Reporte de Examen práctico.

Problema #: Descripción del problema

Insertar nodos a un arbol binario y recorrer en preorden, inorden y postorden.

Nombre del alumno(a):	Cruz Virgen Nora Hilda	Fecha:	04/09/2025

Código en el lenguaje Fortran

```
PROGRAM ARBOL_BINARIO
IMPLICIT NONE
INTEGER, PARAMETER :: NMAX = 100
INTEGER :: VALOR(NMAX), IZQ(NMAX), DER(NMAX)
INTEGER :: RAIZ, DISP, I
COMMON /ARBOL/ VALOR, IZQ, DER, RAIZ, DISP


RAIZ = 0
DISP = 1
DO I = 1, NMAX
   IZQ(I) = 0
   DER(I) = 0
END DO

CALL INSERTAR(RAIZ, 10)
CALL INSERTAR(RAIZ, 8)
CALL INSERTAR(RAIZ, 13)
CALL INSERTAR(RAIZ, 5)
CALL INSERTAR(RAIZ, 9)
CALL INSERTAR(RAIZ, 12)
CALL INSERTAR(RAIZ, 20)

PRINT *, 'PREORDEN:'
CALL PREORDEN(RAIZ)

PRINT *, 'INORDEN:'
CALL INORDEN(RAIZ)

PRINT *, 'POSTORDEN:'
CALL POSTORDEN(RAIZ)
```

Ejecución

```
PREORDEN:
10
8
5
9
13
12
20
INORDEN:
5
8
9
10
12
13
20
POSTORDEN:
5
9
8
12
20
13
10
ENTER PPARA SALIR
```

```
PRINT *, "ENTER PPARA SALIR"
READ(*,*)

END PROGRAM ARBOL_BINARIO

RECURSIVE SUBROUTINE INSERTAR(NODO, X)
IMPLICIT NONE
INTEGER NODO, X
INTEGER, PARAMETER :: NMAX = 100
INTEGER :: VALOR(NMAX), IZQ(NMAX), DER(NMAX
INTEGER :: RAIZ, DISP
COMMON /ARBOL/ VALOR, IZQ, DER, RAIZ, DISP

IF (NODO .EQ. 0) THEN
   NODO = DISP
   VALOR(NODO) = X
   IZQ(NODO) = 0
   DER(NODO) = 0
   DISP = DISP + 1
ELSE IF (X .LT. VALOR(NODO)) THEN
   CALL INSERTAR(IZQ(NODO), X)
ELSE
   CALL INSERTAR(DER(NODO), X)
END IF

END SUBROUTINE INSERTAR

RECURSIVE SUBROUTINE PREORDEN(NODO)
IMPLICIT NONE
INTEGER NODO
INTEGER, PARAMETER :: NMAX = 100
```

```
INTEGER :: VALOR(NMAX), IZQ(NMAX), DER(NMAX)
INTEGER :: RAIZ, DISP
COMMON /ARBOL/ VALOR, IZQ, DER, RAIZ, DISP

IF (NODO .NE. 0) THEN
   PRINT *, VALOR(NODO)
   CALL PREORDEN(IZQ(NODO))
   CALL PREORDEN(DER(NODO))
END IF
END SUBROUTINE PREORDEN

RECURSIVE SUBROUTINE INORDEN(NODO)
IMPLICIT NONE
INTEGER NODO
INTEGER, PARAMETER :: NMAX = 100
INTEGER :: VALOR(NMAX), IZQ(NMAX), DER(NMAX)
INTEGER :: RAIZ, DISP
COMMON /ARBOL/ VALOR, IZQ, DER, RAIZ, DISP

IF (NODO .NE. 0) THEN
   CALL INORDEN(IZQ(NODO))
   PRINT *, VALOR(NODO)
   CALL INORDEN(DER(NODO))
END IF
END SUBROUTINE INORDEN

RECURSIVE SUBROUTINE POSTORDEN(NODO)
IMPLICIT NONE
INTEGER NODO
INTEGER, PARAMETER :: NMAX = 100
INTEGER :: VALOR(NMAX), IZQ(NMAX), DER(NMAX)
INTEGER :: RAIZ, DISP
COMMON /ARBOL/ VALOR, IZQ, DER, RAIZ, DISP

IF (NODO .NE. 0) THEN
   CALL POSTORDEN(IZQ(NODO))
   CALL POSTORDEN(DER(NODO))
   PRINT *, VALOR(NODO)
END IF
END SUBROUTINE POSTORDEN
```

Código en el lenguaje Pascal	Ejecución

```
program ARBOLBIN;
uses crt;
type
PNodo = ^Nodo;
Nodo = record
valor: integer;
izq,der: PNodo;
end;

var
raiz: PNodo;

function CrearNodo(x: integer): PNodo;
var
nuevo: PNodo;

begin
new(nuevo);
nuevo^.valor :=x;
nuevo^.izq := nil;
nuevo^.der :=nil;
CrearNodo := nuevo;
end;

function Insertar(nodo: PNodo; x: integer): PNodo;
begin
if nodo= nil then
Insertar := CrearNodo(x)
else
begin
if x < nodo^.valor then
nodo^.izq := Insertar(nodo^.izq, x)
else
nodo^.der := Insertar(nodo^.der, x);
Insertar := nodo;
end;
end;
```

```
PREORDEN:
10859131220
INORDEN:
58910121320
POSTORDEN:
59812201310
```

```
procedure Preorden(nodo: PNodo);
begin
if nodo <> nil then
begin
write(nodo^.valor, '');
Preorden(nodo^.izq);
Preorden(nodo^.der);
end;
end;

procedure Inorden(nodo: PNodo);
begin
 if nodo <> nil then
 begin
 Inorden(nodo^.izq);
 write(nodo^.valor,'');
 Inorden(nodo^.der);
 end;
 end;
procedure Postorden(nodo: PNodo);
begin
if nodo <> nil then
begin
Postorden(nodo^.izq);
Postorden(nodo^.der);
write(nodo^.valor,'');
end;
end;

begin
clrscr;
raiz := nil;

raiz := Insertar(raiz, 10);
Insertar(raiz,8);
Insertar(raiz,13);
Insertar(raiz,5);
Insertar(raiz,9);
Insertar(raiz,12);
```

```
Insertar(raiz,20);

writeln('PREORDEN:');
Preorden(raiz);

writeln(#13#10, 'INORDEN:');
Inorden(raiz);

writeln(#13#10,'POSTORDEN:');
Postorden(raiz);

readln;
end.
```

Código en el lenguaje C/C++	Ejecución

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Nodo {
    int valor;
    struct Nodo* izq;
    struct Nodo* der;
} Nodo;
Nodo* crearNodo(int valor) {
    Nodo* nuevo = (Nodo*) malloc(sizeof(Nodo));
    nuevo->valor = valor;
    nuevo->izq = NULL;
    nuevo->der = NULL;
    return nuevo;
}
Nodo* insertar(Nodo* nodo, int x) {
    if (nodo == NULL) {
        return crearNodo(x);
    }
    if (x < nodo->valor) {
        nodo->izq = insertar(nodo->izq, x);
    } else {
        nodo->der = insertar(nodo->der, x);
    }
    return nodo;
```

```
PREORDEN:
10 8 5 9 13 12 20
INORDEN:
5 8 9 10 12 13 20
POSTORDEN:
5 9 8 12 20 13 10
```

```c
void preorden(Nodo* nodo) {
    if (nodo != NULL) {
        printf("%d ", nodo->valor);
        preorden(nodo->izq);
        preorden(nodo->der);
    }
}
void inorden(Nodo* nodo) {
    if (nodo != NULL) {
        inorden(nodo->izq);
        printf("%d ", nodo->valor);
        inorden(nodo->der);
    }
}
void postorden(Nodo* nodo) {
    if (nodo != NULL) {
        postorden(nodo->izq);
        postorden(nodo->der);
        printf("%d ", nodo->valor);
    }
}
void liberarArbol(Nodo* nodo) {
    if (nodo != NULL) {
```

```c
void liberarArbol(Nodo* nodo) {
    if (nodo != NULL) {
        liberarArbol(nodo->izq);
        liberarArbol(nodo->der);
        free(nodo);
    }
}
int main() {
    Nodo* raiz = NULL;

    raiz = insertar(raiz, 10);
    insertar(raiz, 8);
    insertar(raiz, 13);
    insertar(raiz, 5);
    insertar(raiz, 9);
    insertar(raiz, 12);
    insertar(raiz, 20);

            printf("PREORDEN:\n");
            preorden(raiz);
            printf("\nINORDEN:\n");
            inorden(raiz);
            printf("\nPOSTORDEN:\n");
            postorden(raiz);

    printf("\n");
    liberarArbol(raiz);
    return 0;
}
```

Código en el lenguaje Java	Ejecución
	PREORDEN: 10 8 5 9 13 12 20 INORDEN: 5 8 9 10 12 13 20 POSTORDEN: 5 9 8 12 20 13 10 BUILD SUCCESSFUL

```java
public class Arbol_binario {

    static class Nodo {
        int valor;
        Nodo izq, der;

        public Nodo(int valor) {
            this.valor = valor;
            izq = der = null;
        }
    }

    Nodo raiz;
    static public Nodo insertar(Nodo nodo, int x) {
        if (nodo == null) {
            return new Nodo(x);
        }
        if (x < nodo.valor) {
            nodo.izq = insertar(nodo.izq, x);
        } else {
            nodo.der = insertar(nodo.der, x);
        }
        return nodo;
    }
    static public void preorden(Nodo nodo) {
        if (nodo != null) {
                    System.out.print(nodo.valor + " ");
                    preorden(nodo.izq);
                    preorden(nodo.der);
        }
    }
 static public void inorden(Nodo nodo) {
     if (nodo != null) {
         inorden(nodo.izq);
                 System.out.print(nodo.valor + " ");
         inorden(nodo.der);
     }
 }
 static public void postorden(Nodo nodo) {
     if (nodo != null) {
         postorden(nodo.izq);
         postorden(nodo.der);
                 System.out.print(nodo.valor + " ");
     }
 }
 public static void main(String[] args) {
     Arbol_binario arbol = new Arbol_binario();
     arbol.raiz = arbol.insertar(arbol.raiz, 10);
     arbol.insertar(arbol.raiz, 8);
     arbol.insertar(arbol.raiz, 13);
     arbol.insertar(arbol.raiz, 5);
     arbol.insertar(arbol.raiz, 9);
     arbol.insertar(arbol.raiz, 12);
     arbol.insertar(arbol.raiz, 20);
             System.out.println("PREORDEN:");
                     arbol.preorden(arbol.raiz);
              System.out.println("\nINORDEN:");
                     arbol.inorden(arbol.raiz);
              System.out.println("\nPOSTORDEN:");
                     arbol.postorden(arbol.raiz);
 }
```
