C++ Reference Guide – Updated 19 October 2017

Variable Declarations

Variables must be declared before they are used. Variables can be declared in a function, in an if block, or in a loop. Variables can only be used where they were declared (scope). Global variables (declared outside of any function) are not allowed. If a function needs to use a variable from another function, then it must be passed as a parameter.

```
int age;                        // integer
int age = 0;                    // integer initialized
float temp;                     // single precision real number
double distance;                // double precision real number
float temp = 2.3;               // real number initialized (use decimal)
bool valid;                     // boolean
bool valid = true;              // boolean initialized (true or false)
char letter;                    // char
char letter = 'A';              // char initialized (single quotes)
char text[256];                 // char array up to 256 characters
                                //   (including terminating '\0')
char text[256] = "Hi";          // char array initialized
```

Expressions

Expressions combine variables and literals with numeric and boolean operators. Order of Operation Rules define which operators are evaluated first. Expressions can be assigned to variables (using the = operator) or used in If and Loop conditions.

```
2 * (x + y)                     // Parenthesis performed first
x++                             // Add 1 to x
x--                             // Subtract 1 from x
2 * x++                         // Add 1 to x after doing the
                                //   expression
2 * ++x;                        // Add 1 to x before doing the
                                //   expression
!a                              // true if a is false ("not")
x * y;                          // Multiply
x / y;                          // Division (integer division rounds
                                //   down). Cannot divide by 0.
x % 2;                          // Modulo (remainder). Cannot modulo
                                //   by 0.
x + y;                          // Addition
x - y;                          // Subtraction
x > y                           // true if x is greater than y
x >= y                          // true if x is greater than or equal
                                //   to y
x < y                           // true if x is less than y
x <= y                          // true if x is less than or equal to y
x == y                          // true if x is equal to y (2 equal signs)
x != y                          // true if x does not equal y
a && b                          // true if both a and b are true ("and")
a || b                          // true if either a or b are true ("or")
x = 2;                          // Set variable previously declared
y += 2;                         // Add the value on the right to the
                                //   variable on the left
y -= 2;                         // Subtract the value on the right from
                                //   the variable on the left
y *= 2;                         // Multiply the variable on the left with
                                //   the value on the right
y /= 2;                         // Divide the variable on the left by the
                                //   value on the right (not 0)
y %= 2;                         // Modulo (remainder) the variable on the
                                //   left by the value on the right (not 0)
```

If Blocks

```
if (x >= 0 && x < 100)          // Boolean expression in parenthesis. No
{                               //   semicolon after parenthesis.
   // Do something
}
else if (x < 200)               // Considered only when previous if
{                               //   resulted in false
   // Do something
}
else if (x < 300)
{
   // Do something
}
else                            // All other values
{
   // Do something
}
```

Loops

```
while (onGround)                // While Loop. No semicolon after
{                               //   parenthesis.
   // Do something
}

do                              // Do While Loop
{
   // Do something at least once
}
while (selection != 0);         // Semicolon is required.

for (int i=0; i<100; i++)       // For Loop (going up)
{
   // Do something 100 times. The variable i will go from 0 to 99.
}

for (int i=99; i>=0; i--)       // For Loop (going down)
{
   // Do something 100 times.  The variable i will go from 99 to 0
}
```

Miscellaneous

```
#include <iostream>             // cout, cin
#include <iomanip>              // setw
#define PI 3.14                 // Replace PI with 3.14 in code
using namespace std;            // Avoid putting "std::" in front of code
```

Screen Input/Output

```
cout << "Hello World";          // Print text
cout << endl << "\n";           // Print newlines
cout << "\t\n\"\\";             // Print tab, newline, double quote,
                                //   and backslash
cout << "Value = " << value;    // Print text and variable
cout << setw(5) << value;       // Right align value by 5 spaces
cout.setf(ios::fixed);          // Display floats as decimals without
                                //   scientific notation
cout.setf(ios::showpoint);      // Display the decimal point for floats
cout.precision(2);              // Set float precision to 2
cin >> number;                  // Read number from keyboard and store
                                //   in variable
```

File Input/Output

```
ifstream fin("myfile.txt");     // Create a stream variable to read
                                //   from myfile.txt
ofstream fout("myfile.txt");    // Create a stream variable to write to
                                //   myfile.txt
fin.fail()                      // True if failure to read (or write)
                                //   to file
fin >> text;                    // Read word from file stream and put in
                                //   variable
fout << text;                   // Write word from a variable to a file
                                //   stream
```

Functions

Functions must be declared before they are used:

1. Put functions in order with main at the end; or
2. List function prototypes (with semicolons) before main with function declarations in any order after main

Three Types of Parameters:

1. Input passed by value. The variable in the calling function will not get updated.
2. Output returned (specify data type only). The calling function must use the "=" sign to store the result.
3. Input/Output passed by reference (use the "&" in the function declaration). The variable in the calling function will get updated.

Examples of Functions with the Three Types of Parameters:

1. No input or output parameters:

```
void myFunction1()
{
   // Do something
}
```

2. One output parameter (return) and no input parameters:

```
int myFuntion2()
{
   int x = 0;
   // Do something to set x
   return x;
}
```

3. One input parameter (passed by value) and no output parameters:

```
void myFunction3(int param)
{
   // Do something with param
}
```

4. Two input parameters (passed by value) and one output parameter (return):

```
float myFunction4(int param1, int param2)
{
   float y = 0.0;
   // Do something with param1 and param2 to set y
   return y;
}
```

5. One input parameter (passed by value) and one output parameter (passed by reference):

```
void myFuntion5(int param, float &result)
{
   // Do something with param to update result
}
```

6. One input parameter (passed by value) and two output parameters (one passed by reference and one return):

```
bool myFunction6(int param, float &result)
{
   bool result = false;
   // Do something with param to update result
   // Do something to set result
   return result;
}
```

Calling the example functions:

```
int main()
{
   myFunction1();
   int result = myFunction2();
   myFunction3(result);
   float value = myFunction4(result, 100);
   myFunction5(72, value);
   bool error = myFunction6(99, value);
   return 0;
}
```