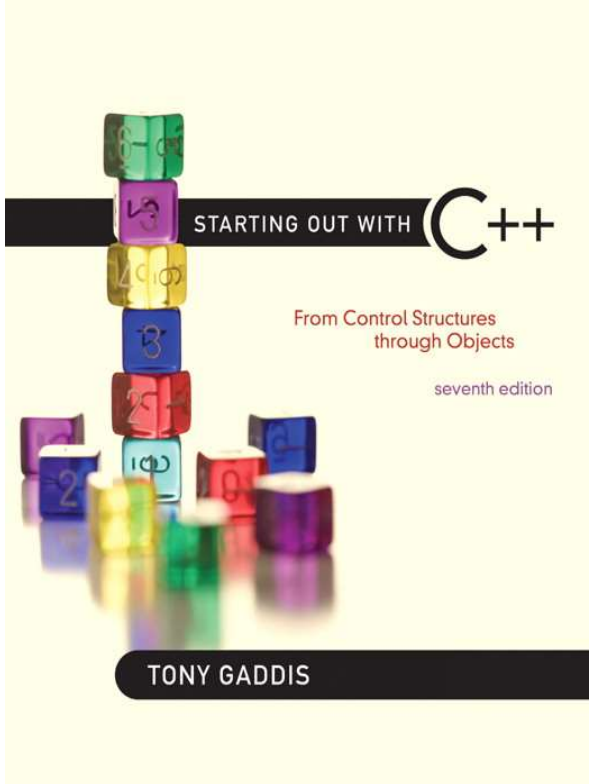
18 - 10 - 2022



The `For` Loop

The `for` Loop

- Useful for counter-controlled loop
- General Format:

```
for(initialization; test; update)
    statement; // or block in { }
```

- No semicolon after the `update` expression or after the `)`

`for` Loop - Mechanics

```
for(initialization; test; update)
          statement; // or block in { }
```

1) Perform *`initialization`*
2) Evaluate *`test`* expression
   - If `true`, execute *`statement`*
   - If `false`, terminate loop execution
3) Execute *`update`*, then re-evaluate *`test`* expression

`for` Loop - Example

```
int count;

for (count = 1; count <= 5; count++)
   cout << "Hello" << endl;
```

A Closer Look at the Previous Example

Step 1: Perform the initialization expression.

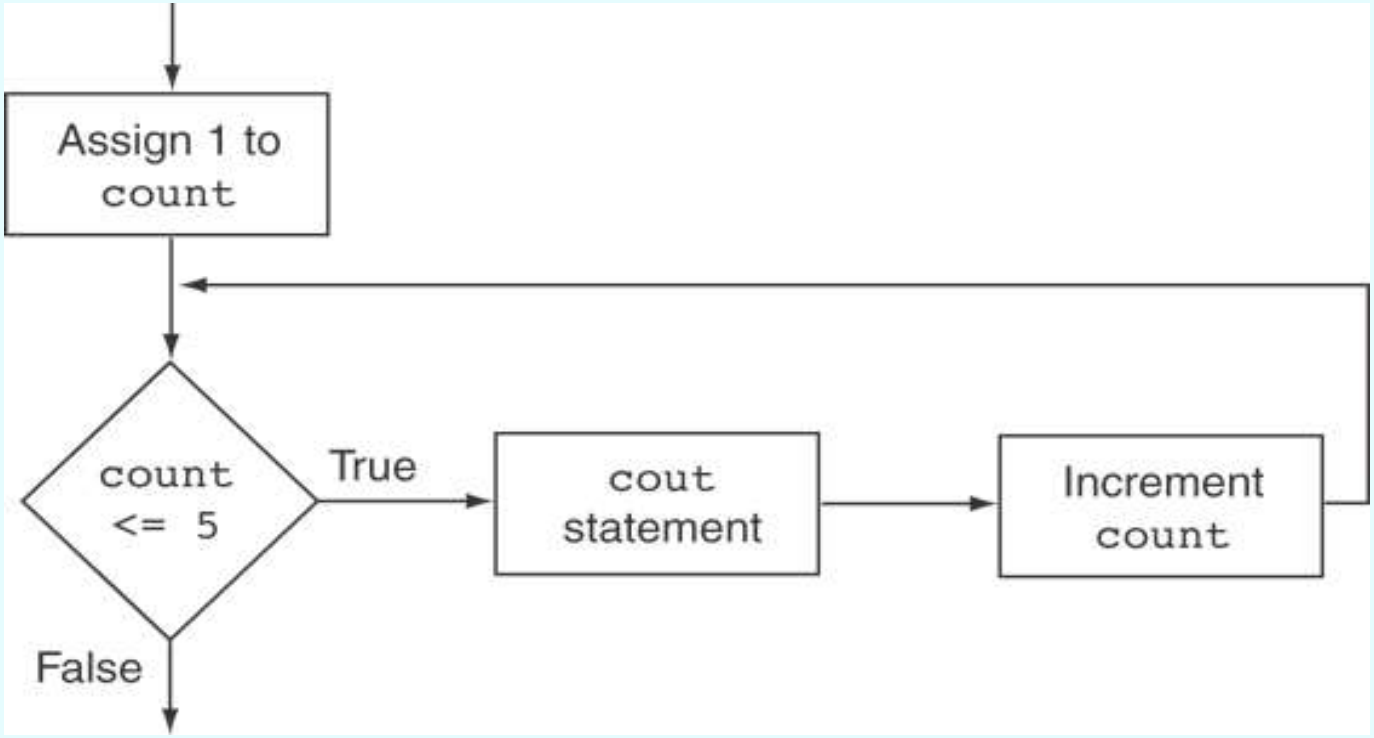
Step 2: Evaluate the test expression. If it is true, go to Step 3. Otherwise, terminate the loop.

```
for (count = 1; count <= 5; count++)
   cout << "Hello" << endl;
```

Step 3: Execute the body of the loop.

Step 4: Perform the update expression, then go back to Step 2.

Flowchart for the Previous Example



A `for` Loop in Program 5-9

Program 5-9

```
// This program displays the numbers 1 through 10 and
// their squares.
#include <iostream>
using namespace std;

int main()
{
   const int MIN_NUMBER = 1,    // Starting value
             MAX_NUMBER = 10;   // Ending value
   int num;

   cout << "Number Number Squared\n";
   cout << "-------------------------\n";

   for (num = MIN_NUMBER; num <= MAX_NUMBER; num++)
      cout << num << "\t\t" << (num * num) << endl;

   return 0;
}
```

Continued…

A `for` Loop in Program 5-9

Program Output

```
Number Number Squared
---------------------
1            1
2            4
3            9
4            16
5            25
6            36
7            49
8            64
9            81
10           100
```

A Closer Look at Lines 15 through 16 in Program 5-9

Step 1: Perform the initialization expression.

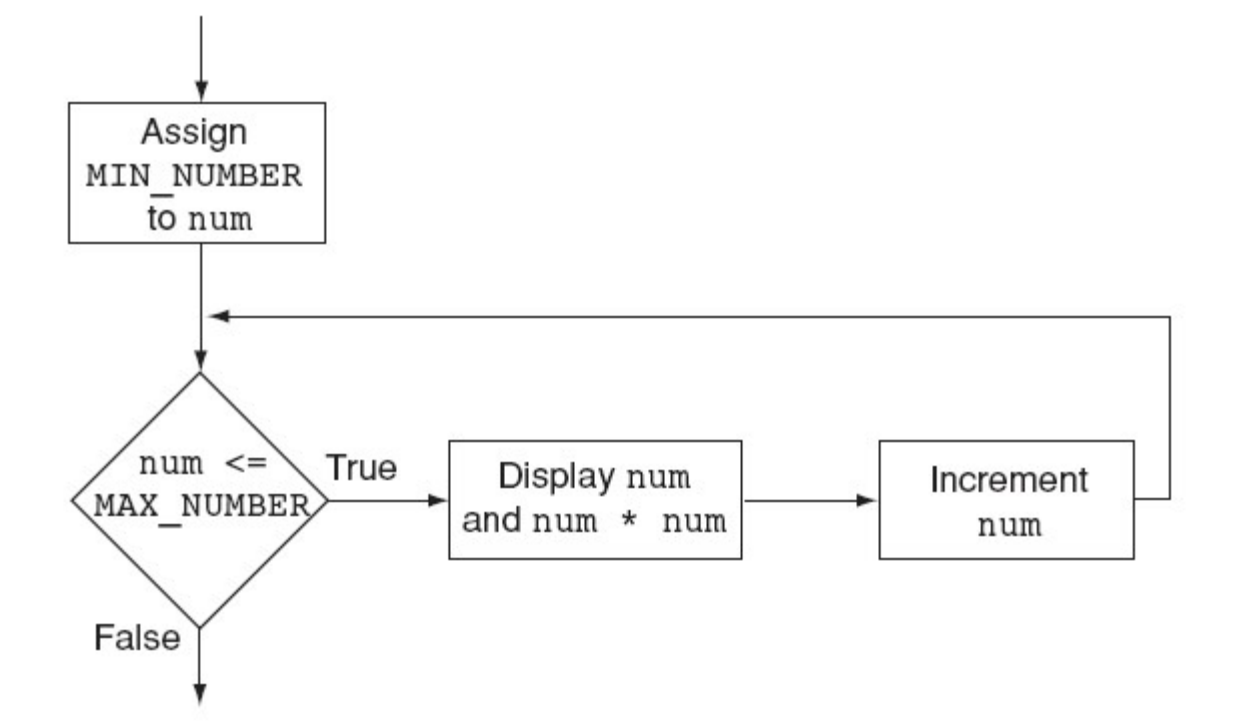
Step 2: Evaluate the test expression. If it is true, go to Step 3. Otherwise, terminate the loop.

Step 4: Perform the update expression, then go back to Step 2.

```
for (num = MIN_NUMBER; num <= MAX_NUMBER; num++)
   cout << num << "\t\t" << (num * num) << endl;
```

Step 3: Execute the body of the loop.

Flowchart for Lines 15 through 16 in Program 5-9



When to Use the `for` Loop

- In any situation that clearly requires
  - an initialization
  - a false condition to stop the loop
  - an update to occur at the end of each iteration

The `for` Loop is a Pretest Loop

- The for loop tests its test expression before each iteration, so it is a pretest loop.
- The following loop will never iterate:

```
for (count = 11; count <= 10; count++)
   cout << "Hello" << endl;
```

`for` Loop - Modifications

- You can have multiple statements in the *`initialization`* expression. Separate the statements with a comma:

Initialization Expression

```
int x, y;
for (x=1, y=1; x <= 5; x++)
{
    cout << x << " plus " << y
         << " equals " << (x+y)
         << endl;
}
```

`for` Loop - Modifications

- You can also have multiple statements in the *`update`* expression. Separate the statements with a comma:

Update Expression

```
int x, y;
for (x=1, y=1; x <= 5; x++, y++)
{
    cout << x << " plus " << y
         << " equals " << (x+y)
         << endl;
}
```

`for` Loop - Modifications

- You can omit the *`initialization`* expression if it has already been done:

```
int sum = 0, num = 1;
for (; num <= 10; num++)
    sum += num;
```

`for` Loop - Modifications

- You can declare variables in the *`initialization`* expression:

```
int sum = 0;
for (int num = 0; num <= 10;
num++)
      sum += num;
```

The scope of the variable `num` is the `for` loop.