Namespaces

Victor Eijkhout, Susan Lindsey

Fall 2025

last formatted: November 13, 2025

1. Namespaces in action

How do you indicate that something comes from a namespace?

Option: explicitly indicated.

```
1 #include <vector>
2 int main() {
3   std::vector<stuff> foo;
4 }
```

Import the whole namespace:

```
1 #include <vector>
2 using namespace std;
3 int main() {
4   vector<stuff> foo;
5 }
```

Good compromise:

```
1 #include <vector>
2 using std::vector;
3 int main() {
4   vector<stuff> foo;
5 }
```

2. Defining a namespace

Introduce new namespace:

```
1 namespace geometry {
2   // definitions
3   class vector {
4     // stuff
5   };
```

3. Namespace usage

Double-colon notation for namespace and type:

```
1 geometry::vector myobject();
```

or

```
1 using geometry::vector;
2 vector myobject();
```

or even

```
1 using namespace geometry;
2 vector myobject();
```

4. Example of using a namespace

Suppose we have a *geometry* namespace containing a `vector`, in addition to the `vector` in the standard namespace.

```
1 // namespace/geo.cpp
2 #include <vector>
3 #include "geolib.hpp" // this contains the geometry namespace
4 int main() {
5   // std vector of geom segments:
6   std::vector<geometry::segment> segments;
7   segments.push_back( geometry::segment( geometry::point(1,1),geometry::
      point(4,5) ) );
```

What would the implementation of this be?

5. Namespace'd declarations

```
1  // namespace/geolib.hpp
2  namespace geometry {
3    class point {
4    private:
5      double xcoord,ycoord;
6    public:
7      point() {};
8      point( double x,double y );
9      double x();
10     double y();
11   };
12   class vector {
13   private:
14     point from,to;
15   public:
16     vector( point from,point to);
17     double size();
18   };
19 }
```

6. Namespace'd implementations

```
1 // namespace/geolib.cpp
2 namespace geometry {
3   point::point( double x,double y ) {
4       xcoord = x; ycoord = y; };
5   double point::dx( point other ) {
6     return other.xcoord-xcoord; };
7     /* ... */
8   template< typename T >
9   vector<T>::vector( std::string name,int size )
10    : name_(name),std::vector<T>::vector(size) {};
11 }
```

7. Why not 'using namespace std'?

Illustrating the dangers of `using namespace std`:

This compiles, but should not:

```
1 // func/swapname.cpp
2 #include <iostream>
3 using namespace std;
4
5 def swop(int i,int j) {};
6
7 int main() {
8   int i=1,j=2;
9   swap(i,j);
10  cout << i << '\n';
11  return 0;
12 }
```

This gives an error:

```
1 // func/swapusing.cpp
2 #include <iostream>
3 using std::cout;
4
5 def swop(int i,int j) {};
6
7 int main() {
8   int i=1,j=2;
9   swap(i,j);
10  cout << i << '\n';
11  return 0;
12 }
```

Exercise 1

Add a `vector` class to the *geometry* namespace, which inherits from `std::vector`, and which makes the following code work:

```
1 // namespace/geo.cpp
2   std::vector<geometry::vector<float>> vectors;
3   vectors.push_back( geometry::vector<float>( "a", 5 ) );
4   cout << fmt::format("First vector, \"{}\" has size {}\n",
5                       vectors[0].name(),vectors[0].size());
```