Iterative Statements: While, For, do-while

C++ Notes Mrs. Alano

Loops or **iterative** statements tell the program to repeat a fragment of code several times or as long as a certain condition holds. Without iterations, a programmer would have to write separately every instruction executed by the computer, and computers are capable of executing millions of instructions per second. Instead, programmers can use fewer instructions which the computer will repeat numerous times.

C++ offers three convenient iterative statements: *while, for* and *do-while.* Strictly speaking, any iterative code can be implemented using only the *while* statement, but the other two add flexibility and make the code more concise and idiomatic.

For Loops:
Is a shorthand for the *while loop* that combines initialization, condition, and increment in one statement.
General Form:

```
for (initialization; condition; increment)
{
     statement1;
     statement2;
}
```

- **Initialization** is a statement that is always executed once before the first pass through the loop.
- **Condition** is tested before each pass through the loop
- **Increment** is a statement executed at the end of each pass through the loop.

Example: returns the sum of all integers from 1 to n, if n >= 1, and 0 otherwise.

```
//n previously declared and given a value from the user
int sum = 0;
int i;
for (i = 1; i <= n; i++)
{
     sum += i;
}
//value of sum is output to the use in some way
```

While Loop
General Form:

```
while (condition)
{
     statement1;
     statement2;
}
```

- condition can be any arithmetic or logical expression.
- It is evaluated exactly the same way as an if statement
- Statements inside braces are called the **body**
- If the body consists of **only one** statement, the surrounding braces can be dropped

```
while (condition)
     statement1;
```

****NOTE****

It is important **NOT** to put a semicolon after the while (condition). With a semicolon, the body of the loop would be an empty statement, which would leave *statement1* completely out of the loop.

Three elements must be present, in one form or another, with any WHILE loop: initialization, a test of the condition, and incrementing.

1. Initialization: the variables tested in the *condition* must first be initialized to some values. For example: `int i = 1`
2. Testing: The condition is tested before each pass through the loop. If it is false, the body is not executed, iterations end, and the program continues with the next statement after the loop. If the condition is false at the very beginning, the body of the *while* loop is not executed at all.
3. Incrementing: At least one of the variables tested in the condition must change within the body of the loop. Otherwise, the loop will be repeated over and over and never stop, and your program will **hang.** (use `i++`)

Do-While Loop
The do-while loop differs from the *while* loop in that the condition is tested *after* the body of the loop. This assures that the program goes through the iteration *at least once*.

General Form:

```
do
{
     ….
}
while(condition);     //Semicolon is at the end of this loop
```

The program repeats the body of the loop as long as *condition* remains true. It is better always to keep the braces with this type of loop.

****NOTE**** **This is the type of loop you might want to use with a game or a program that asks the user if they would like to do that again?**