

Namespaces-Enumerations

October 20, 2021

1 Namespaces

- <https://docs.microsoft.com/en-us/cpp/cpp/namespaces-cpp?view=msvc-160>
- a declarative region that provides a scope to the identifiers (variables, constants, functions, etc.)
- used to organize code into logical groups to prevent name collisions that can occur especially when your code base includes multiple libraries
- we've used C++ builtin `std namespace` in the previous chapters, e.g.

1.1 Creating namespaces

- C++ allows you to create your own namespaces
- syntax:

```
namespace NAME {  
    // declare names  
    // such as: constants, variables, functions, user-defined types, etc.  
}
```

```
[1]: #include <iostream>  
      #include <string>  
      using namespace std;
```

```
[2]: namespace MY_SPACE {  
      const float PI = 3.14156;  
      const double G = 6.67384e-11; // gravitational force in cubic metre per_  
      ↪second squared per kilogram  
      const double c = 2.99792458e8; // speed of light in vacuum in meter per_  
      ↪second  
      string first_name = "Jake";  
      string last_name = "Smith";  
  }
```

1.2 Accessing names from namespaces

- three different ways:
 1. use the fully qualified name with `::` namespace resolution operator
 - `namespace_name::identifier_name`
 2. use a `using` declaration to bring each identifier into current scope

3. use a `using` directive to bring everything in the namespace into current scope
– as we've done with `using namespace std;`

```
[3]: // area of circle with radius 4 unit
float area = MY_SPACE::PI*4*4; // #1
```

```
[4]: // can't use PI itself because it's not defined in the global scope
cout << PI;
```

```
input_line_11:3:9: error: use of undeclared identifier
```

```
'PI'; did you mean 'MY_SPACE::PI'?
```

```
cout << PI;
```

```
^~
```

```
MY_SPACE::PI
```

```
input_line_9:2:17: note: 'MY_SPACE::PI' declared
here
```

```
const float PI = 3.14156;
```

```
^
```

Interpreter Error:

```
[10]: // gives error on jupyter notebook...not sure why but should work!
MY_SPACE::first_name = "Johhny";
```

```
In file included from input_line_5:1:
```

```
In file included from
```

```
/Users/rbasnet/miniconda3/envs/cpp/include/xeus/xinterpreter.hpp:14:
```

```
/Users/rbasnet/miniconda3/envs/cpp/include/c++/v1/string:2321:67:
```

```
error: no member named 'value' in
```

```
'std::__1::integral_constant<bool, true>'
```

```
__alloc_traits::propagate_on_container_move_assignment::value>());
```

```
^
```

```
input_line_17:3:22: note: in instantiation of member
```

```
function 'std::__1::basic_string<char, std::__1::char_traits<char>,
```

```
std::__1::allocator<char> >::operator=' requested here
```

```
MY_SPACE::first_name = "Johhny";
```

```
^
```

Interpreter Error:

```
[6]: // let's try different way!
using MY_SPACE::first_name;

[7]: first_name = "Johnny";

[8]: //2. use a using declaration to bring each identifier into current scope
using MY_SPACE::c;

[9]: long dist = 100000;

[10]: double time_taken = dist/c; // time to travel 1000000 meters by light in vacuum

[11]: cout << "Light takes " << time_taken << " seconds to travel " << dist << "
↳meters.";
```

Light takes 0.000333564 seconds to travel 100000 meters.

```
[12]: // 3. use a using directive to bring everything in the namespace into current
↳scope
using namespace MY_SPACE;

[13]: cout << PI << " " << c << " " << first_name << endl;
```

3.14156 2.99792e+08 Johnny

2 Enumerations

- <https://docs.microsoft.com/en-us/cpp/cpp/enumerations-cpp?view=msvc-160>
- an enumeration is a user-defined type
- consists of a set of named integral constants that are known as enumerators

2.1 Defining enumeration types

- syntax to declare enumeration type:

```
enum TypeName {name1, name2, name3, ...};
```

- each name in an enum type is assigned an integral value that corresponds to its place in the order of the values listed
- by default, the first value is assigned 0, the next one is 1, and so on.
- however, you can explicitly set the value of an enumerator

```
[14]: enum Suit {Diamonds, Hearts, Clubs, Spades};
enum COLOR {RED, BLUE, GREEN, YELLOW};

[15]: enum SUIT {Diamonds=10, Hearts=20, Clubs=30, Spades=40};
```

2.2 Using enumeration types

- declare variables of enum types
- values of enum types must be one of the names in enumerations
 - similar to selecting one of the values from drop-down list on an online form

```
[16]: // since enumeration names are same in Suit and SUI enum types,  
      // you must use namespace qualifier to avoid ambiguity  
      Suit myCard = Suit::Clubs;
```

```
[17]: SUI best_suite = SUI::Spades;
```

```
[18]: COLOR my_favColor = RED;
```

```
[19]: // try assigning integer value  
      COLOR some_color = 0;
```

```
input_line_26:3:7: error: cannot initialize a variable  
of type 'COLOR' with an rvalue of type 'int'  
COLOR some_color = 0;  
    ~             ~
```

Interpreter Error:

```
[20]: // must explicitly cast type to a valid enumeration value  
      COLOR a_colr = (COLOR)0;
```

```
[21]: cout << a_colr;
```

0

```
[22]: cout << myCard;
```

2

```
[23]: cout << best_suite;
```

40

```
[24]: cout << my_favColor;
```

0

3 Applications and Exercises

- namespaces and enumeration types are crucial in large-scale software development using C++
- use these concepts as much as possible to learn and be familiar with them

[]: