Práctica 19

Arboles Binarios de Búsqueda

1. Ejecute el programa ***Dev-C++***.
2. En el menú ***Archivo*** seleccione la opción ***Nuevo*** y a continuación ***Archivo Fuente***.
3. Transcriba el código del siguiente programa, mediante el cual se crea un árbol binario de búsqueda, que permite ingresar diferentes valores enteros de manera ordenada, los cuales podemos listar con diferentes criterios: PREORDEN, INORDEN o POSTORDEN, así como hacer una búsqueda de un determinado valor para ver si está o no en el árbol.
4. Analice la manera recursiva en que fue programado cada uno de los métodos.

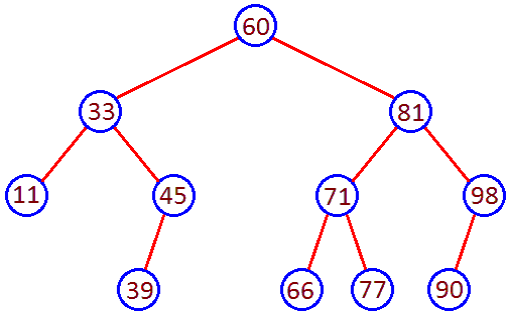
```
#include<iostream>
#include <iomanip>
using namespace std;

template<class T>
class Arbol;

template<class T>
class NodoArbol {
   private: NodoArbol<T> *HijoIzq;
            NodoArbol<T> *HijoDer;
            T Info;
   public:  NodoArbol();
            friend class Arbol<T>;
};

template<class T>
NodoArbol<T>::NodoArbol() {
   HijoIzq=NULL;
   HijoDer=NULL;
}

template<class T>
class Arbol {
   private: NodoArbol<T> *Raiz;
   public:  Arbol();
            NodoArbol<T> *RegresaRaiz();
            NodoArbol<T> *Busca(NodoArbol<T>*,T);
            void Inserta (T, NodoArbol<T>*);
            void Preorden (NodoArbol<T> *);
            void Inorden (NodoArbol<T> *);
            void Postorden (NodoArbol<T> *);
```

```
};

template<class T>
Arbol<T>::Arbol() {
   Raiz=NULL;
}

template<class T>
NodoArbol<T> * Arbol<T>::RegresaRaiz() {
   return Raiz;
}

template<class T>
NodoArbol<T> *Arbol<T>::Busca (NodoArbol<T> *Apunt, T Dato) {
   if(Apunt != NULL)
      if (Apunt->Info==Dato) return Apunt;
      else
         if(Apunt->Info>Dato) return Busca (Apunt->HijoIzq, Dato);
         else return Busca (Apunt->HijoDer, Dato);
   else return NULL;
}

template<class T>
void Arbol<T>::Inserta(T Dato,NodoArbol<T> *Apunt) {
   NodoArbol<T> *ApAux;
   if(Apunt!=NULL) {
      if(Dato<Apunt->Info) {
         Inserta(Dato,Apunt->HijoIzq);
         Apunt->HijoIzq=Raiz;
      }
      else
         if(Dato>Apunt->Info) {
            Inserta(Dato,Apunt->HijoDer);
            Apunt->HijoDer=Raiz;
         }
         if(Dato==Apunt->Info) cout<<endl<<"Nodo duplicado."<<endl;
         Raiz=Apunt;
      }
   else {
      ApAux=new NodoArbol<T>();
      ApAux->Info=Dato;
      cout<<endl<<"Nodo insertado con exito."<<endl;
      Raiz=ApAux;
   }
}
```

```
template <class T>
void Arbol<T>::Preorden (NodoArbol<T> *Apunt) {
   if (Apunt) {
      cout<<setw(6)<<Apunt->Info;
      Preorden(Apunt->HijoIzq);
      Preorden(Apunt->HijoDer);
   }
}

template <class T>
void Arbol<T>::Inorden (NodoArbol<T> *Apunt) {
   if (Apunt) {
      Inorden(Apunt->HijoIzq);
      cout<<setw(6)<<Apunt->Info;
      Inorden(Apunt->HijoDer);
   }
}

template <class T>
void Arbol<T>::Postorden (NodoArbol<T> *Apunt) {
   if (Apunt) {
      Postorden(Apunt->HijoIzq);
      Postorden(Apunt->HijoDer);
      cout<<setw(6)<<Apunt->Info;
   }
}

int main (void) {
   Arbol<int> Arb;
   NodoArbol<int> *Apu, *Regresa;
   char Opcion ;
   int Valor;
   do {
      cout<<endl<<"Implementacion de Arboles Binarios de Busqueda\n";
      cout<<"\n1.- Insertar \n2.- Buscar \n3.- Imprime preorden\n";
      cout<<"4.- Imprime inorden \n5.- Imprime postorden \n6.- Salir\n\n";
      cout<<"Opcion: ";
      cin>>Opcion;
      switch(Opcion) {
            case '1': cout<<endl<<"Introduce un valor: ";
                      cin>>Valor;
                      Apu=Arb.RegresaRaiz();
                      Arb.Inserta(Valor, Apu);
                      break;
            case '2': cout<<endl<<"Introduce un valor a buscar: ";
                      cin>>Valor;
```

```
                    Apu=Arb.RegresaRaiz();
                    Regresa=Arb.Busca(Apu, Valor);
                    if(Regresa) cout <<endl<<"El valor si se encuentra en el arbol."<<endl;
                    else cout <<endl<<"No existe el valor en el arbol."<<endl;
                    break;
            case '3': Apu=Arb.RegresaRaiz();
                    cout <<endl;
                    Arb.Preorden(Apu);
                    cout <<endl;
                    break;
            case '4': Apu=Arb.RegresaRaiz();
                    cout <<endl;
                    Arb.Inorden(Apu);
                    cout <<endl;
                    break;
            case '5': Apu=Arb.RegresaRaiz();
                    cout <<endl;
                    Arb.Postorden(Apu);
                    cout <<endl;
                    break;
        }
    } while(Opcion!= '6');
}
```

5. Ahora localice el icono de ***Compilar*** que está al inicio de la barra de herramientas y oprímalo, proporcione el nombre de ***ejercicio 19*** y elija el destino que guste para guardarlo.
6. Ejecute su programa e ingrese el árbol de la figura, siguiendo la siguiente secuencia: 60, 33, 45, 39, 11, 81, 98, 71, 90, 66, 77.



7. Liste los nodos según los tres criterios de ordenamiento: PREORDEN, INORDEN o POSTORDEN.
8. Intente insertar un nodo duplicado, haga diversas búsquedas de nodos existentes y no existentes.
9. Fin de la Práctica.