Functions and Headers

Kate Gregory
www.gregcons.com/kateblog

pluralsight
hardcore developer training

Functions

- **Must be declared before they are called**
  - By implementing, or just by declaring
- **Have a return type**
- **Take parameters, which have a type and a name**
- **The compiler enforces type rules when you call a function**

```
int add(int x, int y)
{
return x+ y;
}
//...
int a = add(3,4);
```

Type Safety

- **When you call a function, the arguments you supply might be converted to the parameters the function takes**
  - Possibly risky – compiler will warn
- **The return value from the function might be converted as part of assigning the value**

Overloading

- **Imagine you wanted to add three numbers**
- **Should you write** `add3(double a, double b, double c)` **?**
  - And rename add to add2 ?
- **In C++ there is no need for this**
  - Two functions can have the same name as long as the compiler can tell them apart
  - Taking a different number of arguments is a great way to distinguish overloads
  - Return type can never be used to distinguish overloads
  - Taking the same number of arguments, but of different types, is risky

Multiple Files

- **A C++ application can be one giant .cpp file**
  - Requires compiling it all when making any changes
  - Difficult to co-ordinate the work of several developers
  - Difficult to find what you want to change in a 10,000 line file
- **In practice, you use multiple files**
  - Tell the compiler to compile each of them and the linker to link them
  - Mechanics for this vary from tool to tool
- **You must tell the compiler what is implemented in the other files if you plan to use it**

Header files

- **Long collections of declarations in many files have disadvantages**
  - Challenge to maintain
  - Hides the “more important” code
- **Solution – put them in a separate file that is included into each file as you compile**

```
#include
```

- **Anything that starts # is an instruction to the preprocessor – a step that runs before the compiler**
- **Result: code is neater, easier to understand, easier to maintain**

Two places to make mistakes

- **You can forget to declare a function before you call it**
  - Forget to include header
  - Function is not declared in header
- **This causes a compiler error**

- **You can forget to implement a function**
  - Code for function is not in the .cpp file
  - The .cpp file is not being linked
- **This causes a linker error**

- **Don't try to fix what isn't wrong**

Summary

- **Writing functions and calling them is better than a giant block of code**
  - Don't Repeat Yourself
  - Expressive Code
- **Functions can be implemented in a separate .cpp file**
  - But must be declared before they are used
- **Including a header file is an easy way to declare many functions at once**
- **Compiler errors and linker errors are caused by different mistakes**