```
#include <iostream>
#include "bintree.h"
#include<queue>

using namespace std;


template <class T>
bool esHoja(const bintree<T> & A, const typename bintree<T>::node &v)
{
  return ( v.left().null()  && v.right().null() );
}

template <class T>
bool esInterno(const bintree<T> & A, const typename bintree<T>::node &v)
{
  return ( !v.left().null() || !v.right().null() );
}

template <class T>
void PreordenBinario(const bintree<T> & A,
typename bintree<T>::node v) {
  if (!v.null()) {
       cout << *v; // acciÃ³n sobre el nodo v.
       PreordenBinario(A, v.left());
       PreordenBinario(A, v.right());
  }
}

template <class T>
void InordenBinario(const bintree<T> & A,
 typename bintree<T>::node v)
{
  if (!v.null()) {
       InordenBinario(A, v.left());
       cout << *v; //acciÃ³n sobre el nodo v.
       InordenBinario(A, v.right());
  }
}

template <class T>
void PostordenBinario(const bintree<T> & A,
 typename bintree<T>::node v)
{
  if (!v.null()) {
       PostordenBinario(A, v.left());
       PostordenBinario(A, v.right());
       cout << *v; // acciÃ³n sobre el nodo v.
  }
}

template <class T>
void ListarPostNiveles(const bintree<T> &A, typename bintree<T>::node n) {
  queue<typename bintree<T>::node> nodos;
  if (!n.null()) {
    nodos.push(n);
    while (!nodos.empty()) {
      n = nodos.front(); nodos.pop();
      cout << *n;
      if (!n.left().null()) nodos.push(n.left());
      if (!n.right().null())
         nodos.push(n.right());
```



```
    }
  }
}


template <class T>
ostream & operator << (ostream & os, bintree<T> &arb)
{

cout << "Preorden:";

for (typename bintree<T>::preorder_iterator i = arb.begin_preorder(); i!=arb.end
_preorder(); ++i)
  cout << *i << " ";

cout << endl;
}

int main()
{  // Creamos el Ã¡rbol:
  //        7
  //     /    \
  //    1      9
  //  /  \    /
  // 6    8  5
  //       \
  //        4
typedef bintree<int> bti;
bintree<int> Arb(7);
Arb.insert_left(Arb.root(), 1);
Arb.insert_right(Arb.root(), 9);
Arb.insert_left(Arb.root().left(), 6);
Arb.insert_right(Arb.root().left(), 8);
Arb.insert_right(Arb.root().left().right(), 4);
Arb.insert_left(Arb.root().right(), 5);


cout << "Preorden:";

for (bintree<int>::preorder_iterator i = Arb.begin_preorder(); i!=Arb.end_preord
er(); ++i)
  cout << *i << " ";

cout << endl;


cout << "Inorden:";

for (bintree<int>::inorder_iterator i = Arb.begin_inorder(); i!=Arb.end_inorder(
); ++i)
  cout << *i << " ";

cout << endl;


cout << "Postorden:";

for (bintree<int>::postorder_iterator i = Arb.begin_postorder(); i!=Arb.end_post
order(); ++i)
  cout << *i << " ";

cout << endl;
```

```
cout << "Por Niveles:";
for (bintree<int>::level_iterator i = Arb.begin_level(); i!=Arb.end_level(); ++i
)
  cout << *i << " ";
cout << endl;

cout << Arb;









}
```