LABORATORIO 10

INGENIERIA EN COMPUTACION

MARIANA ESTEFANIA BARCENAS RODRIGUEZ
UAZ 3°A

Actividad 19:

```
//estructura de datos
//mariana estefania barcenas rodriguez
/* OBJETIVOS: Crear arbol balanceado a partir de medioa del arreglo */
// 12/05/2022

#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
typedef struct nodoA{
    int info;
    int  FE;
    struct nodoA *izq;
    struct nodoA *der;
}tiponodo;
typedef tiponodo *NodoA;
NodoA arbol;
//bool balance=false;
NodoA nodo1=NULL;
NodoA nodo2=NULL;
NodoA aux=NULL;
NodoA aux1=NULL;
NodoA otro=NULL;

NodoA nuevoNodo(NodoA,int,int,NodoA );
NodoA Inserta_Balanceado(NodoA,bool, int);
void Restructura_Izq(NodoA, bool);
void Restructura_Der(NodoA, bool );
NodoA Elimina_Balanceado(NodoA,bool,int );
//RECORRIDOS DE LOS ARBOLES
void inorden(NodoA );
void preorden (NodoA );
void postorden(NodoA );
int compara(int,int);


int main()
{
    int elementoelimina;
    int x;
    int elemento[] = {1,2,3,4};
    int tamanioElemento = sizeof elemento[0];
    int longitud = sizeof elemento / tamanioElemento;
```

```
    // Obtener media
    int sumatoria = 0;
    for (x = 0; x < longitud; x++) {
        sumatoria += elemento[x];

        printf(" %d ", elemento[x]);
    }
    float media = (float) sumatoria / (float) longitud;


    printf("\n Media: %f", media+(0.5));

    do{
    printf("\nDame el elemento a eliminar\n");
    scanf("%d",&elementoelimina);
    arbol= Elimina_Balanceado(arbol,false,elementoelimina );

    }while(elementoelimina!=0);


    return 0;
    }
    NodoA nuevoNodo(NodoA izq,int inf,int FE,NodoA der){
    NodoA q;
    q=(NodoA)malloc(sizeof(tiponodo));
    if (!q){
    printf("\n Error al crear el nuevo nodo");
    exit(0);
    }
    q->info=inf;
    q->izq=izq;
    q->der=der;
    q->FE=FE;
    return q;
    }

    NodoA Inserta_Balanceado(NodoA nodo,bool balance, int info){ //al
iniciar es falso
    if (nodo!=NULL){
    //balance=false;
    if(info<nodo->info){
    nodo->izq=Inserta_Balanceado(nodo->izq,balance,info);
    if(balance==true){
if(nodo->FE==1){
```

```
    nodo->FE=0;
    balance=false;
    }
    else if(nodo->FE==0)
    nodo->FE=-1;
    else if(nodo->FE==-1){
    nodo1=nodo->izq;
    if(nodo1->FE<=0) {//rotación II
    nodo->izq=nodo1->der;
    nodo1->der=nodo;
    nodo->FE=0;
    nodo=nodo1;
    } //fin rotación II
    else{ //Rotación ID
    nodo2=nodo1->der;
    nodo->izq=nodo2->der;
    nodo2->der=nodo;
    nodo1->der=nodo2->izq;
    nodo2->izq=nodo1;
    if (nodo2->FE=-1)
    nodo->FE=1;
    else
    nodo->FE=0;
    if (nodo2->FE==1)

    nodo1->FE=-1;
    else
    nodo1->FE=0;
    nodo=nodo2;
    }//fin rotación ID
    nodo->FE=0;
    balance=false;
    }
    }
    else
    return nodo;
    }//FIN DEL ES Menor
    else if(info>nodo->info){
    nodo->der=Inserta_Balanceado(nodo->der,balance, info);
    if(balance==true){
    if (nodo->FE==-1){
    nodo->FE=0;
    balance=false;
    }
    else if(nodo->FE==0)
```

```
    nodo->FE=1;
    else if(nodo->FE==1){
    nodo1=nodo->der;

    if(nodo1->FE>=0){//rotación DD
    nodo->der=nodo1->izq;
    nodo1->izq=nodo;
    nodo->FE=0;
    nodo=nodo1;
    }//termina rotación DD
    else{//ROTACIÓN DI
    nodo2=nodo1->izq;
    nodo->der=nodo2->izq;
    nodo2->izq=nodo;
    nodo1->izq=nodo2->der;
    nodo2->der=nodo1;
    if(nodo2->FE==1)
    nodo->FE=-1;
    else
    nodo->FE=0;

    if(nodo2->FE=-1)
    nodo1->FE=1;
    else
    nodo1->FE=0;
    nodo=nodo2;
    }//fin rotación DI
    nodo->FE=0;
    balance=false;
    }
    }
    else
    return nodo;

    }//fin es mayor
    /*else{
    printf("\nLa información ya se encuentra en el arbol\n");
    return nodo;
    }*/


    }//fin del if is null
    else{
    balance=true;
    nodo=nuevoNodo(NULL,info,0,NULL);
```

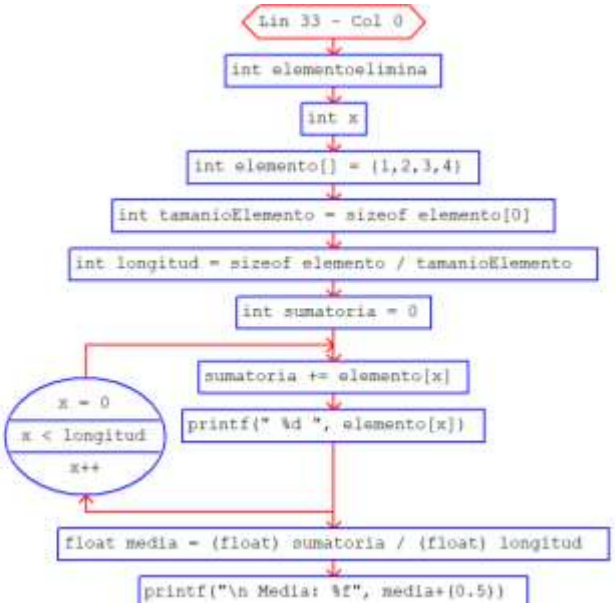
```
    return nodo;

    }


    }
    //eliminación
    void Restructura_Izq(NodoA nodo, bool balance){
    if (balance==true){
    if(nodo->FE==-1)
    nodo->FE=0;
    else if(nodo->FE==0){
    nodo->FE=1;
    balance=false;
    }
    else if(nodo->FE==1){ //reestrucutracion del árbol

                nodo1=nodo->der;

                if(nodo1->FE>=0){ //Rotación DD
                nodo->der=nodo1->izq;
                nodo1->izq=nodo;
                if(nodo1->FE==0){
                nodo->FE=1;
                nodo1->FE=-1;
                balance=false;
                }
                else if(nodo1->FE==1){
                nodo->FE=0;
                nodo1->FE=0;
                }
                nodo=nodo1;
                }//termina rotación DD
                else{//rotación DI
                nodo2=nodo1->izq;
                nodo->der=nodo2->izq;
                nodo2->izq=nodo;
                nodo1->izq=nodo2->der;
                nodo2->der=nodo1;

                if (nodo2->FE==1)
                nodo->FE=-1;
                else
                nodo->FE=0;
                if(nodo2->FE=-1)
```

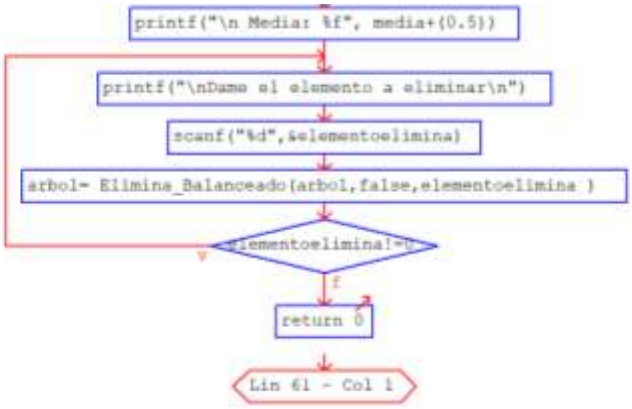
```
                nodo1->FE=1;
                else
                nodo1->FE=0;
                nodo=nodo2;
                nodo2->FE=0;
                //termina rotación DI
            }
            }
        }

    }

    void    Restructura_Der(NodoA nodo, bool balance){
        if(balance==true){
        if(nodo->FE==1)
        nodo->FE=0;
        else if(nodo->FE==0){
    nodo->FE=-1;
        balance=false;
        }
        else if(nodo->FE==-1){ //restructuracionm del arbol
        nodo1=nodo->izq;
        if(nodo1->FE<=0){//rotacion II
        nodo->izq=nodo1->der;
        nodo1->der=nodo;
        if (nodo1->FE==0){
        nodo->FE=-1;
        nodo1->FE=1;
    balance=false;
        }else if(nodo1->FE=-1){
        nodo->FE=0;
        nodo1->FE=0;
        }
        nodo=nodo1;
        }//fin rotacion II
        else{//rotación ID
        nodo2=nodo1->der;
        nodo->izq=nodo2->der;
    nodo2->der=nodo;
        nodo1->der=nodo2->izq;
        nodo2->izq=nodo1;
        if(nodo2->FE==-1)
        nodo->FE=1;
        else
        nodo->FE=0;
```

```
        if(nodo2->FE==1)
    nodo1->FE=-1;
    else
        nodo1->FE=0;
        nodo=nodo2;
        nodo2->FE=0;
        }//FIN ROTACIÓN id
        }
        }//fin balance


    }
    NodoA Elimina_Balanceado(NodoA nodo,bool balance, int info){
    if(nodo!=NULL){
    if(info<nodo->info){
    nodo->izq=Elimina_Balanceado(nodo->izq,balance,info);
    Restructura_Izq(nodo,balance);
    }//es es menor
    else if(info>nodo->info){
    nodo->der=Elimina_Balanceado(nodo->der,balance,info);
    Restructura_Der(nodo,balance);
    }//fin es mayor
    else{
    otro=nodo;
    balance=true;
    if(otro->der==NULL){
    nodo=otro->izq;
    }//fin si otros es null
    else{
    if(otro->izq==NULL)
    nodo=otro->der;
    else{
    aux=nodo->izq;
    balance=false;
    while(aux->der!=NULL){
    aux1=aux;
    aux=aux->der;
    balance=true;
    }
    nodo->info=aux->info;
    otro=aux;
    if(balance==true)
    aux1->der=aux->izq;
    else
    nodo->izq=aux->izq;
```

```
    Restructura_Der(nodo->izq,balance);
    }//fin else otro->izq==NULL
    }
    free(otro);
    }
    //printf("No se encuentra");
    }//fin del null
    else
    printf("arbol vacío");
    return nodo;
    }//fin function

    void inorden(NodoA raiz){
        if (raiz!=NULL){
    inorden(raiz->izq);
    printf(" %d ",raiz->info);
    inorden(raiz->der);
    }
    }
```

DIAGRAMA FLUJO:




printf("\n Media: %f", media+(0.5))
printf("\nDame el elemento a eliminar\n")
scanf("%d",&elementoelimina)
arbol= Elimina_Balanceado(arbol,false,elementoelimina)
elementoelimina!=0
V
F
return 0
Lin 61 - Col 1