

## Arbol Balanceado

```
#include<stdio.h>
#include <conio.h>
#include<malloc.h>

struct nodo
{
    int key,con;
    int bal;
    struct nodo *izq,*der;
};
typedef struct nodo *punt;

void crear(punt &p)
{
    p=NULL;
}

void insertar(punt &p,int x,int &h)
{
    punt p1,p2;
    if(p==NULL)/*insertar*/
    {
        p=(nodo*)malloc(sizeof(nodo));
        h=1;
        p->key=x;
        p->con=1;
        p->izq=NULL;
        p->der=NULL;
        p->bal=0;
    }
    else
        if(p->key>x)
        {
            insertar(p->izq,x,h);
            if(h)/*La rama izquierda crecio*/
                switch(p->bal)
                {
                    case 1:{
                        p->bal=0;
                        h=0;
                        break;
                    }
                    case 0:{
                        p->bal=-1;
                        break;
                    }
                    case-1:{/*Rebalancear*/
                        p1=p->izq;
                        if(p1->bal==1)/*Rotacion simple lado
izquierdo*/
```

```

        {
            p->izq=p1->der;
            p1->der=p;
            p->bal=0;
            p=p1;
        }
    else/*Rotacion doble lado derecho*/
    {
        p2=p1->der;
        p1->der=p2->izq;
        p2->izq=p1;
        p->izq=p2->der;
        p2->der=p;
        if (p2->bal== -1)
            p->bal=1;
        else
            p->bal=0;
        if (p2->bal== 1)
            p1->bal= -1;
        else
            p1->bal=0;
        p=p2;
    }
    p->bal=0;
    h=0;
}/*fin Case*/
}/*fin switch*/
}/*fin if*/

else
if (p->key<x)
{
    insertar(p->der,x,h);
    if(h)/*La rama derecha crecio*/
    switch(p->bal)
    {
        case -1:{
            p->bal=0;
            h=0;
            break;
        }
        case 0:{
            p->bal=1;
            break;
        }
        case 1:{/*Rebalancear*/
            p1=p->der;
            if (p1->bal==1)/*Rotacion simple lado
derecho*/
            {
                p->der=p1->izq;
                p1->izq=p;
            }
        }
    }
}

```

```

        p->bal=0;
        p=p1;
    }
    else/*Rotacion doble lado izq*/
    {
        p2=p1->izq;
        p1->izq=p2->der;
        p2->der=p1;
        p->der=p2->izq;
        p2->izq=p;
        if (p2->bal==1)
            p->bal=-1;
        else
            p->bal=0;
        if (p2->bal==-1)
            p1->bal=1;
        else
            p1->bal=0;
        p=p2;
    }
    p->bal=0;
    h=0;
    }/*fin case*/
}/*fin switch*/
}/*fin if*/
else p->con++;
}

```

```

void balani(punt &p,int &h)
{punt p1,p2;
int b1,b2;
switch(p->bal)
{case -1:{p->bal=0;
        break;}
  case 0:{p->bal=1;
        h=1;
        break;}
  case 1:{/*rebalanceo*/
        p1=p->der;
        b1=p1->bal;
        if (b1>=0)/*rotacion simple*/
        {
            p->der=p1->izq;
            p1->izq=p;
            if (b1==0)
            {
                p->bal=1;
                p1->bal=-1;
                h=0;
            }
            else

```

```

        {
        p->bal=0;
        p1->bal=0;
        }
        p=p1;
    }/*fin rotacion simple*/
    else{/*rotacion doble*/
        p2=p1->izq;
        b2=p2->bal;
        p1->izq=p2->der;
        p2->der=p1;
        p->der=p2->izq;
        p2->izq=p;
        if(b2==1)
            p->bal=-1;
        else
            p->bal=0;
        if(b2==-1)
            p1->bal=1;
        else
            p1->bal=0;
        p=p2;
        p2->bal=0;
    }/*fin rotacion doble*/
    }/*fin case*/
}/*fin switch*/
}/*fin balani*/

```

```

void baland(punt &p,int &h)
{
    punt p1,p2;
    int b1,b2;
    switch(p->bal)
    {
        case 1:{
            p->bal=0;
            break;
        }
        case 0:{
            p->bal=-1;
            h=0;
            break;
        }
        case -1:{/*rebalanceo*/
            p1=p->izq;
            b1=p1->bal;
            if(b1<=0)/*rotacion simple*/
            {
                p->izq=p1->der;
                p1->der=p;
                if(b1==0)
                {

```

```

        p->bal=-1;
        p1->bal=1;
        h=0;
    }
    else
    {
        p->bal=0;
        p1->bal=0;
    }
    p=p1;
}/*fin rotacion simple*/
else{/*rotacion doble*/
    p2=p1->der;
    b2=p2->bal;
    p1->der=p2->izq;
    p2->izq=p1;
    p->izq=p2->der;
    p2->der=p;
    if (b2==-1)
    //if (b2==1)
        p->bal=1;
    else
        p->bal=0;
    if (b2==1)
        if (b2==-1)                //****
            p1->bal=-1;
        else
            p1->bal=0;
    p=p2;
    p2->bal=0;
}/*fin rotacion doble*/
}/*fin case*/
}/*fin switch*/
}/*fin baland*/

void sup(punt &r,int &h, punt &c)
{
    if((r->der)!=NULL)
    {
        sup(r->der,h,c);
        if (h)
            baland(r,h);
    }
    else{
        c=(nodo*)malloc(sizeof(nodo));
        c->key=r->key;
        c->con=r->con;
        c=r;
        r=r->izq;
        h=1;
    }
}/*fin sup*/

```

```

void suprimir(punt &p,int x,int h)
{
    punt q;
    if(p==NULL)
        printf("\nLa llave no esta en el arbol");
    else
        if(p->key>x)
        {
            suprimir(p->izq,x,h);
            if(h)
                balani(p,h);
        }
        else
            if(p->key<x)
            {
                suprimir(p->der,x,h);
                if(h)
                    baland(p,h);
            }
            else /*eliminar p->*/
            {
                q=p;
                if(q->der==NULL)
                {
                    p=q->izq;
                    h=1;
                }
                else
                    if(q->izq==NULL)
                    {
                        p=q->der;
                        h=1;
                    }
                else
                {
                    punt c;
                    sup(q->izq,h,c);
                    //agrego
                    p->key=c->key;
                    p->con=c->con;
                    //fin
                    if(h)
                        balani(p,h);
                }

                printf("%d",p->key);
                ;
                getchar();
            }
    }
}

```

```

void mostrar(punt p)
{
    static int i=0;
    if(p!=NULL)
    {
        printf("\n %d    ",p->key);
        printf("\n izquierda  ");
        mostrar(p->izq);
        printf("\n derecha    ");
        mostrar(p->der);
    }
}

void main ()
{
    punt p;
    int op,ele,letra;
    crear(p);
    do
    {
        clrscr();

        printf("\n\n Menú \n");
        printf("\n 1_ Insertar\n");
        printf("\n 2_ Suprimir\n");
        printf("\n 3_ Mostrar\n");
        printf("\n 4_ Salir\n");
        printf("\n \n");

        printf("Ingrese opción  ");
        scanf("%d",&op);
        getchar();
        switch (op)
        {
            case 1:
            {
                printf("Ingrese elemento a insertar, el ingreso finaliza
con 0  ");
                scanf("%d",&ele);
                getchar();
                while (ele!=0)
                {
                    insertar(p,ele,letra);
                    printf("Ingrese elemento a insertar, el ingreso finaliza
con 0  ");
                    scanf("%d",&ele);
                    getchar();
                }
                break;
            }
            case 2:

```

```

    {
        printf("Ingrese elemento a suprimir, el ingreso finaliza
con 0 ");
        scanf("%d",&ele);
        getchar();
        while (ele!=0)
        {
            letra=0;
            suprimir(p,ele,letra);
            printf("Ingrese elemento a suprimir, el ingreso finaliza
con 0 ");
            scanf("%d",&ele);
            getchar();
        }
        break;
    }
    case 3:
    {
        mostrar(p);
        getchar();
        break;
    };
}
}
while (op!=4);
}

```