CS 3460

Introduction to Namespaces

Namespaces

- Motivation
  - Want to compare our `sin` function (`utilities.[hc]pp`) with the `sin` function in the standard library.
  - How do we make calls to these two functions, with the exact same name and parameters?
  - namespaces!!

Namespaces

- A way to organize code
  - Prevent naming conflicts
  - Semantic organization
- We’ve already been using namespaces: `std::`
- Have a similar flavor to Java packages, but not the same thing
  - Namespaces say nothing about code folders
  - Side Note: C++ 20 modules

Namespaces

- Wrap header code like…

```
namespace mymath
{
     double sin(double angle);
     void swap(int& x, int& y);
}
```

- Wrap implementation code like...

```
namespace mymath
{
    double sin(double angle)
    {
        return (4 * angle * (180 - angle)) / (40500 - angle * (180 - angle));
    }

    void swap(int& x, int& y)
    {
        int temp = x;
        x = y;
        y = temp;
    }

}
```

Namespaces

- Can eliminate the need for the namespace prefix

```
using namespace std;
```

- Never, ever, on your life, in a header file
- Strongly recommend against this at file scope
- Can be useful, and not ‘dangerous’ at function scope

Namespaces – More Info

- Namespaces may be nested
- Same namespace can be used in multiple files
- Can have several namespaces in one file
- Anonymous namespace by omitting the namespace identifier

namespaces – Code Demo

Namespace – Aliases

- Lets say we have this nested namespace

```
namespace outer
{
    namespace inner
    {
        namespace goodstuff
        {
            std::array<int, 4> getPrimes()
            {
                return { 2, 3, 5, 7 };
            }
        }
    }
}
```

- Two ways to call this function

```
auto primes = outer::inner::goodstuff::getPrimes();

namespace gs = outer::inner::goodstuff;   // namespace alias
auto primes = gs::getPrimes();
```