Why we use, "using namespace std"?

Course Name

ICT 303: Object Oriented Programming (C++)

Course Teacher

Md. Sharif Hossen
Assistant Professor
Dept. of Information and Communication Technology
Comilla University

Contact

sharif5615@gmail.com

Referred Book

Theory and Problems of Data Structures by Seymour Lipschutz

Namespaces

Namespaces allow to group entities like classes, objects and functions under a name. This way the global scope can be divided in "sub-scopes", each one with its own name.

The format of namespaces is:

```
namespace identifier
{
entities
}
```

Where `identifier` is any valid identifier and `entities` is the set of classes, objects and functions that are included within the namespace. For example:

```
namespace myNamespace
{
  int a, b;
}
```

Namespaces

In this case, the variables `a` and `b` are normal variables declared within a namespace called `myNamespace`. In order to access these variables from outside the `myNamespace` namespace we have to use the scope operator `::`. For example, to access the previous variables from outside `myNamespace` we can write:

```
myNamespace::a
myNamespace::b
```

```
// namespaces
#include <iostream>
using namespace std;

namespace first
{
  int var = 5;
}

namespace second
{
  double var = 3.1416;
}

int main () {
  cout << first::var << endl;
  cout << second::var << endl;
  return 0;
}
```

```
5
3.1416
```

In this case, there are two global variables with the same name: `var`. One is defined within the namespace `first` and the other one in `second`. No redefinition errors happen thanks to namespaces.

Using

The keyword `using` is used to introduce a name from a namespace into the current declarative region. For example:

```
// using
#include <iostream>
using namespace std;

namespace first
{
  int x = 5;
  int y = 10;
}

namespace second
{
  double x = 3.1416;
  double y = 2.7183;
}

int main () {
  using first::x;
  using second::y;
  cout << x << endl;
  cout << y << endl;
  cout << first::y << endl;
  cout << second::x << endl;
  return 0;
}
```

```
5
2.7183
10
3.1416
```

Using

```
// using
#include <iostream>
using namespace std;

namespace first
{
  int x = 5;
  int y = 10;
}

namespace second
{
  double x = 3.1416;
  double y = 2.7183;
}

int main () {
  using namespace first;
  cout << x << endl;
  cout << y << endl;
  cout << second::x << endl;
  cout << second::y << endl;
  return 0;
}
```

```
5
10
3.1416
2.7183
```

Using

```
// using namespace example
#include <iostream>
using namespace std;

namespace first
{
  int x = 5;
}

namespace second
{
  double x = 3.1416;
}

int main () {
  {
    using namespace first;
    cout << x << endl;
  }
  {
    using namespace second;
    cout << x << endl;
  }
  return 0;
}
```

```
5
3.1416
```

Using

Namespace alias

We can declare alternate names for existing namespaces according to the following format:

```
namespace new_name = current_name;
```

Namespace std

All the files in the C++ standard library declare all of its entities within the `std` namespace. That is why we have generally included the `using namespace std;` statement in all programs that used any entity defined in `iostream`.

Header file:
#include<iostream>

```
namespace std{
    istream cin;
    ostream cout;
}
```