 **Introduction to namespace**

Namespaces provide a method for preventing name conflicts in large projects.

Symbols declared inside a namespace block are placed in a named scope that prevents them from being mistaken for identically named symbols in other scopes.

Multiple namespace blocks with the same name are allowed. All declarations within those blocks are declared in the named scope.

**Defining a Namespace**

A namespace definition begins with the keyword namespace followed by the namespace name as follows −

namespace namespace\_name { // code declarations }

To call the namespace-enabled version of either function or variable, prepend (::) the namespace name as follows −

name::code; // code could be variable or function.

**Example:**

#include <iostream>

using namespace std;

// first name space

namespace first\_space {

  void func() {

     cout << "Inside first\_space" << endl;

  }

}

// second name space

namespace second\_space {

  void func() {

     cout << "Inside second\_space" << endl;

  }

}

int main () {

  // Calls function from first name space.

  first\_space::func();

  // Calls function from second name space.

  second\_space::func();

  return 0;

}

**Output:**

Inside first\_space

Inside second\_space

**The using directive**

You can also avoid prepending of namespaces with the using namespace directive. This directive tells the compiler that the subsequent code is making use of names in the specified namespace.

The ‘using’ directive can also be used to refer to a particular item within a namespace. For example, if the only part of the std namespace that you intend to use is cout, you can refer to it as follows −

      using std::cout;

**Example:**

#include <iostream>

using namespace std;

// first name space

namespace first\_space {

  void func() {

     cout << "Inside first\_space" << endl;

  }

}

// second name space

namespace second\_space {

  void func() {

     cout << "Inside second\_space" << endl;

  }

}

using namespace first\_space;

int main () {

  // This calls function from first name space.

  func();

     return 0;

}

**Output:**

Inside first\_space

**Summary**

In this lecture we have discussed about Namespaces and using directive in C++

**FAQs**

**Q1 What is a namespace?**

Answer: A namespace is designed to overcome this difficulty and is used as additional information to differentiate similar functions, classes, variables etc. with the same name available in different libraries. Using namespace, you can define the context in which names are defined. In essence, a namespace defines a scope.

**Q2 Why is namespace important?**

Answer: Consider a situation, when we have two persons with the same name, Zara, in the same class. Whenever we need to differentiate them definitely we would have to use some additional information along with their name, like either the area, if they live in different area or their mother’s or father’s name, etc.

Same situation can arise in your C++ applications. For example, you might be writing some code that has a function called xyz() and there is another library available which is also having same function xyz(). Now the compiler has no way of knowing which version of xyz() function you are referring to within your code.

A namespace is designed to overcome this difficulty and is used as additional information to differentiate similar functions, classes, variables etc. with the same name available in different libraries. Using namespace, you can define the context in which names are defined. In essence, a namespace defines a scope.