Lecture 9

Chapter 4.2
Topics

• Overloading
• Signature
• Rules for resolving overloading
• Default arguments
Overloading

• Overloading is when there are two or more function definitions for one function name.
• Usually one would have to use different names for a function that sums 2 number vs. 3 numbers

```c
int total2(int one, int two)
    return one + two;
```

```c
int total3(int one, int two, int three)
    return one + two + three;
```
Overloading

• Overloading allows both functions to have the same name.

```cpp
int total(int one, int two)
    return one + two;
```

```cpp
int total(int one, int two, int three)
    return one + two + three;
```
Overloading

• But the two functions cannot differ only in the return type.
• They must have different signatures.
Signature

• In this case the signature is the name of a function and the following sequence of types.
  • int sum(double one, double two);
  • double sum(double two, double one);
  • int sum(int one, double two);
  • int sum(double one, int two);
  • int sum(int one, int two, int three);
Rules for resolving overloading

• The rules the compiler uses to for resolving which version of the overload to use is as follows:
  
  1. Exact matching: If the number and types of arguments exactly match a definition, then that is the definition used.

  2. Matching using automatic type conversion: If automatic type conversion can be used to fit one of the definitions then that match is used.

  3. If it is still ambiguous it will throw an error.
Default arguments

• It is possible to fill in input parameters into the function definition.
• This is useful if the programmer wants to be able to omit a parameter.

void showVolume(int length, int width = 1, int height = 1);