Namespace

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 or father 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. A namespace defines a scope.

Defining a Namespace:

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

```
namespace namespace_name
{

}
```

Example:

```
Namespace check
{
        int x;
        void show(int y)
        {
                cout<<y<<endl;
        }
}
```

Here the variable x and function show( ) are inside the scope defined by the namespace ‘check’. The value for x can be assigned using qualifying namespace name followed by scope resolution operator.

```
Check::x=10;
```

But always using the qualifier to access the variable becomes difficult. Hence, we use a ‘using” directive to simplify the access. The syntax is as:

```
using namespace name_of_namespace;
```

Example:

```
using namespace check;
x=2;
show(3);
```

here all the members declared within the namespace check can be accesses directly.

Example:

```
#include <iostream>
using namespace std;
namespace CSE
{
        int a=6;
        void display()
        {
                cout<<"hello"<<endl;
        }
}
int main()
{
        using namespace CSE;
        cout< <a<<endl;
        display();
        return 0;
}
```

In the above program variable 'a' and function 'display()' is defined in the scope CSE. Hence to use variable 'a' and function 'display()' in 'main()', we need to use the statement

using namespace CSE;

the other way of doing the same is by preceding the variable name and function name with namespace name followed by scope resolution operator as follows.

```
int main()
{
        cout<<CSE::a<<endl;
        CSE::display();
        return 0;
}
```

To define a function display function outside the namespace

Syntax:

```
Return_type namespace_name :: function_name(arglist)
{
}
```