C++: 4_
Namespaces, Headers

Namespaces

- Allow grouping code so there are no name conflicts. For instance..

ms1.h

```
namespace MySpace1{
    void myFunc2();
}
```

ms2.h

```
namespace MySpace2{
    void myFunc2();
}
```

main.cpp

```
#include "ms1.h"
#include "ms2.h"
int main()
{

    MySpace1::myFunc2();
    MySpace2::myFunc2();
```

Namespaces

- Use ‘using’ construct – tells compiler to look in a particular namespace.

```
using namespace std;
```

- Allows cout instead of std::cout
- There are many namespaces. Wrap your code in namespaces if there is a chance that your functions have the same name as others (encrypt(),decrypt, open etc…)

Header File - Overview

- Break up large files, Speeds compilation process
- Organizes code
- Separates interface from implementation
- (and reduces your need to know what goes on ‘under the hood’)

- But adds slight complexity

Header File Rules –
1. YOU MUST USE INCLUDE GUARDS

```
//a.h
const int myInt=3;
//main.h
#include "a.h"  //define myInt here
#include "a.h"  //attempt to redefine
                //error C2370
```

No include guards you get multiply defined symbols

Instead wrap in an include guard

```
//a.h
#ifndef MY_UNIQUEID //if not included yet
#define MY_UNIQUEID //then define this symbol
                    //and include the const def
                    //next time included,
                    //MY_UNIQUEID defined
                    //so const def not included
    const int myInt=3;
#endif
```

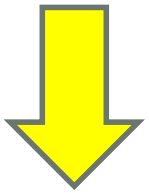
VC++ uses

```
//a.h
#pragma once    //only once
const int myInt=3;
```

Upshot:: ALWAYS USE INCLUDE GUARDS ON HEADERS

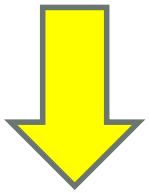
Header File Rules – Just declarations no definitions

declaration
In .h file



```
int a2();
```

definition
In .cpp file



```
int a2(){
    return 2;
}
```

Header File Rules – minimal exposure

In .h file

Only include those files necessary to make header self contained (that is no compiler errors).

```
#pragma once
//B function definitions
#include <string>

std::string b1();
std::string b2();

std::string b3();
```

In .cpp file

All other includes

Header File General Rules

- <> for system header files
- “” for your header files
- Only const variables unless part of a class
- Header file should contain only related stuff

- Never include a .cpp or source file
- Never put a “using namespace …” declaration in a header file

Header Files – Location (eclipse)

- Big projects – Organization is key
- Source in one dir, Headers in another
  - Use relative paths (ex. #include "../includes_usr/constants.h"
  - Let IDE find headers by specifying which directories to search