

namespace

namespace란?

- ▶ 특정 영역(공간)의 범위를 지정하고 **이름을 붙여주는 것**
- ▶ namespace가 다르면 **같은 이름**의 선언이 허용된다
- ▶ 여러 외부 라이브러리 사용 중 발생하는 문제 해결을 위하여 만들어졌다

namespace

```
namespace A
{
    void Func()
    {
        std::cout << "namespcae A에서 정의한 함수" << std::endl;
    }
}
```

```
namespace B
{
    void Func()
    {
        std::cout << "namespcae B에서 정의한 함수" << std::endl;
    }
}
```

```
int main()
{
    A::Func();
    B::Func();
}
```

namespace

```
namespace A  
{  
    int a = 20;  
}
```

```
int main()  
{  
    int a = 10;  
    std::cout << a << std::endl;  
    std::cout << A::a << std::endl;  
}
```

namespace

▶ 범위지정 연산자(::)

```
int a = 1;
```

```
int main()  
{  
    int a = 10;  
    std::cout << a << std::endl;    //지역 변수 a : 10  
    std::cout << ::a << std::endl; //전역 변수 a : 1  
}
```

using

using이란?

- ▶ 어떤 namespace의 정보를 읽어 들일때 그 **이름을 무시하고 사용**하겠다
- ▶ 표준 namespace로 보는 사용 예
 - `using std::cout;`
 - `using namespace std;`

using

- ▶ 사용시 주의 점 : 모호함

```
namespace A
{
    int value = 10;
}
namespace B
{
    int value = 20;
}

using namespace A;
using namespace B;
int main()
{
    std::cout << value << std::endl; //namespace A? namespace B? Error!!
}
```


using

- ▶ 사용시 주의 점 : 모호함

```
int cout()  
{  
    return 1;  
}
```

```
using namespace std;  
int main()  
{  
    cout << "Hello world!!";  
}
```