

# Namespaces

Victor Eijkhout, Susan Lindsey

Fall 2023

last formatted: March 8, 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  
⇒ 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 {  
    };  
|
```

## 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:

```
// func/swapname.cpp
#include <iostream>
using namespace std;

def swop(int i,int j) {};

int main() {
    int i=1,j=2;
    swap(i,j);
    cout << i << '\n';
    return 0;
}
```

(Why?)

This gives an error:

```
// func/swapusing.cpp
#include <iostream>
using std::cout;

def swop(int i,int j) {};

int main() {
    int i=1,j=2;
    swap(i,j);
    cout << i << '\n';
    return 0;
}
```

## 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.

```
// namespace/geo.cpp
#include <vector>
#include "geolib.hpp"
using namespace geo;
int main() {
    // std vector of geom segments:
    std::vector< segment > segments;
    segments.push_back( segment( point(1,1),point(4,5) ) );
}
```

What would the implementation of this be?

## 9. Namespace'd declarations

```
// namespace/geolib.hpp
namespace geo {
    class point {
    private:
        double xcoord,ycoord;
    public:
        point( double x,double y );
        double dx(point);
        double dy(point);
    };
    class segment {
    private:
        point from,to;
```

## 10. Namespace'd implementations

```
// namespace/geolib.cpp
namespace geo {
    point::point( double x,double y ) {
        xcoord = x; ycoord = y; };
    double point::dx( point other ) {
        return other.xcoord-xcoord; };
    /* ... */
    template< typename T >
    vector<T>::vector( std::string name,int size )
        : _name(name),std::vector<T>::vector(size) {};
}
```