Functions: Optional Parameters and overloading

- We can have a single function that has different parameter lists
- We can have two functions with the same name but different parameter lists
- must be sufficiently different that the compiler can tell which one to use.

```
void myFunction ( int x, char c = 'a', float z = 1.5)
// The user can make any of the following calls:
myFunction ( 5 );
myFunction ( 5, 'b' );
myFunction ( 5, 1.8);
myFunction ( 1.8);
myFunction ( 1.8, 1.8);
myFunction (1.8, 'x', 1.9);
```

Functions: Optional Parameters and overloading

- We can have a single function that has different parameter lists - Optional Params

- We can have two functions with the same name but different parameter lists - Overloading

- must be sufficiently different that the compiler can tell which one to use.

```
void myFunction ( int x, char c = 'a', float z = 1.5)
// The user can make any of the following calls:
myFunction ( 5 );
myFunction ( 5, 'b' );
myFunction ( 5, 1.8);
myFunction ( 1.8);
myFunction ( 1.8, 1.8);
myFunction (1.8, 'x', 1.9);
```

When declaring functions with a single name, they must be designed so there is no ambiguity

2 meanings

The compiler decides which version to use based on params only.

```
void f1(int x);
int f1(float x);
   :
int i = f1(3); // could be error, because
               // compiler could choose the
               // void f1.
```

```
void fun(int x=1, int y=2, string z="blah");

fun(3);  // assigns x=3, y=2, z="blah"

fun(0, "hello"); // fails. If 0 fits for
                 // x, it assigns 0 to x
                 // and expects no more values,
                 // or a value for y.
```

Eg: Overloading

```
#include <iostream>
#include <string>
using namespace std;

void printDate(int mm, int dd = 1);
void printDate(string wkday = "Monday");

int main()
{
    printDate(5, 17);
    printDate();
    printDate("Tuesday");
    printDate(6);
}

void printDate(int mm, int dd)
{
    cout << mm << "\" << dd;
}
void printDate(string wkday)
{
    cout << wkday;
}
```

We can have multiple functions with the same name as long as every call will be completely unambiguous, by virtue of the parameters passed.