```
Mylib1.cpp
    class ABC
    {

    };
    int power(int n , int m )
    {

    }
```

```
Mylib2.cpp
    class ABC
    {

    };
    double sqrt(double d)
    {

    }
```

```
If somebody uses both the file in same project
#include "mylib1.cpp"
#include "mylib2.cpp"
main()
{
        ABC obj; //Error
}
```

Namespace

- Namespaces allow us to group elements of C++/C# such as classes, function, pre-processor directives etc.
- A namespace definition begins with the keyword ***namespace*** followed by the namespace name as follows: Namespace declarations appear ***only at global scope***.
  Namespace name
  {
  }
- No need to give semicolon after the closing brace of definition of namespace.

```
Mylib1.cpp
namespace ABC
{     class ABC
      {

      };
      int power(int n , int m )
      {

      }
}
```

```
Mylib2.cpp
namespace BBC
{     class ABC
      {

      };
      double sqrt(double d)
      {

      }
}
```

```
If somebody uses both the file in same project
#include "mylib1.cpp"
#include "mylib2.cpp"
main()
{
        ABC:: ABC obj1;
        BBC:: ABC obj2;
}
```

- ***Benefits***
  - ***Easy search (Linking be efficient)***
  - Group the logically related Element
  - ***Minimize the name conflict problems***
  - Distribution of API become EASY

- We can access member of namespace by two ways
  - By using ***full qualifier name***
    ```
    main()
    {
            std::cout << "hello";
            ABC::ABC s1 ;
            BBC::ABC s2 ;
    }
    ```
  - ***using*** keyword
    ```
    using namespace ABC;
    main ()
    {
    ```

```
          ABC  s1,s2,s3 ;
          BBC::ABC x2;
 }
```

- ***Tips***
    - The syntax for creation of a namespace is similar to classes ***except for the semicolon.***
    - Declaration that falls outside all the namespace is still the member of ***global namespace.***
    - A namespace definition can continued over the multiple file

```
//A.CPP
namespace A
{
          char fun1();
          void display ();
}

//B.CPP
namespace A
{
          int var;
          void hello();
}
```

- Note
    - This continuation is called ***Extension - Namespace – Definition***

- We can give the ***alternative name for namespace***

```
namespace hardware_and_software_library
{
}
 namespace hwsw = hardware_and_software_library;
```

(4) Members of namespace is define inside the namespace or outside the name space

```
namespace hard_and_soft_library
{
        void f1()
        {
        }
        void f2();
 }
namespace hwsw = hard_and_soft_library
void hwsw::f2()
{
}
```

- **Namespace must global**

```
main ()
{           namespace a
            {}
}
```

- ***namespace can be nested***

```
namespace Z
{
                class x{};
                namespace P
                {
                            class o {};
                }
}
using namespace Z::P;
main()
{
      Z::x   m ;
      Z::P::o k;
      o k ;
}
```

e.g.

```
namespace A
{
            int factorial(int n)
            {
                        int i , f = 1 ;
                        for (  i = 1 ; i<= n ; i++ )
                        {
                                    f = f * i ;
                        }
                        return f;
            }
}
namespace B
{
            int power(int  a, int  b)
            {
                        int i , p = 1 ;
                        for (  i = 1 ; i<= b ; i++ )
                        {
                                    p = p* a ;
                        }
                        return p;
            }
}
#include<iostream>
using namespace std;
using namespace A;
main()
{
            int n , f ;
            cout << endl << "Enter number";
            cin >> n ;
            f = factorial(n);
            cout << endl << "factorial is " << f;

            int a , b , c;
            cout << endl << "Enter two values ";
            cin >> a  >> b ;
            c = B::power(a,b);
            cout << endl << "factorial is " << c;
}
```