- **Modular programming**: Breaking a program into smaller, manageable parts
  - Improves maintainability of programs
  - Simplifies process of writing code
- **Function**: A collection of statements to preform a task
- `main` can call any number of function
  - Functions can call any other functions
- Global Variables (non constants) are automatically 0, or Null
- `static` local variables retain their contents between function calls
- `static` local variables are defined and initialized only during the first function execution.

```cpp
#include <iostream>

class MyClass {
public:
    void count() {
        static int num = 0; // Initialized to 0 only once, on the first call
        num++;
        std::cout << "Count is: " << num << std::endl;
    }
};

int main() {
    MyClass obj;
    obj.count(); // Prints "Count is: 1"
    obj.count(); // Prints "Count is: 2"
    obj.count(); // Prints "Count is: 3"
    return 0;
}
```

- When using default arguments, put non default first.

```
int getSum(int, int=0, int=0);// OK
int getSum(int, int=0, int);  // NO
```

- Reference variables (`&`) can be used and implemented within variables to change an existing variable

Program 6-25

The & here in the prototype indicates that the parameter is a reference variable.

```
1  // This program uses a reference variable as a function
2  // parameter.
3  #include <iostream>
4  using namespace std;
5
6  // Function prototype. The parameter is a reference variable.
7  void doubleNum(int &);
8
9  int main()
10 {
11    int value = 4;
12
13    cout << "In main, value is " << value << endl;
14    cout << "Now calling doubleNum..." << endl;
15    doubleNum(value);
16    cout << "Now back in main. value is " << value << endl;
17    return 0;
18 }
```

Here we are passing value by reference.

The & also appears here in the function header.

```
20 //*********************************************************
21 // Definition of doubleNum.                               *
22 // The parameter refVar is a reference variable. The value *
23 // in refVar is doubled.                                  *
24 //*********************************************************
25
26 void doubleNum (int &refVar)
27 {
28    refVar *= 2;
29 }
```

Program Output

```
In main, value is 4
Now calling doubleNum...
Now back in main. value is 8
```

- A reference variable can be called like `int& num` or `int &num`, no difference.
- Function overload are functions with a same name and different parameters.
- `exit()` automatically exits the program and can be called from anywhere.
  - Requires the library `cstdlib`