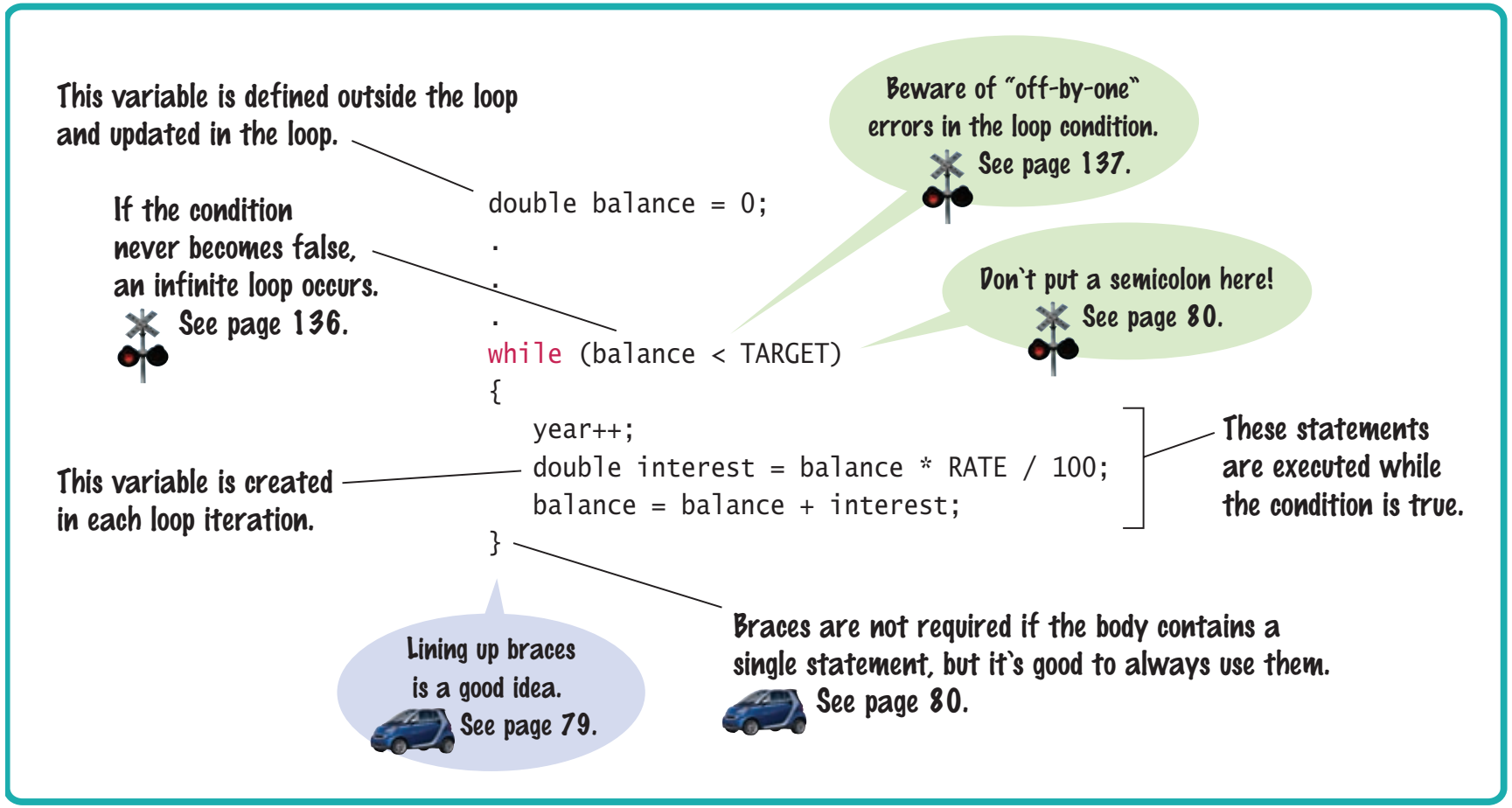
Loops

`while` loop

The *while* Loop Syntax



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 semicolon 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

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

What is the output?

```
..
```

Example of a Problem – An Infinite Loop

The output never ends

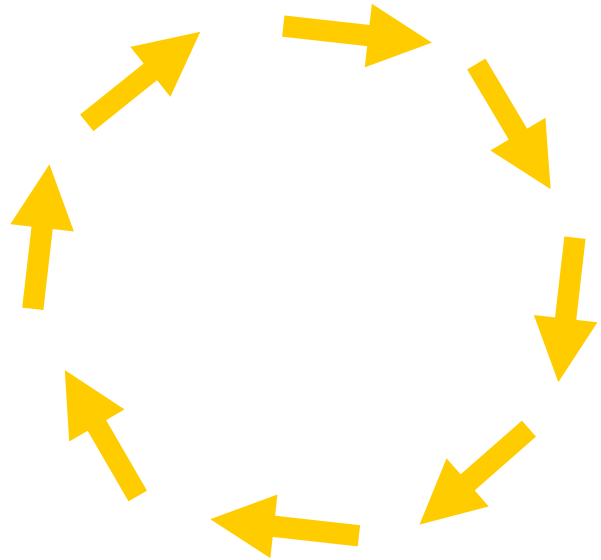
- *i* is set to 5
- The *i++;* statement makes *i* get bigger and bigger
- the condition will never become false –
- an infinite loop

```
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.

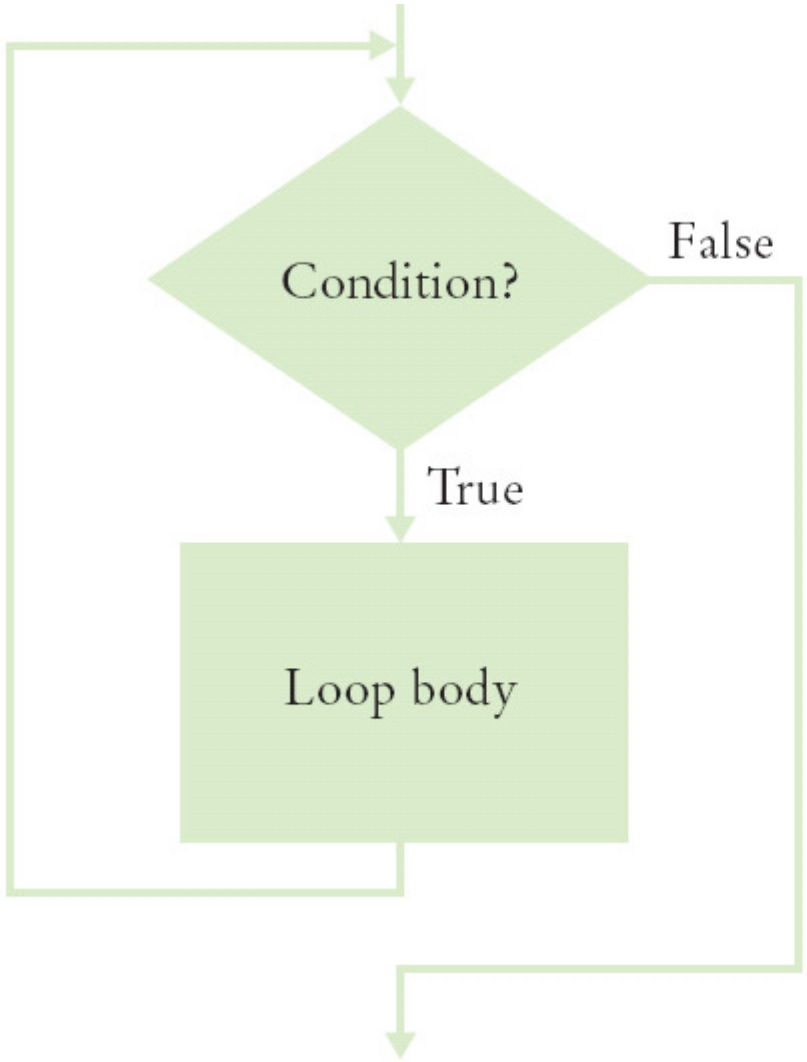
do { } Loop Code: getting user input Repeatedly

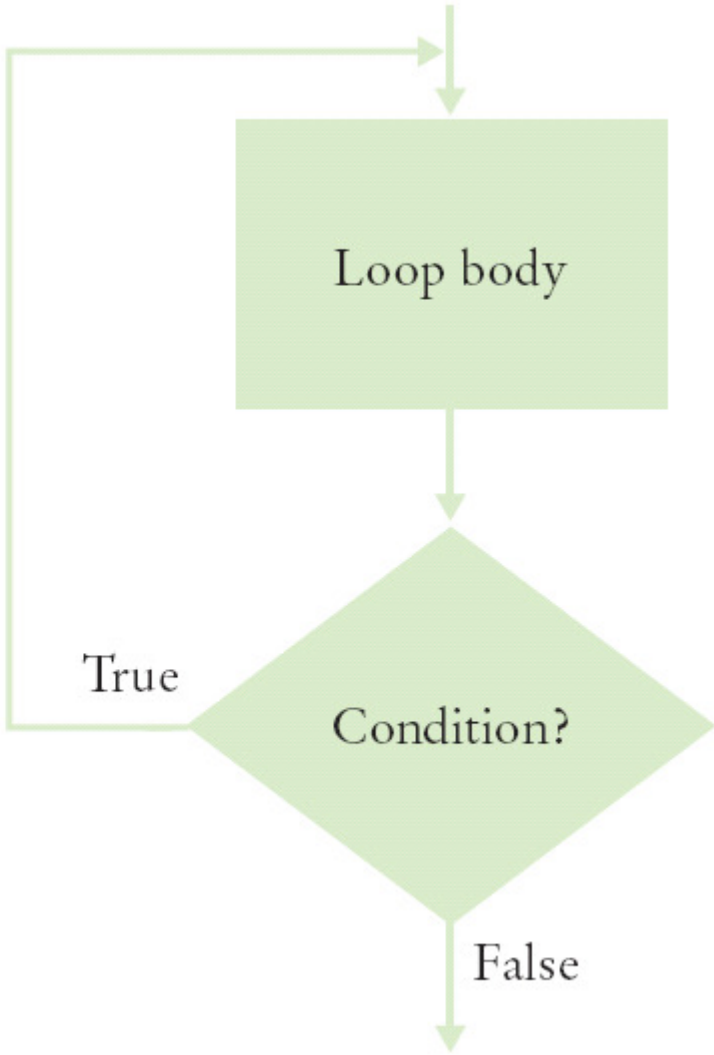
- Code to keep asking a user for input until it satisfies a condition, such as non-negative for applying the sqrt():

```
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);
```

`for` 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:

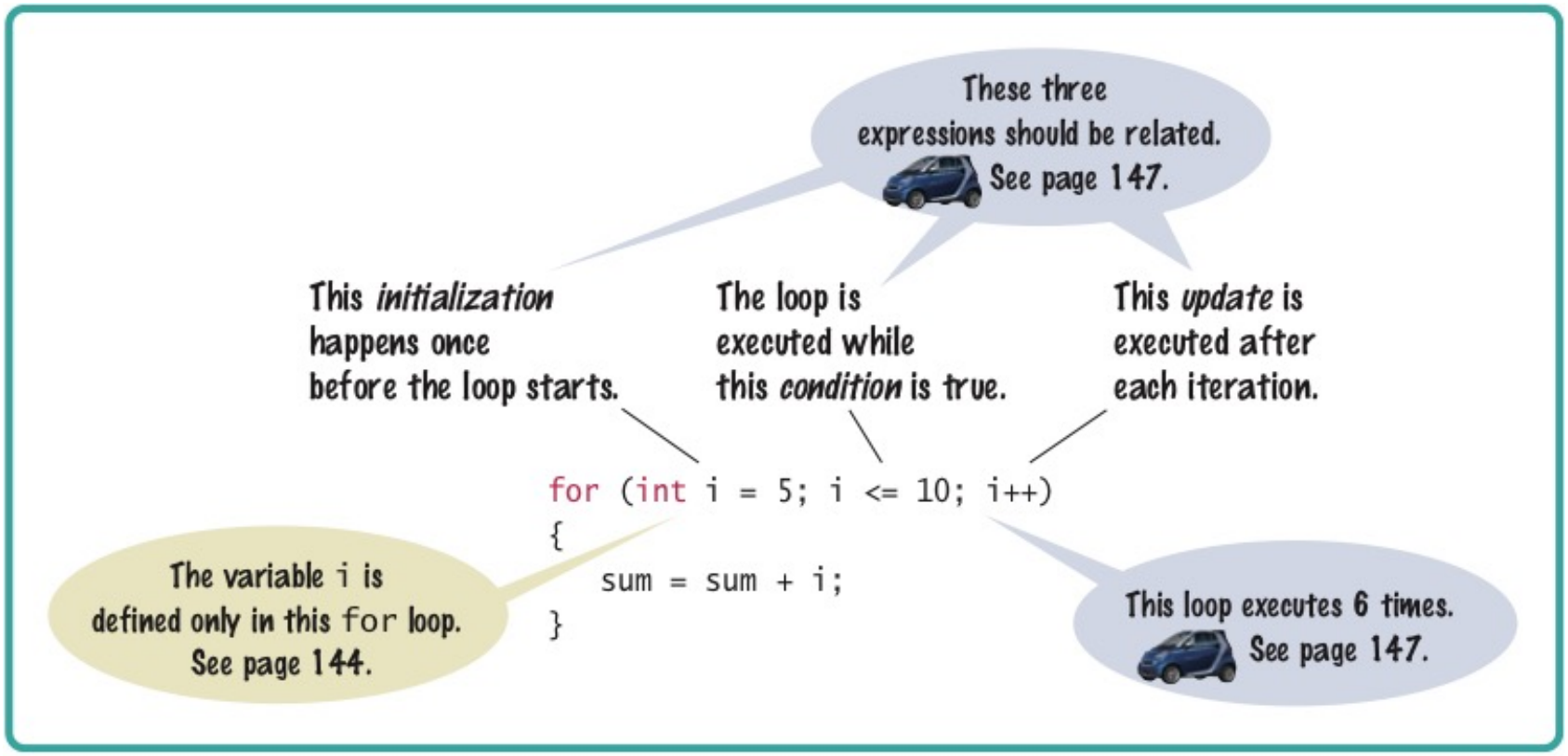
```
num = 1; // Initialize the variable
while (num <= 10) // Check the variable
{
   cout << num << endl;
   num++; // Update the variable
}
```

The `for` Loop

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

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

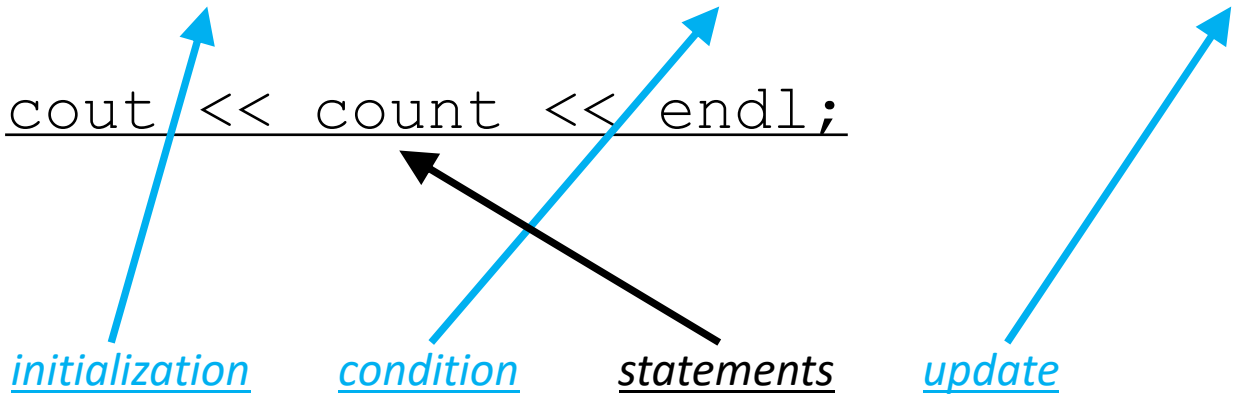
The `for` Loop Syntax



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;
}
```



`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".

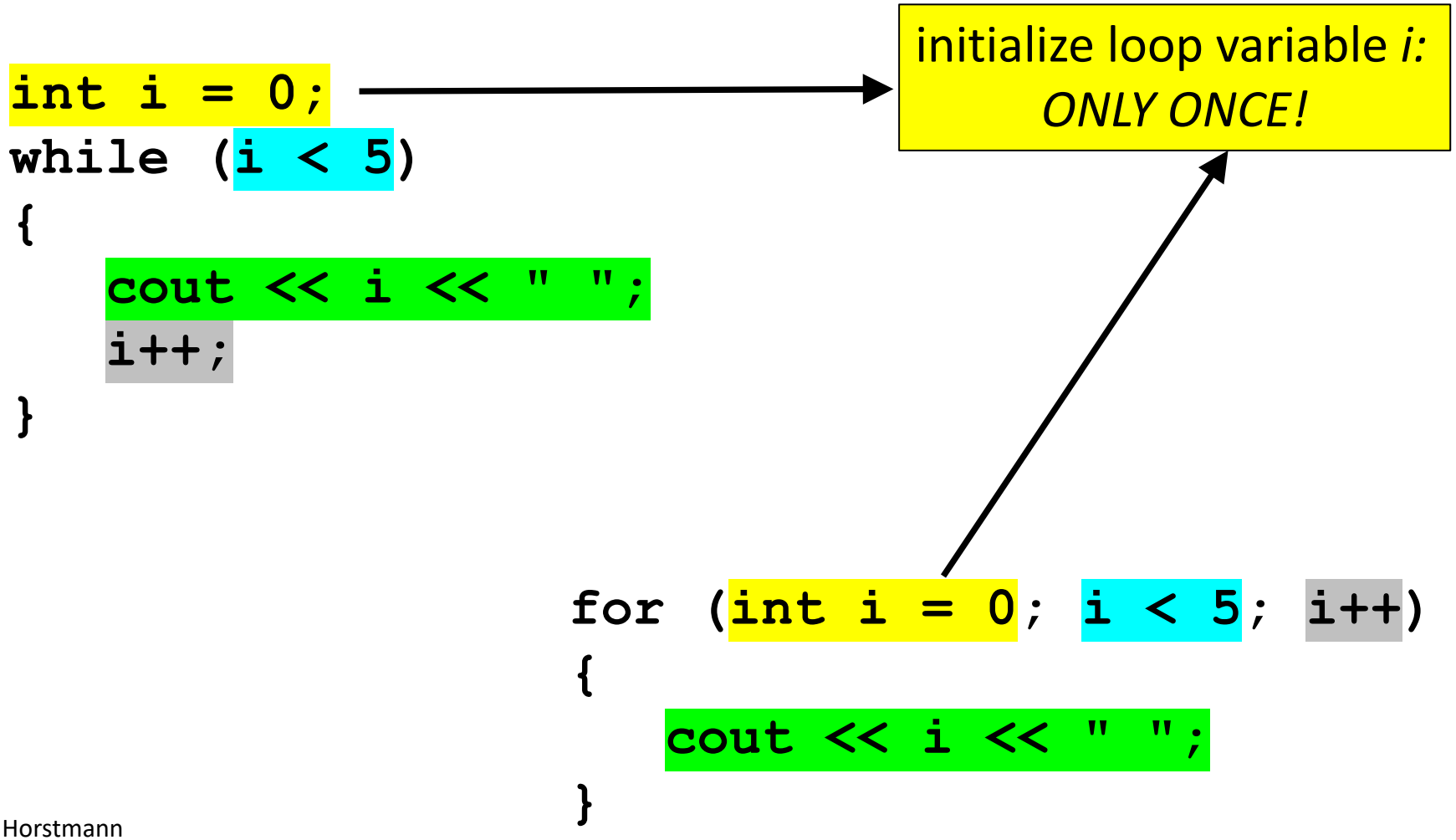
The `for` Can Count Up or Down

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

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

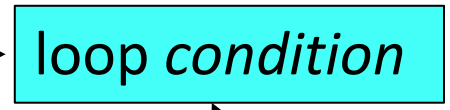
- Notice that in this examples, the loop variable is defined **in** the *initialization* (where it really should be!).

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



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

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




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

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

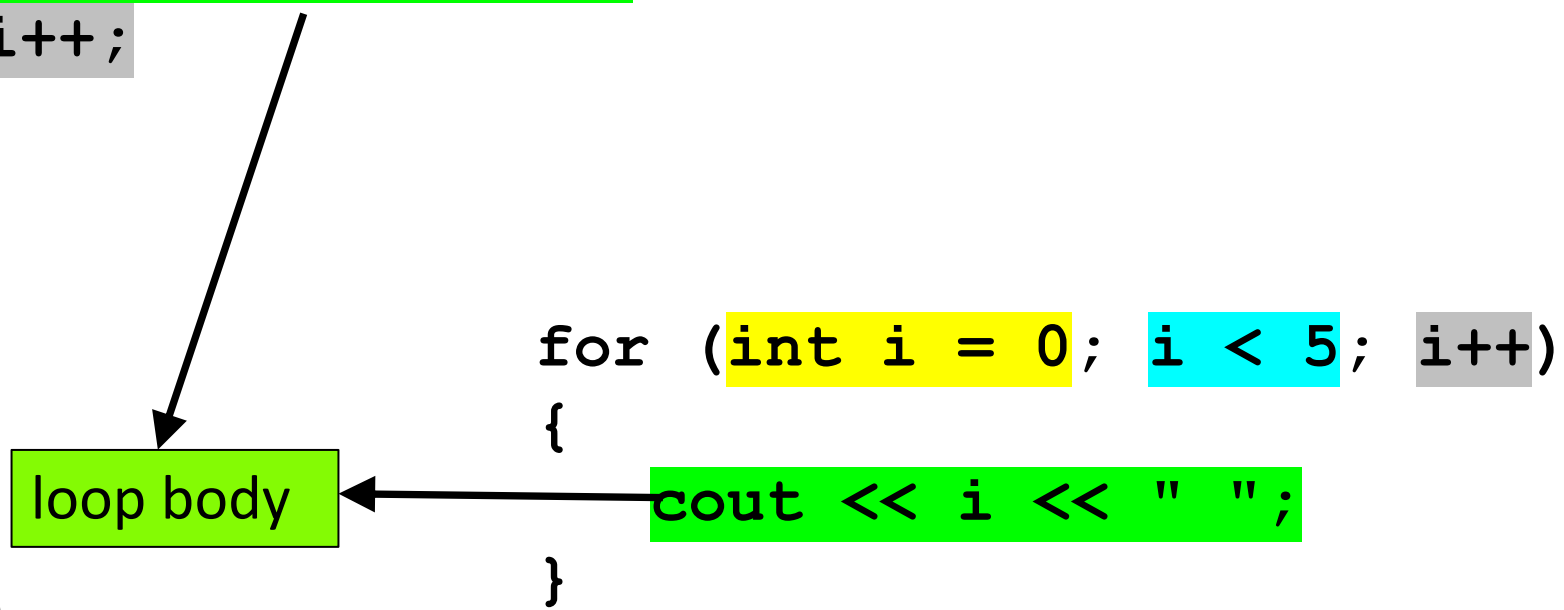
```
int i = 0;
while (i < 5)
{
   cout << i << " ";
   i++;
}
```

update loop variable *i*

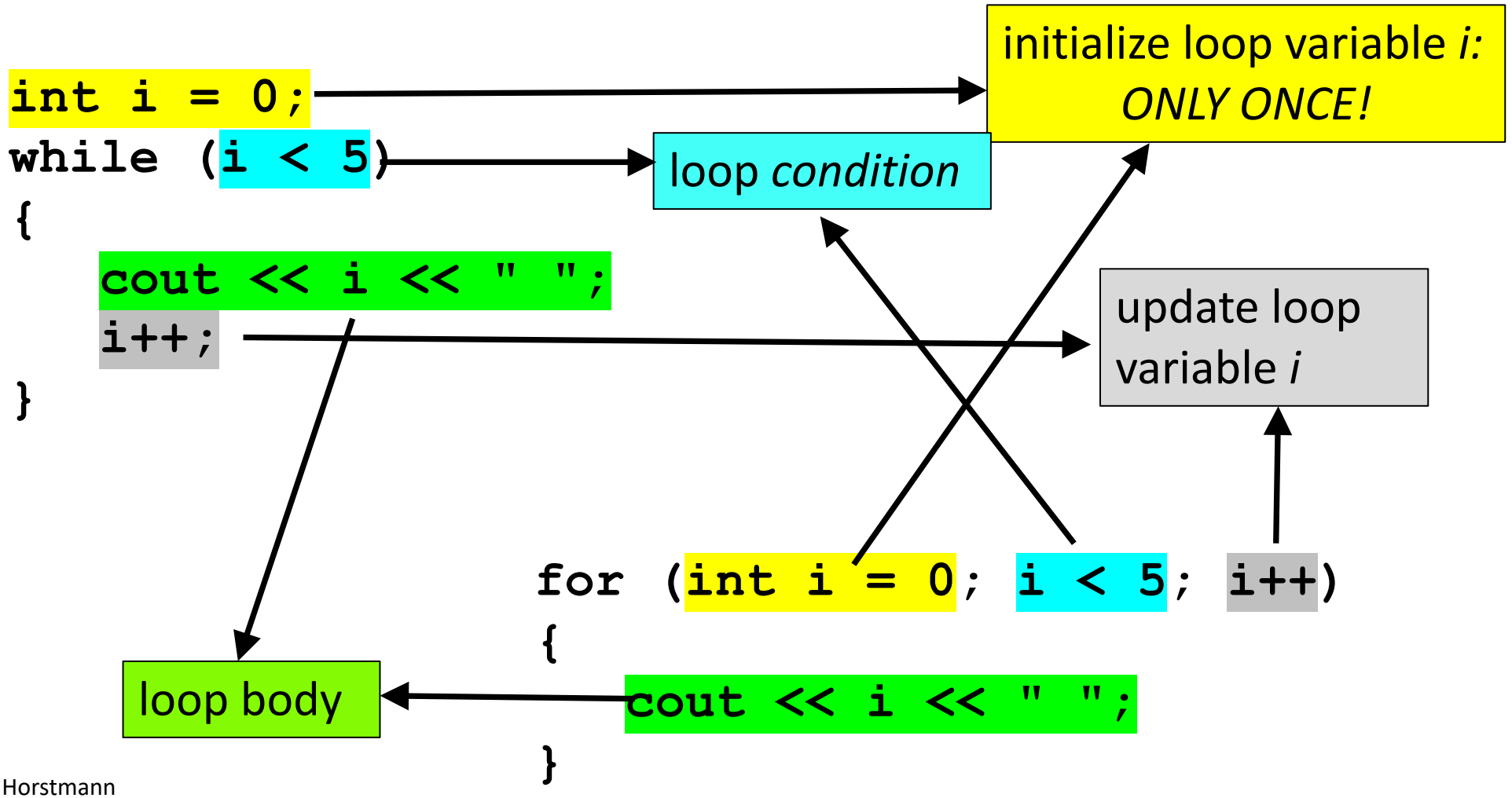
```
for (int i = 0; i < 5; i++)
{
   cout << i << " ";
}
```

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

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



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



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++