

Standard C++ Libraries

Standard C++ Include Files

Standard C++ include files have slightly different names than the include files we have been using, despite their containing the same information.

Old Name

iostream.h

iomanip.h

fstream.h

assert.h

ctype.h

float.h

limits.h

math.h

stddef.h

stdlib.h

string.h

New Name

iostream

iomanip

fstream

cassert

cctype

cfloat

climits

cmath

cstddef

cstdlib

cstring

New Name Rule:

Header files that are C++ related have the **.h** removed.

Header files that were part of C have the **.h** removed and the letter **c** inserted at the beginning.

Input/Output with Standard C++

Original program in C or non-standard C++:

```
#include <iostream.h>
void main( )
{
    cout << "Hello world!" << endl;
}
```

What happens if we substitute in the standard C++ library notation?

```
#include <iostream>
void main( )
{
    cout << "Hello world!" << endl;
}
```

It won't compile!!

Namespaces

- Namespaces allow a programmer to collect a group of identifiers, including constants and functions, into a named package.
- To use the identifiers in a program, the namespace of the identifier as well as the identifier must be specified.
- All of our familiar identifiers from the C libraries we have been using are contained in the **std** namespace in standard C++ (as well as others that we have not yet encountered).

Specifying a Namespace

There are three ways to specify a namespace of an identifier in a program.
We look at only two of them here:

1. **Use a qualified name:** Prefix each use an identifier in the namespace with the name of the namespace and a double colon.

Examples: `cout` is replaced with `std::cout`
`endl` is replaced with `std::endl`

2. **Insert a global “using directive”**

Example: `using namespace std;`

This allows any identifier in the `std` namespace to be used without the prefix

Two Solutions to Our Earlier Program Problem

Use qualified names:

```
#include <iostream>
void main( )
{
    std::cout << "Hello world!" << std::endl;
}
```

Use a using directive:

```
#include <iostream>
using namespace std;
void main( )
{
    cout << "Hello world!" << endl;
}
```

For our purposes, the using directive is the easiest method