Week 9 Tutorial

NWEN 241
Systems Programming

Alvin Valera

Alvin.valera@ecs.vuw.ac.nz

Moving from C to C++

hello.cpp

```
#include <stdio.h>
int main(void)
{
   printf("Hello world\n");
}
```

Programs written in C may be valid in C++

Compile: g++ hello.cpp –o helloex

Run: ./helloex

Moving from C to C++

hello.cpp

```
#include <cstdio>
int main(void)
{
   printf("Hello world\n");
}
```

If you must use C functions, include the C compatibility headers

Compile: g++ hello.cpp –o helloex

Run: ./helloex

Otherwise, use C++ facilities!

hello.cpp

```
#include <iostream>
int main(void)
{
   std::cout << "Hello world\n";
}
```

iostream is the header for standard I/O in C++

Compile: g++ hello.cpp –o helloex

Run: ./helloex

A much better version

hello.cpp

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

std::endl inserts ‘\n’ and flushes the buffer!

Compile: g++ hello.cpp –o helloex

Run: ./helloex

Primitive Data Types

Data Type	Keyword	Modifiers
Character	char	signed, unsigned
Integer	int	signed, unsigned, short, long, long long
Float	float	
Double	double	long
Boolean	bool	
Wide character representations	wchar_t	

Let’s write some code

- Write a simple C++ program that uses `bool`
  - Write a function that will determine whether a year is a leap year (or not)
  - In the `main()` function, ask user for input year and then say whether the year is a leap year on not

- Write a simple C++ program that uses `wchar_t`
  - Write a C++ program to display the message “Ngā mihi”

Namespace

- The scope of a namespace member is **local** to that namespace. All identifiers at namespace scope are visible to one another without qualification.
- Members are **not** visible **outside** its namespace.
- Everything not declared in another namespace/scope is in the global (program-wide) namespace.
- Two ways to access a namespace member outside its namespace:
  - Use `namespace_name::identifier` syntax
  - Use the `using` keyword to access specific or all members of a namespace

Example:

```
namespace myns
{
    const int N = 100;
    int count = 0;
    void printResult(){
    cout<<N;
    }
 }
```

Let’s write some code

- Write a simple C++ program to demonstrate namespace
  - Call the namespace `nwen241`, containing
    - An integer variable called `count`
    - A function called `greet()` which displays “Hello world.”
  - Create the `main()` function outside the namespace
    - The `main()` function should display `count` and call `greet()`
  - What will happen if we move `main()` into the namespace `nwen241`?
  - Rewrite the code to use “`using namespace`”

Defining a Class

- A class is a collection of fixed number of components called **members** of the class

- General syntax for defining a class:

```
class class_identifier {
      class_member_list
};
```

  - `class_member_list` consists of variable declarations and/or methods

Let’s write some code

- Write a simple C++ program to demonstrate the use of classes
  - Define a class called `GameCharacter` to represent a game character
  - Define the function implementations within the class declaration
  - Write a `main()` function that uses the above class
- First revision: Separate the class into a separate header file
- Second revision: Separate the function implementations into a separate source file
- Third revision: Make one of the functions explicit inline. How do you know if compiler granted the inline request?