

# Chapter 10 – Function Overloading

C++ compilers are versatile. One reason is that function names can be *overloaded*. This means that more than one function can exist in a program with the same name and the compiler will select the correct one to use for any given call. In order for the compiler to select the correct function among several functions with the same name, the function headers must differ in some way. This can be the number of arguments, argument types, and return type. For example, functions named *add* could accept two, three, or four integers to add.

```

int Add(int a, int b)      int Add(int a, int b, int c)      int Add(int a, int b, int c, int d)
{
    return a + b;
}
    {
        return a + b + c;
    }
    {
        return a + b + c + d;
    }

```

Calls to the functions would cause the compiler to choose the correct function based on the number of arguments in each call. For example:

```

w = Add(1, 2);           w will receive 3
x = add(1, 2, 3);       w will receive 6
y = add(1, 2, 3, 4);    y will receive 10

```

## Exercises

1. What is function overloading?
2. How does the compiler choose the correct overloaded function to call?

3. What is the output of the following program?

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

int Left(int a, int b);
int Left(int a);

void main() {
    cout << Left(3) << endl;
    cout << Left(4, 2) << endl;
}

int Left(int a, int b) {
    return a / b;
}

int Left(int a) {
    return a * a;
}
```

### **Programming Assignment 10.1**

Create and use three functions within a program that calculates the volume of a cube, a box, and a cylinder. Use the following function headers in the program:

```
double Volume(double side) // for the cube
double Volume(double length, double width, double depth ) // for the box
double Volume(double length, double radius) // for the cylinder
```