Namespaces

Victor Eijkhout, Susan Lindsey

Fall 2024

last formatted: November 11, 2024

1. What is the problem?

Name conflicts:

- there is the `std::vector`
- you want to write your own geometry library with a `vector` class
  $\Rightarrow$ conflict
- also unintentional conflicts from using multiple libraries

2. Solution: namespaces

A namespace is a 'prefix' for identifiers:

```
std::vector xstd; // standard namespace
geo::vector xgeo; // my geo namespace
lib::vector xlib; // from some library.
```

3. Namespaces in action

How do you indicate that something comes from a namespace?

Option: explicitly indicated.

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

Import the whole namespace:

```
#include <vector>
using namespace std;
int main() {
  vector<stuff> foo;
}
```

Good compromise:

```
#include <vector>
using std::vector;
int main() {
  vector<stuff> foo;
}
```

4. Defining a namespace

Introduce new namespace:

```
namespace geometry {
  // definitions
  class vector {
    // stuff
  };
```

5. Namespace usage

Double-colon notation for namespace and type:

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

or

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

or even

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

6. 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 }
```

7. Guideline

- `using namespace` is ok in main program or implementation file
- Never! Ever! in a header file

Example

8. 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"
4 using namespace geometry;
5 int main() {
6   // std vector of geom segments:
7   std::vector< segment > segments;
8   segments.push_back( segment( point(1,1),point(4,5) ) );
```

What would the implementation of this be?

9. Namespace'd declarations

```
1 // namespace/geolib.hpp
2 namespace geometry {
3   class point {
4   private:
5     double xcoord,ycoord;
6   public:
7     point( double x,double y );
8     double dx(point);
9     double dy(point);
10  };
11  class segment {
12  private:
13    point from,to;
```

10. 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 }
```