C++ for Science and Engineering COSC3000/6000

2018 Spring Semester

Part X
Name Space

1 Namespaces

- A **namespace** is a collection of name definitions,such as class definitions and variable declarations
  - If a program uses classes and functions written by different programmers, it may be that the same name is used for different things
  - Namespaces help us deal with this problem

1.1 The Using Directive

- **#include <iostream>** places names such as **cin** and **cout** in the **std** namespace
- The program does not know about names in the **std** namespace until you add

```
using namespace std;
```

  - if you do not use the **std** namespace, you can define **cin** and **cout** to behave differently

1.2 The Global Namespace

- Code you write is in a namespace
  - it is in the **global namespace** unless you specify a namespace
  - The global namespace does not require the using directive

1.3 Name Conflicts

- If the same name is used in two namespaces
  - The namespaces cannot be used at the same time
  - Example: If **my_function** is defined in namespaces **ns1** and **ns2**, the two versions of **my_function** could be used in one program by using local using directives this way:

```
{        // begin scope
        using namespace ns1;
        my_function( );
}        // end scope
```

```
{        // begin scope
        using namespace ns2;
        my_function( );
}        // end scope
```

1.3.1 Scope Rules For using

- A block is a list of statements enclosed in { }s
- The scope of a **using** directive is the block in which it appears
- A **using** directive placed at the beginning of a file, outside any block, applies to the entire file

1.4 Creating a Namespace

- To place code in a namespace
  - Use a namespace grouping

```
namespace Name_Space_Name
{
        ///////////////
        // Some_Code //
        ///////////////
}
```

- To use the namespace created
  - Use the appropriate using directive

```
using namespace Name_Space_Name;
```

1.4.1 Namespaces:Declaring a Function

- To add a function to a namespace
  - Declare the function in a namespace grouping

```
namespace cosc3000
{
        void greeting( );
}
```

1.4.2 Namespaces:Defining a Function

- To define a function declared in a namespace
  - Define the function in a namespace grouping

```
namespace cosc3000
{
        void greeting( )
        {
                cout << "Hello from namespace cosc3000.\n";
        }
}
```

1.4.3 Namespaces:Using a Function

- To use a function defined in a namespace
  - Include the using directive in the program where the namespace is to be used
  - Call the function as the function would normally be called

```
int main( )
{
        {        // begin scope
                using namespace cosc3000;
                greeting( );
        }        // end scope
        :
```

1.5 A Namespace Problem

Suppose you have the namespaces below:

```
namespace ns1
{
        fun1( );
        my_function( );
}
```

```
namespace ns2
{
        fun2( );
        my_function( );
}
```

Is there an easier way to use both namespaces considering that **my_function** is in both?

1.5.1 Qualifying Names

- Using declarations (not directives) allow us to select individual functions to use from namespaces

  -
    ```
    using ns1::fun1; //makes only fun1 in ns1 available
    ```
    - The scope resolution operator identifies a namespace here
    - Means we are using only namespace **ns1**'s version of **fun1**
  - If you only want to use the function once, call it like this
    ```
            ns1::fun1( );
    ```

1.5.2 Qualifying Parameter Names

- To qualify the type of a parameter with a using declaration
  - Use the namespace and the type name
    ```
    int get_number (std::istream input_stream);
    ```
    - **istream** is the **istream** defined in **namespace std**
    - If **istream** is the only name needed from **namespace std**, then you do not need to use
      ```
      using namespace std;
      ```
- Directive/Declaration
  - A using **declaration**
    ```
    using std::cout;
    ```
    makes only one name available from the namespace
  - A using **directive** makes all the names in the namespace available
- A using directive potentially introduces a name
- If **ns1** and **ns2** both define **my_function**,
  ```
  using namespace ns1;
  using namespace ns2;
  ```
  is OK, provided **my_function** is never used!
- A using declaration introduces a name into your code: no other use of the name can be made
  ```
  using ns1::my_function;
  using ns2::my_function;
  ```
  is illegal, even if **my_function** is never used

1.6 Naming Namespaces

- To avoid choosing a name for a namespace that has already been used
  - Add your last name to the name of the namespace
  - Or, use some other unique string

2 Example : define namespace

- Define namespace COSC3000
- Inculde TimeOfDay, vector2D classes in it.
- Modify sample37,38, and 39.cpp