**Reporte de Examen práctico.**

**Problema #: 20.- Insertar nodos a un arbol binario y recorrer en preorden, inorden y postorden. (50 puntos)**

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre del alumno(a): | Juan Carlos Ortiz Delgadillo | Fecha: | 04/09/2025 |

|  |  |
| --- | --- |
| Código en el lenguaje Fortran | Ejecución |
| **ArbolBinario.f90**  ! ArbolBinario.f90 program ArbolBinario  implicit none   type :: Nodo  integer :: valor = 0  type(Nodo), pointer :: izq => null()  type(Nodo), pointer :: der => null()  end type Nodo   type(Nodo), pointer :: raiz => null()  integer :: x, ios  character(len=1) :: pausa   print \*, 'Inserta enteros para el arbol (termina con -1):'   do  read(\*,\*, iostat=ios) x  if (ios /= 0) then  print \*, 'Entrada no valida.'  call liberar(raiz)  stop 1  end if  if (x == -1) exit  call insertar(raiz, x)  end do   write(\*,'(A)', advance='no') new\_line('a')//'Recorrido PREORDEN: '  call preorden(raiz)  print \*  write(\*,'(A)', advance='no') 'Recorrido INORDEN: '  call inorden(raiz)  print \*  write(\*,'(A)', advance='no') 'Recorrido POSTORDEN: '  call postorden(raiz)  print \*   call liberar(raiz)   print \*, new\_line('a'), 'Presiona ENTER para salir...'  read(\*,'(A)') pausa  contains   recursive subroutine insertar(raiz, v)  type(Nodo), pointer :: raiz  integer, intent(in) :: v  if (.not. associated(raiz)) then  allocate(raiz)  raiz%valor = v  raiz%izq => null()  raiz%der => null()  else if (v <= raiz%valor) then  call insertar(raiz%izq, v)  else  call insertar(raiz%der, v)  end if  end subroutine insertar   recursive subroutine preorden(r)  type(Nodo), pointer, intent(in) :: r  if (.not. associated(r)) return  write(\*,'(I0,1X)', advance='no') r%valor  call preorden(r%izq)  call preorden(r%der)  end subroutine preorden   recursive subroutine inorden(r)  type(Nodo), pointer, intent(in) :: r  if (.not. associated(r)) return  call inorden(r%izq)  write(\*,'(I0,1X)', advance='no') r%valor  call inorden(r%der)  end subroutine inorden   recursive subroutine postorden(r)  type(Nodo), pointer, intent(in) :: r  if (.not. associated(r)) return  call postorden(r%izq)  call postorden(r%der)  write(\*,'(I0,1X)', advance='no') r%valor  end subroutine postorden   recursive subroutine liberar(r)  type(Nodo), pointer :: r  if (.not. associated(r)) return  call liberar(r%izq)  call liberar(r%der)  deallocate(r)  nullify(r)  end subroutine liberar  end program ArbolBinario |  |

|  |  |
| --- | --- |
| Código en el lenguaje Pascal | Ejecución |
| **ArbolBinario.PAS**  program ArbolBinario; uses Crt;  type  PNodo = ^TNodo;  TNodo = record  valor: Integer;  izq, der: PNodo;  end;  procedure Insertar(var raiz: PNodo; v: Integer); begin  if raiz = nil then  begin  New(raiz);  raiz^.valor := v;  raiz^.izq := nil;  raiz^.der := nil;  end  else if v <= raiz^.valor then  Insertar(raiz^.izq, v) { duplicados a la izquierda }  else  Insertar(raiz^.der, v); end;  procedure Preorden(r: PNodo); begin  if r = nil then Exit;  Write(r^.valor, ' ');  Preorden(r^.izq);  Preorden(r^.der); end;  procedure Inorden(r: PNodo); begin  if r = nil then Exit;  Inorden(r^.izq);  Write(r^.valor, ' ');  Inorden(r^.der); end;  procedure Postorden(r: PNodo); begin  if r = nil then Exit;  Postorden(r^.izq);  Postorden(r^.der);  Write(r^.valor, ' '); end;  procedure Liberar(var r: PNodo); begin  if r = nil then Exit;  Liberar(r^.izq);  Liberar(r^.der);  Dispose(r);  r := nil; end;  var  raiz: PNodo;  x: Integer; begin  raiz := nil;   WriteLn('Inserta enteros para el arbol (termina con -1):');  while True do  begin  ReadLn(x);  if x = -1 then Break;  Insertar(raiz, x);  end;   WriteLn;  Write('Recorrido PREORDEN: ');  Preorden(raiz); WriteLn;  Write('Recorrido INORDEN: ');  Inorden(raiz); WriteLn;  Write('Recorrido POSTORDEN: ');  Postorden(raiz); WriteLn;   Liberar(raiz);   WriteLn;  Write('Presiona una tecla para salir...');  ReadKey; end. |  |

|  |  |
| --- | --- |
| Código en el lenguaje C/C++ | Ejecución |
| **ArbolBinario.cpp** #include <iostream> using namespace std;  struct Nodo {  int valor;  Nodo\* izq;  Nodo\* der;  Nodo(int v) : valor(v), izq(nullptr), der(nullptr) {} };  Nodo\* insertar(Nodo\* raiz, int v) {  if (!raiz) return new Nodo(v);  if (v <= raiz->valor) raiz->izq = insertar(raiz->izq, v); // dupes a la izquierda  else raiz->der = insertar(raiz->der, v);  return raiz; }  void preorden(Nodo\* r) {  if (!r) return;  cout << r->valor << " ";  preorden(r->izq);  preorden(r->der); }  void inorden(Nodo\* r) {  if (!r) return;  inorden(r->izq);  cout << r->valor << " ";  inorden(r->der); }  void postorden(Nodo\* r) {  if (!r) return;  postorden(r->izq);  postorden(r->der);  cout << r->valor << " "; }  void liberar(Nodo\* r) {  if (!r) return;  liberar(r->izq);  liberar(r->der);  delete r; }  int main() {  cout << "Inserta enteros para el arbol (termina con -1):\n";  ios::sync\_with\_stdio(false);  cin.tie(nullptr);   Nodo\* raiz = nullptr;  while (true) {  int x;  if (!(cin >> x)) {  cerr << "Entrada no valida.\n";  liberar(raiz);  return 1;  }  if (x == -1) break;  raiz = insertar(raiz, x);  }   cout << "\nRecorrido PREORDEN: ";  preorden(raiz);  cout << "\nRecorrido INORDEN: ";  inorden(raiz);  cout << "\nRecorrido POSTORDEN: ";  postorden(raiz);  cout << "\n";   liberar(raiz);  return 0; } |  |

|  |  |
| --- | --- |
| Código en el lenguaje Java | Ejecución |
| **ArbolBinario.java** import java.util.Scanner;  public class ArbolBinario {   static class Nodo {  int valor;  Nodo izq, der;  Nodo(int v) { valor = v; }  }   static Nodo insertar(Nodo raiz, int v) {  if (raiz == null) return new Nodo(v);  if (v <= raiz.valor) raiz.izq = insertar(raiz.izq, v);  else raiz.der = insertar(raiz.der, v);  return raiz;  }   static void preorden(Nodo r) {  if (r == null) return;  System.out.print(r.valor + " ");  preorden(r.izq);  preorden(r.der);  }   static void inorden(Nodo r) {  if (r == null) return;  inorden(r.izq);  System.out.print(r.valor + " ");  inorden(r.der);  }   static void postorden(Nodo r) {  if (r == null) return;  postorden(r.izq);  postorden(r.der);  System.out.print(r.valor + " ");  }   public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("Inserta enteros para el arbol (termina con -1):");   Nodo raiz = null;  while (true) {  if (!sc.hasNextInt()) {  System.err.println("Entrada no valida.");  return;  }  int x = sc.nextInt();  if (x == -1) break;  raiz = insertar(raiz, x);  }   System.out.print("\nRecorrido PREORDEN: ");  preorden(raiz);  System.out.print("\nRecorrido INORDEN: ");  inorden(raiz);  System.out.print("\nRecorrido POSTORDEN: ");  postorden(raiz);  System.out.println();     System.out.println("\nPresiona Enter para salir...");  try { System.in.read(); } catch (Exception e) {}  } } |  |