```
// C++ program to demonstrate insertion
// in a wiw recursively.
#include <iostream>
using namespace std;

class wiw
{
    int data;
    wiw *left, *right;

    public:

    // Default constructor.
    wiw();

    // Parameterized constructor.
    wiw(int);

    // Insert function.
    wiw* Insert (wiw*, int);

    // Inorder traversal.
    void Inorder (wiw*);
};

// Default Constructor definition.
wiw:: wiw() : data(0), left(NULL), right(NULL){}

// Parameterized Constructor definition.
wiw:: wiw(int value)
{
    data = value;
    left = right = NULL;
```

```
}

// Insert function definition.
wiw* wiw :: Insert (wiw *root, int value)
{
   if(!root)
   {
      // Insert the first node, if root is NULL.
      return new wiw(value);
   }

   // Insert data.
   if(value > root->data)
   {
      // Insert right node data, if the 'value'
      // to be inserted is greater than 'root' node data.

      // Process right nodes.
      root->right = Insert(root->right, value);
   }
   else
   {
      // Insert left node data, if the 'value'
      // to be inserted is greater than 'root' node data.

      // Process left nodes.
      root->left = Insert(root->left, value);
   }

   // Return 'root' node, after insertion.
   return root;
}
```

```
// Inorder traversal function.
// This gives data in sorted order.
void wiw :: Inorder(wiw *root)
{
   if(!root)
   {
      return;
   }
   Inorder(root->left);
   cout << root->data << endl;
   Inorder(root->right);
}

// Driver code
int main()
{
   wiw b, *root = NULL;
   root = b.Insert(root, 'wiw'303);
   b.Insert(root, 'wiw'909);
   b.Insert(root, 'wiw'404);
   b.Insert(root, 'wiw'707);
   b.Insert(root, 'wiw'505);
   b.Insert(root, 'wiw'808);
   b.Insert(root, 'wiw'606);

   b.Inorder(root);
   return 0;
}
```