Functions

- A function is a group of statements that together perform a task.
- Every C++ program has at least one function, which is main()
- The general form of a C++ function definition is as follows:

```
return_type function_name( parameter list )
{
        body of the function
}
```

- A function declaration tells the compiler a function's name, return type, and parameters, e.g.

```
int max(int num1, int num2);
```

- Function declarations become more important when we look at header files (.h) later.
- A function definition provides the actual body of the function, e.g.

```
int max(int num1, int num2)
{
      int result;
      if (num1 > num2)
      {
            result = num1;
      }
      else
      {
            result = num2;
      }
      return result;
}
```

- Function calls are when you call your function definition, e.g.

```
int test = max(66, 88);
cout << test << endl;
```

- Use a return type of `void` when your function does not need t return a value.
- This pause() function is a good example of a function without a return value.

```
void pause(int seconds)
{
      long startTime, currentTime;
      startTime = time(NULL);
      do
      {
            currentTime = time(NULL);
      }
      while ((currentTime - startTime) < seconds);
}
```

Default Arguments [C++]

- In C++ functions may have default arguments for parameters that may be omitted. Sometimes a function argument is “almost always” the same and it is pleasanter to mention it only when unusual.

```
// A C++ function with default arguments,
//  it may be called with 2 arguments
//  or 3 arguments or 4 arguments.
int sum(int x, int y, int z = 0, int w = 0)
{
     return (x + y + z + w);
}
```

Overloading Functions [C++]

- C++ allows function names to be “overloaded.” That is, the same function name or operator can be given several different definitions; the types and number of arguments supplied determine which definition is used, e.g.

```
void print(int i) {
     cout << "Printing int: " << i << endl;
}
void print(float  f) {
     cout << "Printing float: " << f << endl;
}
void print(char c) {
     cout << "Printing character: " << c << endl;
}

int iData = 5;
print(iData);
float fData = 5.555;
print(fData);
char cData = 'Z';
print(cData);
```

- A neat feature of C++ is that it also allows the overloading of operators—the special symbols like “+” and “<<”. Though this feature will be covered later when taking about C++ classes.

Example of checking user input

- Functions (and loops) are useful for creating input functions for ensuring user input.
- Here is an "ask" function that ensures user input of an `int` between a `min` and `max` value.

```
int askInt(int min, int max)
{
      int value = 0;
      while (true)
      {
            cout << "Enter a number from " << min <<
                  " to " << max << ": ";
            cin >> value;
            if (value >= min && value <= max)
            {
                  return value;
            }
            cout << "Invalid value. Try again. " << endl;
      };
}
```