LABORATORIO 9

INGENIERIA EN COMPUTACION

Mariana Estefanía Bárcenas Rodriguez

uaz  3°A

**ACTIVIDAD 17**

**CODIGO:**

//estructura de datos

//mariana estefania barcenas rodriguez

// 10/05/2022

#include<conio.h>

#include<stdlib.h>

#include <string.h>

#include<stdio.h>

#include<malloc.h>

#define tam 50

typedef struct nodoA{

 char info;

 struct nodoA \*izq;

 struct nodoA \*der;

}tiponodo;

typedef tiponodo \*NodoA;

NodoA arbol;

int tope=-1,i=0,pos=0;

int tamaExp;

char pila[tam];

char exInf[tam];

int epre[tam];

char epos[tam];

char sim,simbolo;

int prioridad(char operador, char operadorPila);

int prior(char simbolo);

NodoA nuevoNodo(NodoA izq, char inf, NodoA der){

 NodoA q;

 q=(NodoA)malloc(sizeof(tiponodo));

 if (!q){

 printf("\n Error al crear el nuevo nodo");

 return 0;

 }

 q->info=inf;

 q->izq=izq;

 q->der=der;

 return q;

}

NodoA cargarNodos(NodoA ini){

 char dato;

 char opc;

 printf("\n Dame el elemento del nodo a insertar: \n");

 scanf ("\n %c",&dato);

 ini=nuevoNodo(NULL,dato,NULL);

 fflush(stdin);

 printf("\n El nodo %c tiene sub-arbol Izquierdo ? ",dato);

 scanf("%c",&opc);

 fflush(stdin);

 if (opc=='s' || opc=='S')

 ini->izq=cargarNodos(ini->izq);

 fflush(stdin);

 printf("\n El nodo %c tiene sub-arbol derecho ? \n",dato);

 scanf("%c",&opc);

 fflush(stdin);

 if (opc=='s' || opc=='S')

 ini->der =cargarNodos(ini->der);

 return ini;

}

void inorden(NodoA raiz){

 if(raiz !=NULL){

 inorden(raiz->izq);

 printf(" %c",raiz->info);

 inorden(raiz->der);

 }

}

void preorden(NodoA raiz){

 if (raiz!=NULL){

 printf(" %c",raiz->info);

 preorden(raiz->izq);

 preorden(raiz->der);

 }

}

void postorden(NodoA raiz){

 if (raiz!=NULL){

 postorden(raiz->izq);

 postorden(raiz->der);

 printf(" %c",raiz->info);

 }

}

int main(){

    int opc;

    do{

    printf("\n 1..... in,pos,pre orden \n");

    printf("\n 2..... infija a posfija \n");

    printf("\n 3..... infija a prefija \n");

    printf("\n Ingrese opcion:");

        scanf("%d",&opc);

   switch (opc)

   {

   case 1:

       arbol=cargarNodos(arbol);

       printf("\n arbol en inorden \n");

       inorden(arbol);

       printf("\n arbol en postorden \n");

       postorden(arbol);

       printf("\n arbol en preorden \n");

       preorden(arbol);

       break;

   case 2:

       printf("\n ingrese la notacion Infija:");

    gets(exInf);

    while(exInf[i] != '\0')

    {

        sim=exInf[i];

        if(sim=='(')

        {

            tope++;

            pila[tope]=sim;

        }

        else

        {

            if(sim==')')

            {

                while(pila[tope]!='(')

                {

                    epos[pos]=pila[tope];

                    pos++;

                    tope--;

                }

                //pila[tope]='';

                tope--;

            }

            else{

                if(sim!='+' && sim!='-' && sim!='\*' && sim!='/' && sim!='+' && sim!='%' && sim!='^'  )

                {

                    epos[pos]=sim;

                    pos++;

                }

                else

                { //es el operador

                    while(tope>-1 && (prioridad(sim,pila[tope]) <=0))

                    {

                        epos[pos]=pila[tope];

                        tope--;

                        pos++;

                    }

                    tope++;

                    pila[tope]=sim;

                }

            }

        }

        i++;

    }//fin while

    while(tope>-1)

    {

        epos[pos]=pila[tope];

        tope--;

        pos++;

    }

    printf("\n la expresion Posfija es:");

        puts(epos);

       break;

   case 3:

       //printf("\n ingrese nuevamente la Infija: ");

        //scanf("%s",&exInf);

        tope=-1;

        pos=-1;

        tamaExp=strlen(exInf);

        for(int i=tamaExp-1;i>=0;i--)

        {

            simbolo=exInf[i];

            if(simbolo==')')

            {

                tope++;

                pila[tope]=simbolo;

            }

            else if(simbolo=='(')

            {

                while(pila[tope]!=')')

                {

                    pos++;

                    epre[pos]=pila[tope];

                    tope--;

                }

                tope--;

            }

            else if( (simbolo>='a')&&(simbolo<='z') || (simbolo>='A')&&(simbolo<='Z') )

            {

                pos++;

                epre[pos]=simbolo;

            }

            else

            {

                while((prior(simbolo) < prior(pila[tope])))

                {

                    pos++;

                    epre[pos]=pila[tope];

                    tope--;//?

                }

                tope++;

                pila[tope]=simbolo;

            }

        }

        while(tope>-1)

        {

            pos++;

            epre[pos]=pila[tope];

            tope--;//?

        }

        printf("\n La expresion Prefija es:");

        for(int i=pos;i>=0;i--)

        {

            printf("%c",epre[i]);

        }

        getch();

       break;

   default:

       break;

   }

   } while(opc!=4);

 //return 0;

}

int prioridad(char operador, char operadorPila)

{

    int ps=0,pp=0;

    if(operador=='(' || operador==')')

        ps=0;

    if(operador=='+' || operador=='-')

        ps=1;

    if(operador=='/' || operador=='\*' || operador=='%')

        ps=2;

    if(operador=='^')

        ps=3;

    if(operadorPila=='(' || operadorPila==')')

        pp=0;

    if(operadorPila=='+' || operadorPila=='-')

        pp=1;

    if(operadorPila=='/' || operadorPila=='\*' || operadorPila=='%')

        pp=2;

    if(operadorPila=='^')

        pp=3;

    return ps-pp;

}

int prior(char simbolo)

{

    int priori;

    switch(simbolo)

    {

    case '^':

        priori=3;

        break;

    case '/':

        priori=3;

        break;

    case '\*':

        priori=2;

        break;

    case '-':

        priori=1;

        break;

    case '+':

        priori=1;

        break;

    case ')':

        priori=0;

        break;

    case '(':

        priori=0;

        break;

    }

    return priori;

}

**DIAGRAMA DE FLUJO:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama, Aplicación

Descripción generada automáticamente

**ACTIVIDAD 18**

Gráfico

Descripción generada automáticamente con confianza media

**CODIGO:**

//estructura de datos

//mariana estefania barcenas rodriguez

// 10/05/2022

#include<conio.h>

#include<stdlib.h>

#include<stdio.h>

typedef struct nodoA{

    int info;

    struct nodoA \*izq;

    struct nodoA \*der;

}tiponodo;

typedef tiponodo \*NodoA;

NodoA arbol;

NodoA nuevoNodo(NodoA,int,NodoA );

NodoA cargarNodos(int,NodoA);

//RECORRIDOS DE LOS ARBOLES

void inorden(NodoA );

void preorden (NodoA );

void postorden(NodoA );

int compara(int,int);

NodoA Busca(NodoA,int);

void InsertarDato(int);

NodoA elimina(NodoA,int);

int main(){

    int edad=-1,mayor,edades;

    NodoA temp;

    printf("\n\n Cuantas edades quieres introducir en tu arbol ?");

    scanf("%d",&edades);

    for(int i=0;i<edades;i++)

    {

    printf("Dame las edades: \n");

    scanf(" %d",&edad);

    if (edad!=0)

    arbol=cargarNodos(edad,arbol);

    if(edad>mayor)

        {

            mayor=edad;

        }

    }

    printf("\n arbol en inorden\n");

    inorden(arbol);

    printf("\n arbol en preorden\n");

    preorden(arbol);

    printf("\n arbol en postorden\n");

    postorden(arbol);

    printf("\nla edad mayor entre todos es:%d",mayor);

    printf("\n Dame un elemento a buscar");

    scanf("%d",&edad);

    temp=Busca(arbol,edad);

    if (temp!=NULL)

    printf("\n La raiz del arbol es %d ",temp->info);

    else

    printf("\n Elemento no encontrado");

    return 0;

}

NodoA nuevoNodo(NodoA izq,int inf,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;

    return q;

}

NodoA cargarNodos(int dato, NodoA ini){

    if (ini==NULL){

    ini=nuevoNodo(NULL,dato, NULL);     }

    else if (compara(dato, ini->info)==-1){ //va al lado izquierdo

    ini->izq=cargarNodos(dato,ini->izq);

    }

    else if (compara(dato,ini->info)==1){//va al lado derecho

    ini->der=cargarNodos(dato, ini->der);

        }

    return ini;

}//fin funciÃ³n

int compara(int a, int b){

    if (a<b)

    return -1;

    if (a>b)

    return 1;

    if (a==b)

    return 0;

    return NULL;

}

void inorden(NodoA raiz){

    if (raiz!=NULL){

    inorden(raiz->izq);

    printf(" %d", raiz->info);

    inorden(raiz->der);

    }

}

void postorden(NodoA raiz){

    if (raiz!=NULL){

    postorden(raiz->izq);

    postorden(raiz->der);

    printf(" %d", raiz->info);

    }

}

void preorden(NodoA raiz){

    if(raiz!=NULL){

    printf(" %d", raiz->info);

    preorden(raiz->izq);

    preorden(raiz->der);

    }

}

NodoA Busca(NodoA raiz, int datoB){

    int bandera=0;

    if (raiz!=NULL){

        if (datoB==raiz->info){

        bandera=1;

        }

        printf("\n El elemento buscado se encontro y es %d ",raiz->info);

        return raiz;

    }

    else{

        if (compara(datoB, raiz->info)==-1){

        raiz->izq=Busca(raiz->izq,datoB);

        }

        else if (compara(datoB, raiz->info)==1) {

        raiz->der=Busca(raiz->der,datoB);

    }

    }

}

**DIAGRAMA DE FLUJO:**

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Aplicación

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente