int h;

tree \*left;

tree \*right;

};

tree\* add(tree \*root, int x)

{

if(!root)

{

root=(tree \*)malloc(sizeof(tree));

root->key=x;

root->left=0;

root->right=0;

root->h=1;

}

else if(x < root->key)

{

root->left=add(root->left, x);

if(root->h<=root->left->h)

root->h=root->left->h+1;

}

else if(x > root->key)

{

root->right=add(root->right, x);

if(root->h<=root->right->h)

root->h=root->right->h+1;

}

return root;

}

void obhod(tree \*root)

{

printf("%d",root->key);

printf("[");

if(root->left)

obhod(root->left);

printf(",");

if(root->right)

obhod(root->right);

printf("]");

}

tree \*hight(tree \*root)

{

if(!root->left && !root->right)

{

root->h=1;

return root;

}

if(root->left)

{

root->left=hight(root->left);

root->h=root->left->h+1;

}

if(root->right)

{

root->right=hight(root->right);

if(root->h <= root->right->h)

root->h=root->right->h+1;

}

return root;

}

tree \*small\_right\_rotate(tree \*root)

{

tree \*a=root;

tree \*b=root->right;

a->right=b->left;

b->left=a;

return b;

}

tree \*small\_left\_rotate(tree \*root)

{

tree \*b=root->left;

tree \*a=root;

a->left=b->right;

b->right=a;

return b;

}

tree \*big\_right\_rotate(tree \*root)

{

tree \*a=root;

tree \*b=root->right;

tree \*c=b->left;

a->right=c->left;

b->left=c->right;

c->left=a;

c->right=b;

return c;

}

tree \*big\_left\_rotate(tree \*root)

{

tree \*a=root;

tree \*b=root->left;

tree \*c=b->right;

a->left=c->right;

b->right=c->left;

c->left=b;

c->right=a;

return c;

}

tree \*balance(tree \*root)

{

if(root->left && root->left->h>2)

root->left=balance(root->left);

if(root->right && root->right->h>2)

root->right=balance(root->right);

int r=0,rr=0,rl=0,rll=0,rlr=0,rrl=0,rrr=0;

r=root->h;

if(root->right)

{

rr=root->right->h;

if(root->right->left)

rrl=root->right->left->h;

if(root->right->right)

rrr=root->right->right->h;

}

if(root->left){

rl=root->left->h;

if(root->left->left)

rll=root->left->left->h;

if(root->left->right)

rlr=root->left->right->h;

}

if((root->right && (rr-rl)==2 && (rrl <= rrr)))

root=small\_right\_rotate(root);

else

if((root->left && (rl-rr)==2 && (rlr <= rll)))

root=small\_left\_rotate(root);

else

if(root->right && root->right->left && (rr-rl)==2 && (rrl > rrr))

root=big\_right\_rotate(root);

else

if(root->left && root->left->right &&(rl-rr)==2 && (rlr > rll))

root=big\_left\_rotate(root);

root=hight(root);

return root;

}

tree \*generation(tree \*root){

int n;

printf("\nVvedite kolichestvo elementov: ");

scanf("%d", &n);

srand(2);

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

root=add(root,rand()%100);

return root;

}

int main(){

setlocale(LC\_ALL,"Rus");

tree \*root=NULL, \*dop;

int key;

char c='1';

while(c!='0'){

system("cls");

printf("\n1.Dobavit'\n2.Vyvesti\n3.Generaciya\n4.Udalit' derevo\n0.Vyhod\n");

c=getch();

switch(c){

case '1':

printf("\nVvedite klyuch: ");

scanf("%d",&key);

root=add(root,key);

if(root && root->h>2)

root=balance(root);

break;

case '2':

if(root)

obhod(root);

getch();

break;

case '3':

root=generation(root);

if(root && root->h>2)

root=balance(root);

break;

case '4':

free(root);

root=NULL;

printf("\nDerevo udaleno!");

getch();

break;

}

}}

1. **Пример работы программы.**

