

Loops

Today

- String review
- Pre/post increment (pay attention to this)
- While loops
- do While Loop

Due this week

- **Recitation 3**
- **Homework 3**
 - Write solutions in VSCode and paste in Autograder, **Homework 3 CodeRunner**.
 - Zip your .cpp files and submit on canvas **Homework 3**.
- Start going through the textbook readings and watch the videos
 - **Quiz 4**

String Input

Reading two (whitespace separated) strings with cin

```
cout << "Please enter your name: ";  
string fname, lname;  
cin >> fname >> lname;
```

```
//fname gets Harry, lname gets Potter
```

String Input

`getline()` function allows us to accept a full string input (with whitespace)

```
cout << "Please enter your name: ";  
string name;  
getline(cin, name);
```

```
//name gets Harry Potter
```

String Functions

- The `length` *member function* yields the number of characters in a string.
- Unlike the `sqrt` or `pow` function, the `length` function is *invoked* with the *dot notation*:

```
string name = "Harry";  
int n = name.length();
```

String Data Representation & Character Positions

H	e	l	l	o	,		W	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11	12

- In most computer languages, the starting position 0 means “start at the beginning.”
- The first position in a string is labeled 0, the second 1, and so on. And don’t forget to count the space character after the comma—but the quotation marks are **not** stored.
- The position number of the last character is always one less than the length of the **string**.

Strange... but will be on the test....

Incrementing results in a return value!

1) Pre-increment operator: A pre-increment operator is used to increment the value of a variable before using it in an expression. In the Pre-Increment, value is first incremented and then used inside the expression.

Syntax:

```
a = ++x;
```

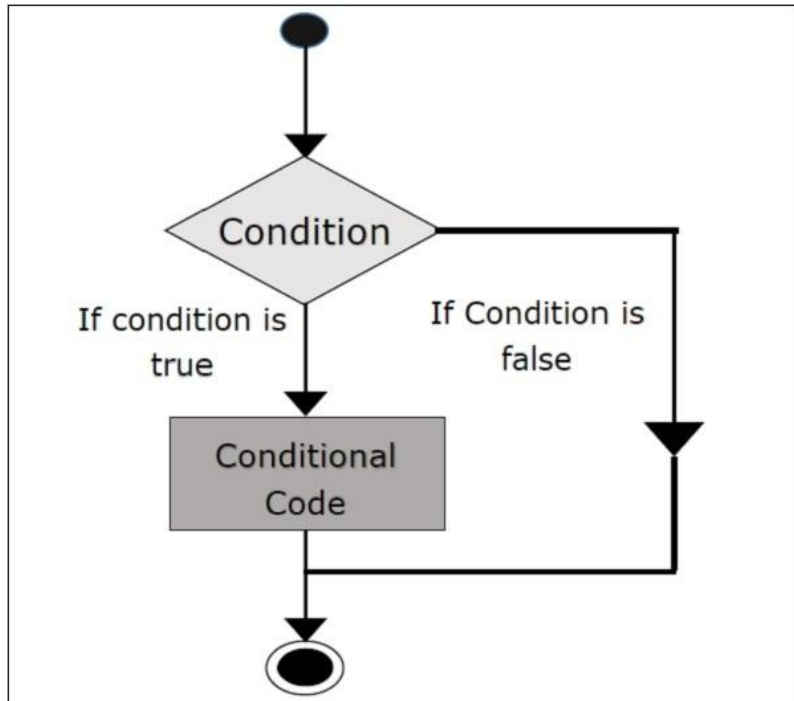
Here, if the value of 'x' is 10 then the value of 'a' will be 11 because the value of 'x' gets modified before using it in the expression.

2) Post-increment operator: A post-increment operator is used to increment the value of the variable after executing the expression completely in which post-increment is used. In the Post-Increment, value is first used in an expression and then incremented.

Syntax:

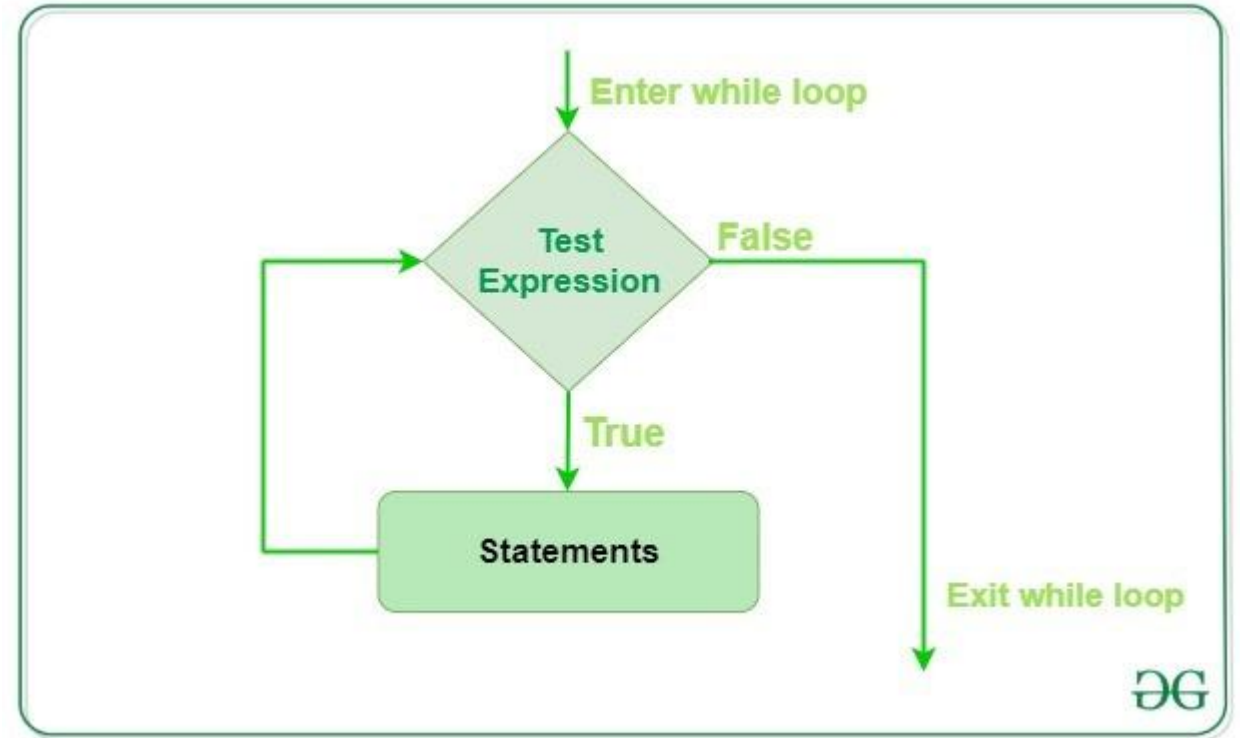
```
a = x++;
```

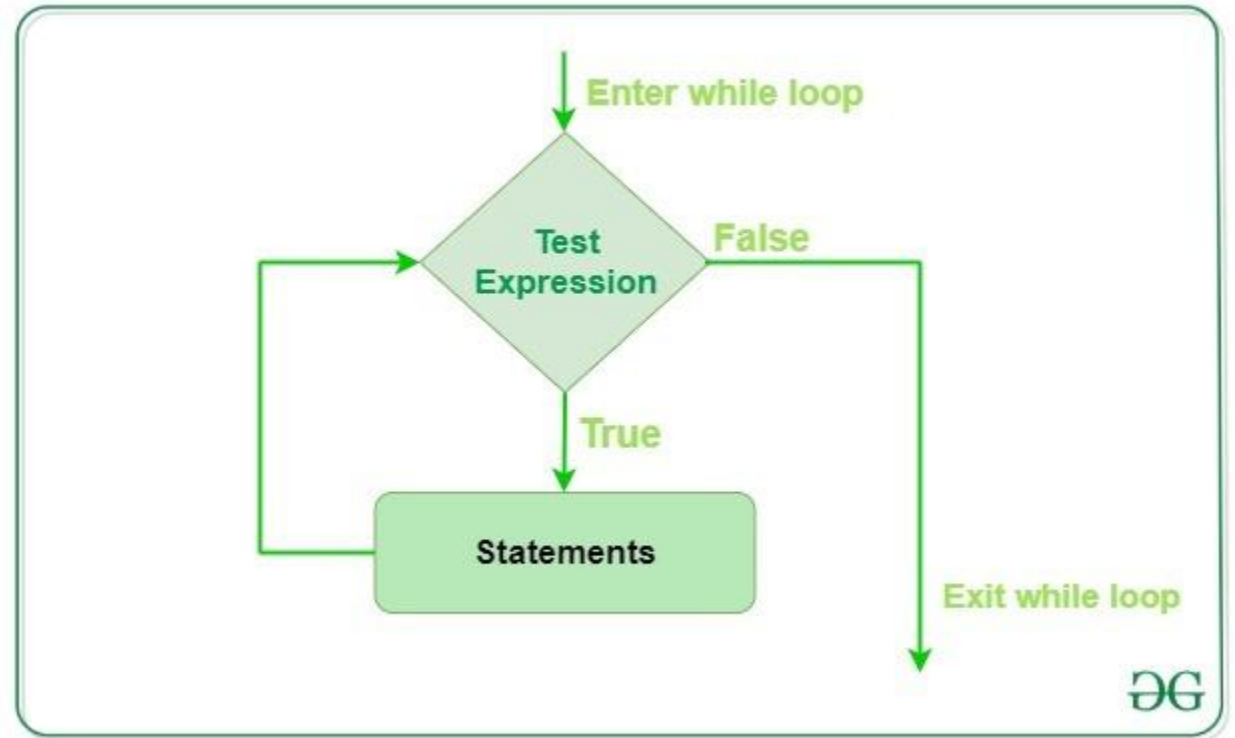
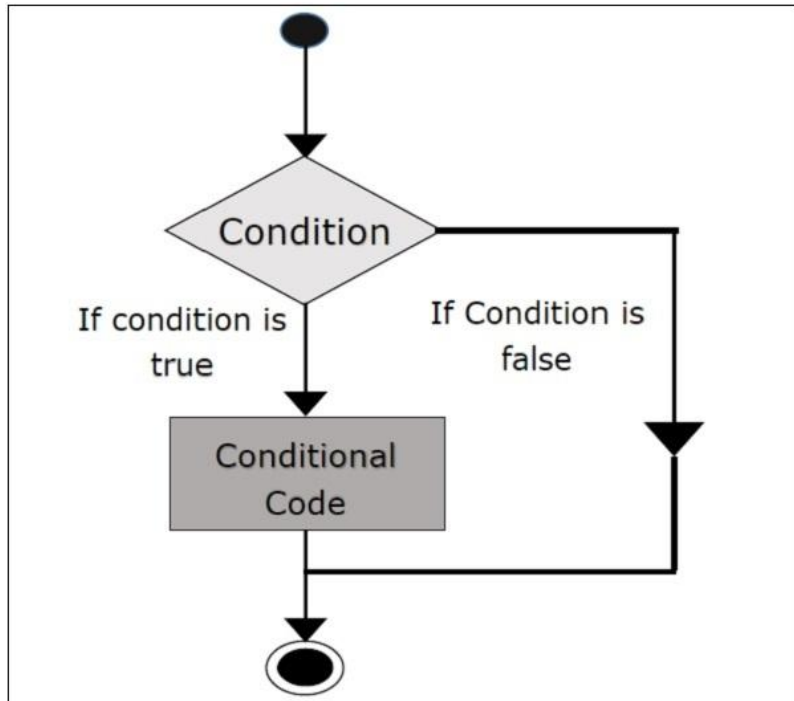
Here, suppose the value of 'x' is 10 then the value of variable 'a' will be 10 because the old value of 'x' is used.



If statement

While loop






Where to?



Loops


The *while* Loop Syntax


This variable is defined outside the loop and updated in the loop.

If the condition never becomes false, an infinite loop occurs.  See page 136.


```
double balance = 0;  
.  
.  
.  
while (balance < TARGET)  
{  
    year++;  
    double interest = balance * RATE / 100;  
    balance = balance + interest;  
}
```


This variable is created in each loop iteration.

Beware of "off-by-one" errors in the loop condition.  See page 137.

Don't put a semicolon here!  See page 80.

These statements are executed while the condition is true.

Lining up braces is a good idea.  See page 79.

Braces are not required if the body contains a single statement, but it's good to always use them.  See page 80.

while Loop Examples		
Loop (all preceded by i=5;)	Output	Explanation
while (i > 0) { cout << i << " "; i--; }	5 4 3 2 1	When i is 0, the loop condition is false, and the loop ends.
while (i > 0) { cout << i << " "; i++; }	5 6 7 8 9 10 11 ...	The i++ statement is an error causing an “infinite loop” (see Common Error 4.1).
while (i > 5) { cout << i << " "; i--; }	(No output)	The statement i > 5 is false, and the loop is never executed.
while (i < 0) { cout << i << " "; i--; }	(No output)	The programmer probably thought, “Stop when i is less than 0”. However, the loop condition controls when the loop is executed, not when it ends (see Common Error 4.2).
while (i > 0); { cout << i << " "; i--; }	(No output, program does not terminate)	Note the <u>semicolon</u> before the {. This loop has an empty body. It runs forever, checking whether i > 0 and doing nothing in the body.

Example of Normal Execution

while loop to hand-trace

What is the output?

```
i = 5;
while (i > 0)
{
    cout << i << " ";
    i--;
}
```

..

Example of a Problem – An Infinite Loop

The output never ends

- *i* is set to 5
- The *i++*; statement makes *i* get bigger and bigger
- the condition will never become false –
- an infinite loop

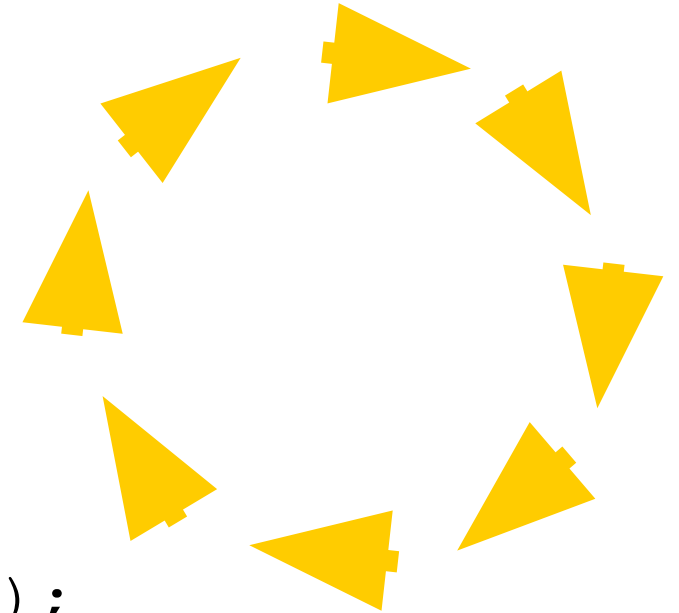
```
i = 5;
while (i > 0)
{
    cout << i << " ";
    i++;
}
```

5 6 7 8 9 10 11...

Common Error – Infinite Loops

- Forgetting to update the variable used in the condition is common.
- In the investment program, it might look like this:

```
year = 1;
while (year <= 20)
{
    balance = balance * (1 + RATE / 100);
}
```



The variable **year** is not updated in the loop body!

Another Programmer Error

What is the output?

```
i = 5;
while (i < 0)
{
    cout << i << " ";
    i--;
}
```

A Very Difficult Error to Find (especially after looking for hours and hours!)

What is the output?

```
i = 5;
while (i < 0)
{
    cout << i << " ";
    i--;
}
```

do loop

The `do { } while ()` Loop

- The `while()` loop's condition test is the first thing that occurs in its execution.
- The `do` loop (or `do-while` loop) has its condition tested only after at least one execution of the statements. The test is at the bottom of the loop:

```
do
{
    statements
}
while (condition);
```

The do Loop

- This means that the do loop should be used only when the statements must be executed before there is any knowledge of the condition.
- This also means that the do loop is the least used loop.

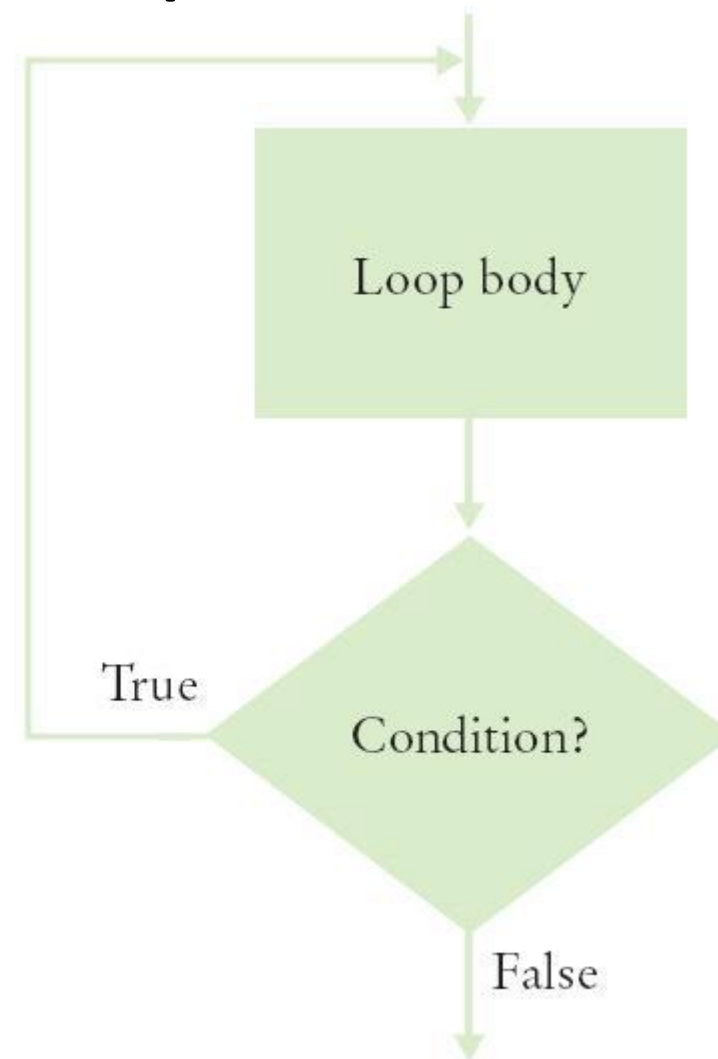
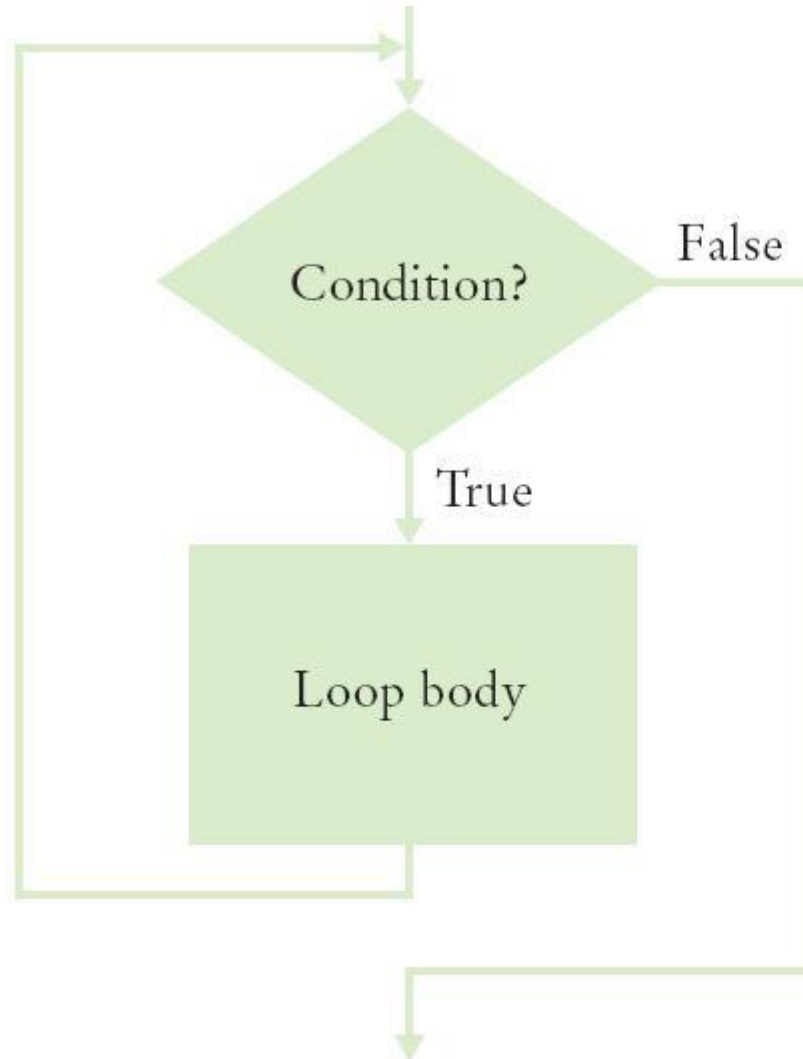
do { } Loop Code: getting user input Repeatedly

- Code to keep asking a user for input until it satisfies a condition, such as non-negative for applying the `sqrt()`:

```
double value;
do
{
    cout << "Enter a number >= 0: ";
    cin >> value;
}
while (value < 0);

cout << "The square root is " << sqrt(value) << endl;
```

Flowcharts for the `while` Loop and the `do` Loop



Practice It: Example of do..while

- What output does this loop generate?

```
int j = 1;
do
{
    int value = j * 2;
    j++;
    cout << value << ", ";
} while (j <= 5);
```

How to Write a Loop

These are the steps to follow when turning a problem description into a code loop:

1. Decide what work must be done inside the loop
 - *For example, read another item or update a total*
2. Specify the loop condition
 - *Such as exhausting a count or invalid input*
3. Determine the loop type
 - *Use for in counting loops, while for event-controlled*
4. Set up variables for entering the loop for the first time
5. Process the result after the loop has finished
6. Trace the loop with typical examples
7. Implement the loop in C++

for vs. while loop

The `for` Loop vs. the `while` loop

- Often you will need to execute a sequence of statements a given number of times.

You could use a `while` loop:

```
counter = 1; // Initialize the counter
while (counter <= 10) // Check the counter
{
    cout << counter << endl;
    counter++; // Update the counter
}
```

The `for` Loop vs. the `while` loop

- Often you will need to execute a sequence of statements a given number of times.

You could use a `while` loop:

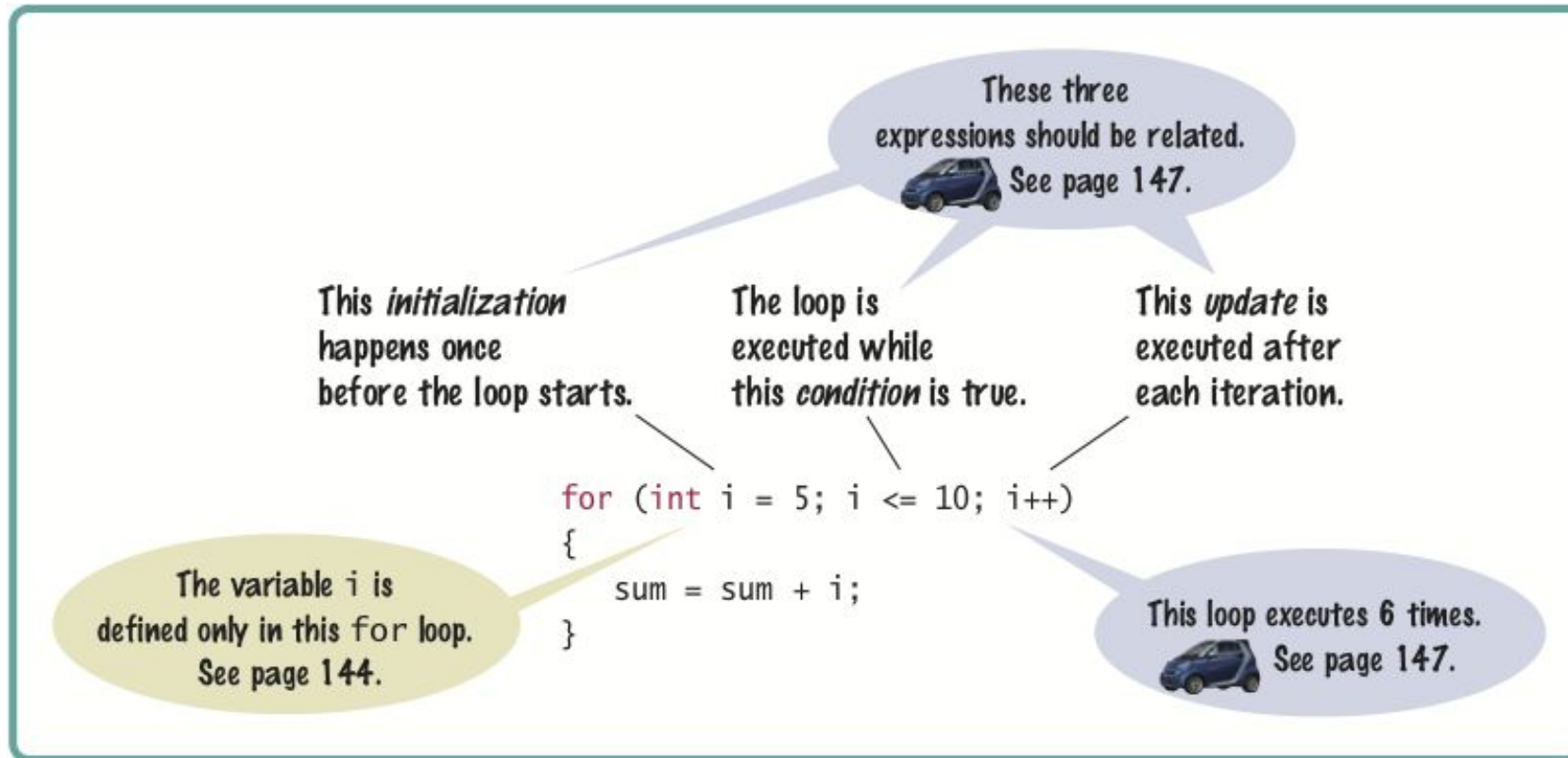
```
counter = 1; // Initialize the counter
while (counter <= 10) // Check the counter
{
    cout << counter << endl;
    counter++; // Update the counter
}
```

The `for` Loop

- C++ has a statement custom made ***for*** this sort of processing: the **for** loop.

```
for (counter = 1; counter <= 10; counter++)  
{  
    cout << counter << endl;  
}
```


The `for` Loop Syntax



The `for` Loop Is Better than `while` for Certain Things

- Doing something a known number of times or causing a variable to take on a sequence of values is so common, C++ has a statement just for that:

```
for (int count = 1; count <= 10; count++)  
{  
    cout << count << endl;  
}
```

The diagram illustrates the four components of a C++ `for` loop. Four blue arrows point from labels at the bottom to specific parts of the code:
- initialization points to `int count = 1`
- condition points to `count <= 10`
- statements points to `cout << count << endl;`
- update points to `count++`
A black arrow points from the `count` variable in the condition back to the `count` variable in the update, indicating the loop's progression.

for () loop execution

```
for (initialization; condition; update)
{
    statements;
}
```

- The **initialization** is code that happens once, before the check is made, to set up counting how many times the *statements* will happen. The loop variable may be created here, or before the `for ()` statement.
- The **condition** is a comparison to test if the loop is done. When this test is false, we skip out of the `for ()`, going on to the next statement.
- The **update** is code that is executed at the bottom of each iteration of the loop, immediate before re-testing the condition. Usually it is a counter increment or decrement.
- The **statements** are repeatedly executed until the condition is false. These also are known as the "loop body".

The `for` Can Count Up or Down

- A `for` loop can count down instead of up:

```
for (counter = 10; counter >= 0; counter--)
```

- The increment or decrement need not be in steps of 1:

```
for (cntr = 0; cntr <= 10; cntr +=2)
```

- Notice that in these examples, the loop variable is defined **in** the *initialization* (where it really should be!).

for Loop Examples, Index Values		
Loop	Values of i	Comment
for (i = 0; i <= 5; i++)	0 1 2 3 4 5	Note that the loop is executed 6 times. (See Programming Tip 4.3)
for (i = 5; i >= 0; i--)	5 4 3 2 1 0	Use i-- for decreasing values.
for (i = 0; i < 9; i = i + 2)	0 2 4 6 8	Use i = i + 2 for a step size of 2.
for (i = 0; i != 9; i += 2)	0 2 4 6 8 10 ... (infinite loop)	You can use < or <= instead of != to avoid this problem.
for (i = 1; i <= 20; i = i * 2)	1 2 4 8 16	You can specify any rule for modifying i, such as doubling it in every step.
for (i = 0; i < str.length(); i++)	0 1 2 ... until the last valid index of the string str	In the loop body, use the expression str.substr(i, 1) to get a string containing the ith character.

Converting from a *while* loop to a *for* loop

```
int i = 0;
while (i < 5)
{
    cout << i << " ";
    i++;
}
```

initialize loop variable *i*:
ONLY ONCE!

```
for (int i = 0; i < 5; i++)
{
    cout << i << " ";
}
```

Converting from a *while* loop to a *for* loop

```
int i = 0;
```

```
while (i < 5)
```

```
{
```

```
    cout << i << " ";
```

```
    i++;
```

```
}
```

loop condition



```
for (int i = 0; i < 5; i++)
```

```
{
```

```
    cout << i << " ";
```

```
}
```

Converting from a *while* loop to a *for* loop

```
int i = 0;
while (i < 5)
{
    cout << i << " ";
    i++;
}
```

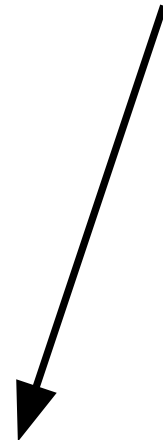
update loop
variable *i*

```
for (int i = 0; i < 5; i++)
{
    cout << i << " ";
}
```


Converting from a *while* loop to a *for* loop

```
int i = 0;  
while (i < 5)  
{  
    cout << i << " ";  
    i++;  
}
```

cout << i << " ";
i++;



loop body

```
for (int i = 0; i < 5; i++)  
{  
    cout << i << " ";  
}
```

cout << i << " ";

Converting from a *while* loop to a *for* loop

