Loops

Today

- while loop
- do while loop
- for loop

Due this week

- **Homework 4**
  - Write solutions in VSCode and paste in Autograder, **Homework 4 CodeRunner**.
  - Check the due date! **No late submissions!!**
- Start going through the textbook readings and watch the videos
  - Take **Quiz 4**.
  - Check the due date! **No late submissions!!**

Loops

Let’s dive in

- Loop is a part of your program that is repeated over and over until you tell it to stop!
- A loop has 3 parts (you need to know about)
  - Variable Initialization
  - A condition (an expression that must evaluate to True or False)
  - Loop Body
- So, loop helps you run the code in the **loop body** as long as the **condition** is true. *You’re right - the program stops executing the loop body when condition becomes false.*
- What’s this **variable initialization** then?? - Well, you need variables to put inside the loop body, don’t you? AND also for that **condition!**

The *while* Loop Syntax

```
// 1. While Loops

// Let's look at syntax

/*

a. define variables you need here!
int count = 1;

b. the loop condition
while (<condition>) {
    c. loop body
    statement 1
    statement 2
    ..... and so on
}
*/
```

Some Examples

- Print values from 1 to 10 on the terminal window. *(Don’t need to put 20 ‘cout’ statements)*

- Input validation - Take in an input from user, and accept until user enters a positive number

- Password Authentication Systems!

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.

Practice

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

A. 4 3 2 1 0
B. 5 4 3 2 1 0
C. 5 4 3 2 1
D. No Output

Practice

```
i = 11;
while (i < 20);
{
    cout << i << " ";
    i= i+2;
}
```

A. 11 12 13 14 15 16 17 18 19
B. 11 13 15 17 19
C. 11 12 13 14 15 16 17 18 19 20
D. No Output

Practice

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

A. 5 4 3 2 1
B. 5 4 3 2 1 0
C. 1 2 3 4 5
D. No 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

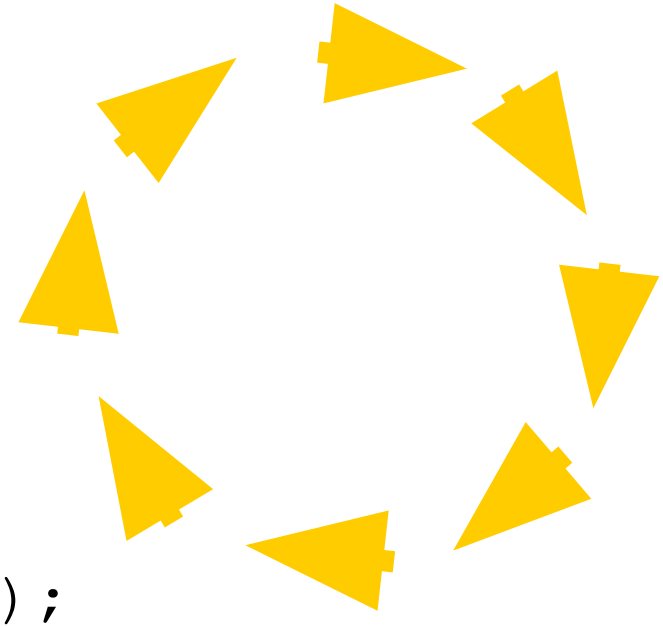
```
i = 5;
while (i > 0)
{
   cout << i << " ";
   i++;
}
```

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);
}
```



Why is the above an infinite loop?

Common Error – Infinite Loops

So remember, after you write the while loop, check if it's going to be an infinite loop.

- You probably missed an update statement
- You probably put a wrong update statement

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.

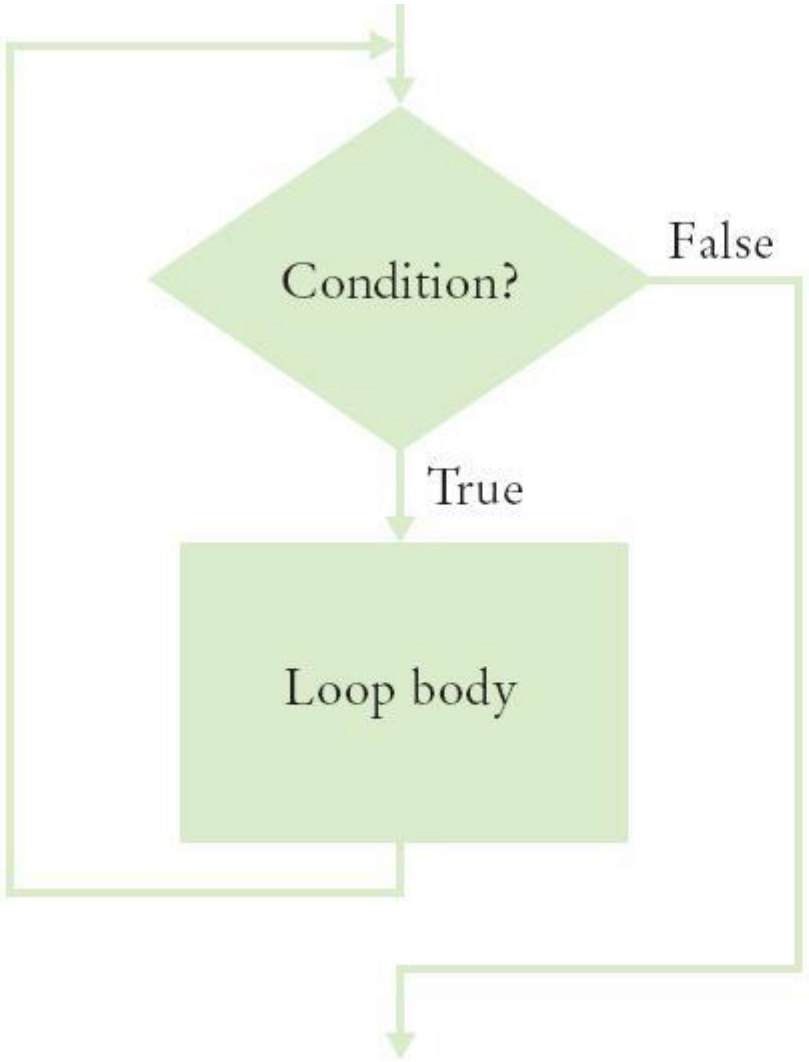
do { } Loop Code: getting user input Repeatedly

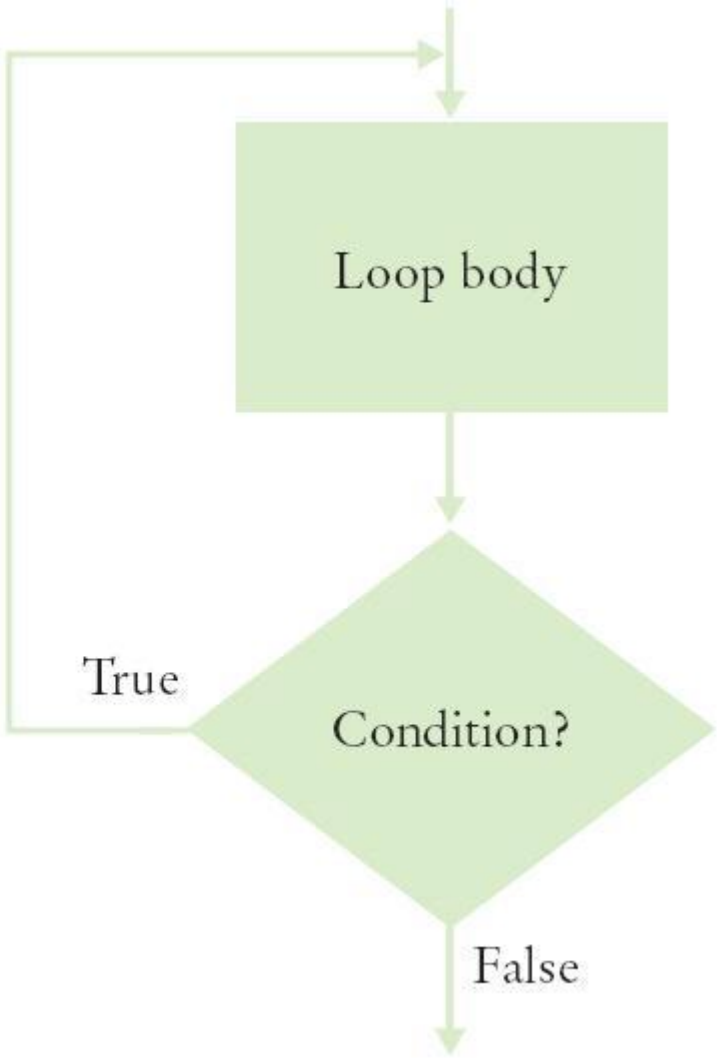
- Code to keep asking a user for input until it satisfies a condition, such as non-negative for applying the sqrt():

```
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++

Today

- Floating Point Comparisons (is 3.0 the same as 3, hmm..)
- Revisit Pre and Post Increment/Decrement Operators
- Examples
- for loops

Floating Point Comparisons

Practice

```
float a = 1.1, b = 2.2;
float sum = a + b;
if (sum == 3.3) {
    cout << “Equal”;
}
else {
    cout << “Not equal”;
}
```

A. Equal
B. Not equal
C. No output

Practice

```
float a = 1.1, b = 2.2;
float sum = a + b;
if (sum == 3.3) {
    cout << “Equal”;
}
else {
    cout << “Not equal”;
}
```

A. Equal
B. Not equal
C. No output

Why does this happen?

- Because of rounding errors when they are stored in memory
- Precisely storing these numbers is very difficult, so the system approximately stores it for us.
- Hence, when you compare 2 floating point numbers, always assume they are within a small tolerance interval (0.00001 for instance).

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:

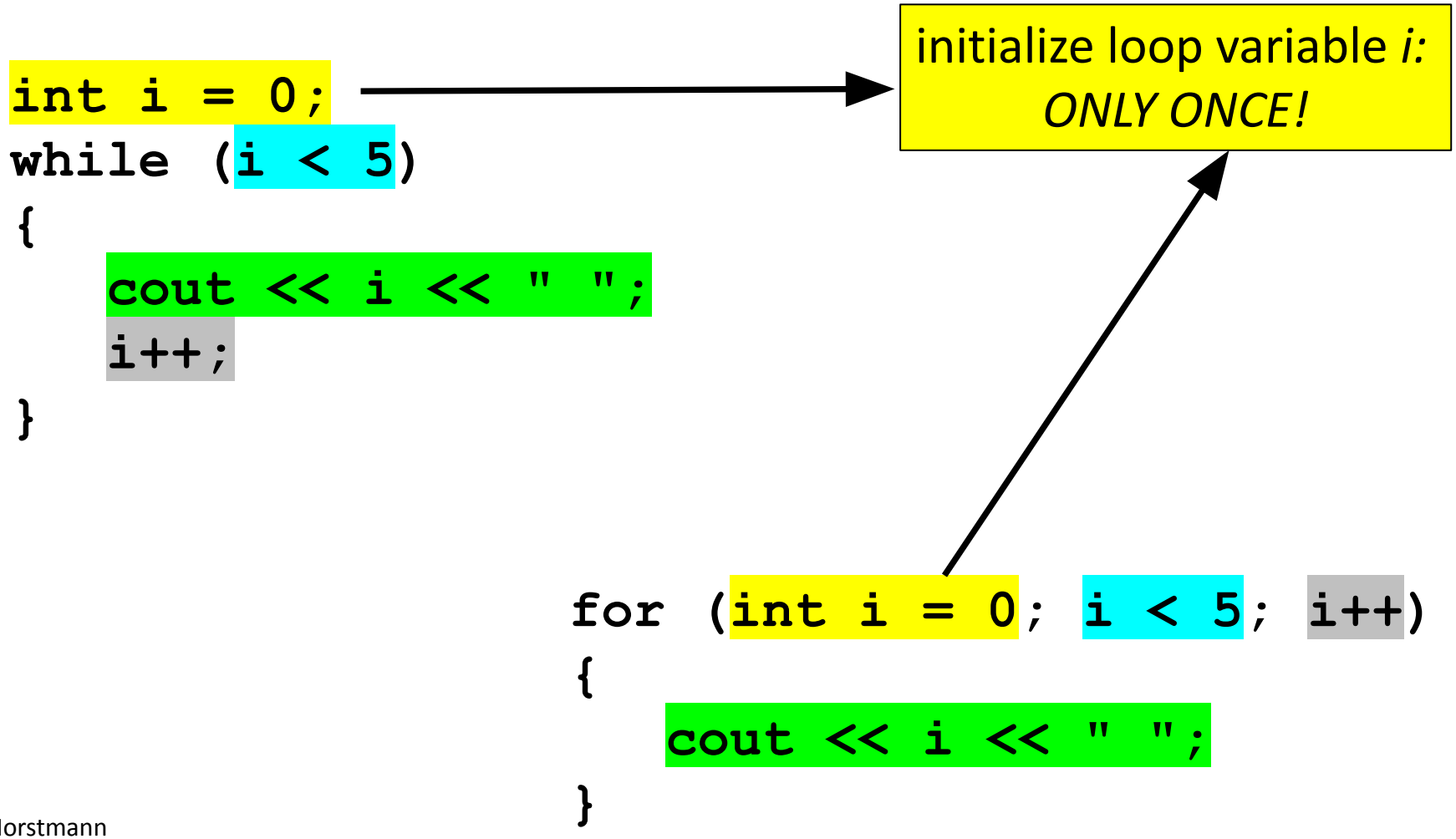
```
counter = 1; // Initialize the counter
while (counter <= 10) // The condition to check
{
   cout << counter << endl; // Loop body
   counter++; // Update the counter
}
```

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;
}
```

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

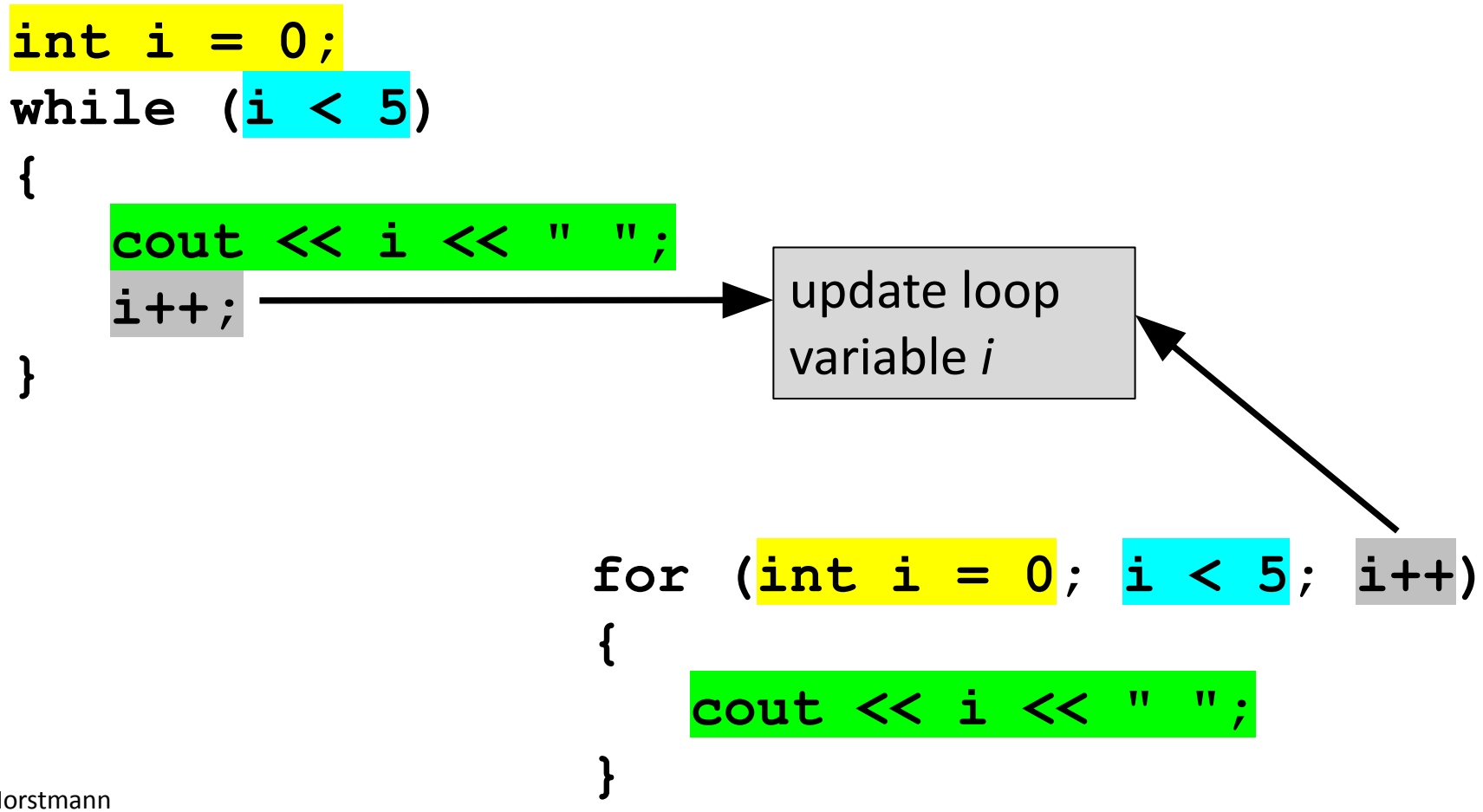


Converting from a *while* loop to a *for* loop

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

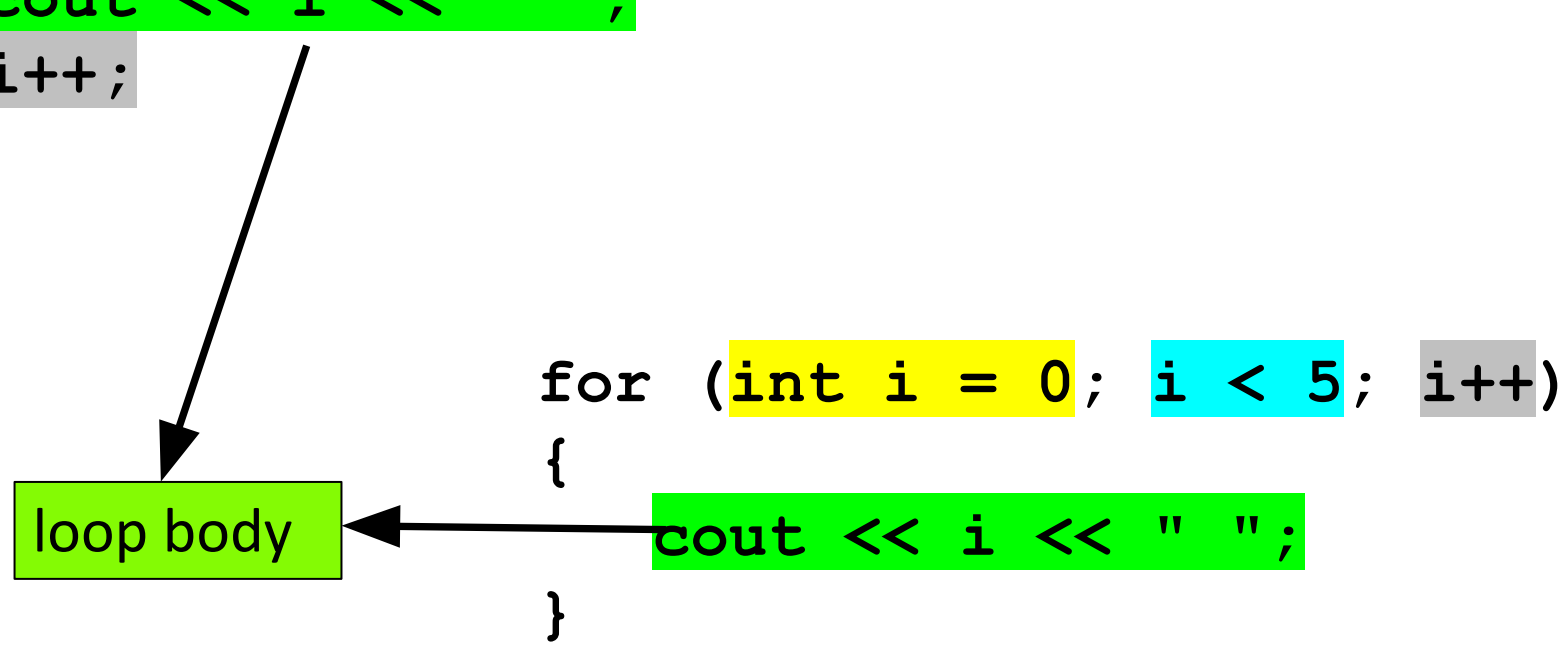

```
for (int i = 0; i < 5; i++)
{
   cout << i << " ";
}
```

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

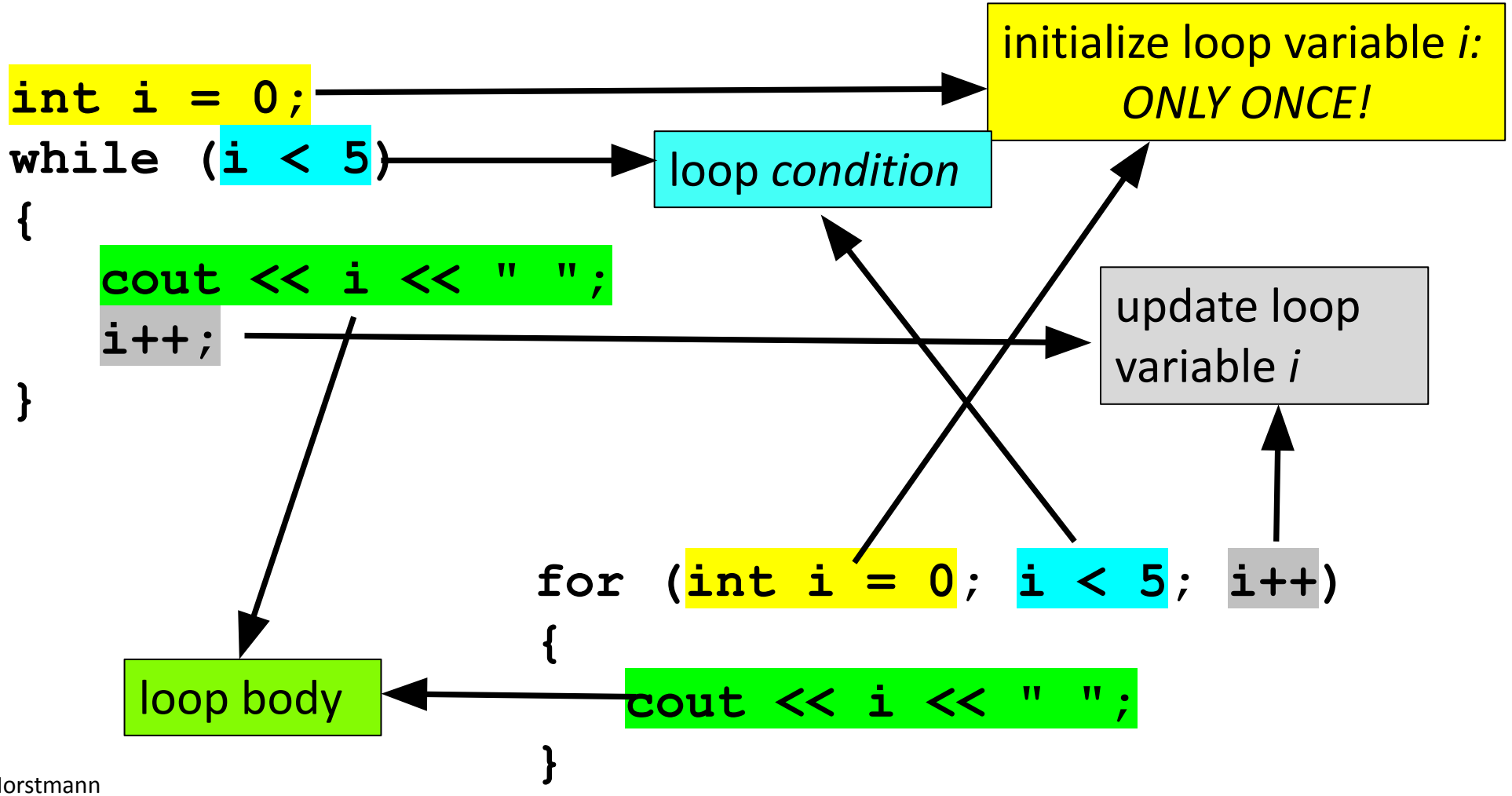


Converting from a *while* loop to a *for* loop

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

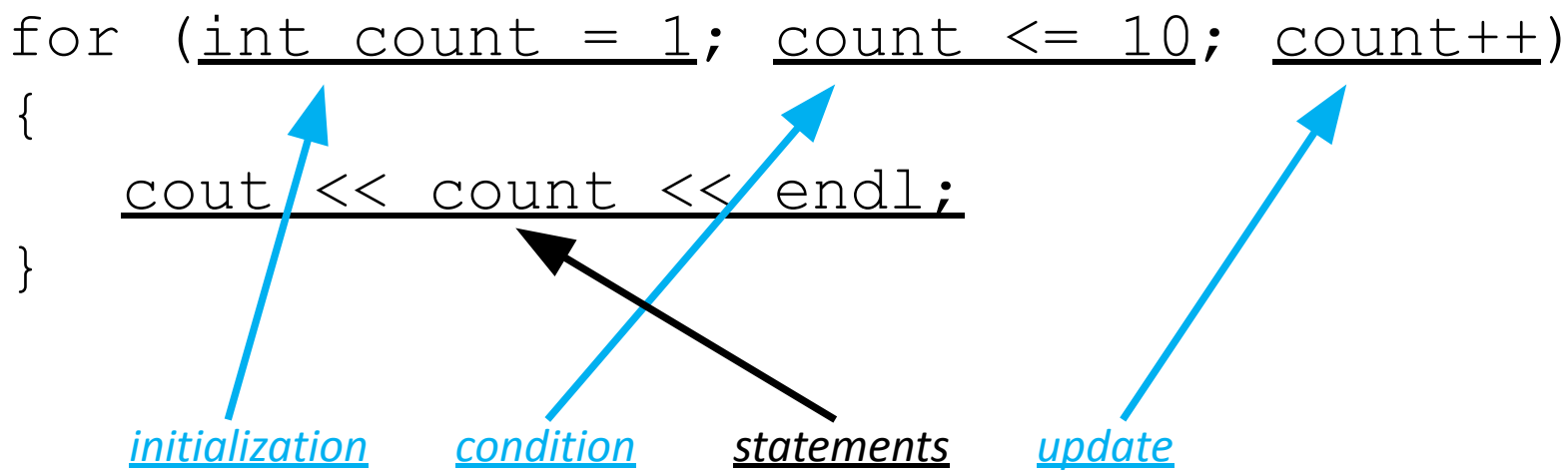


Converting from a *while* loop to a *for* loop



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:



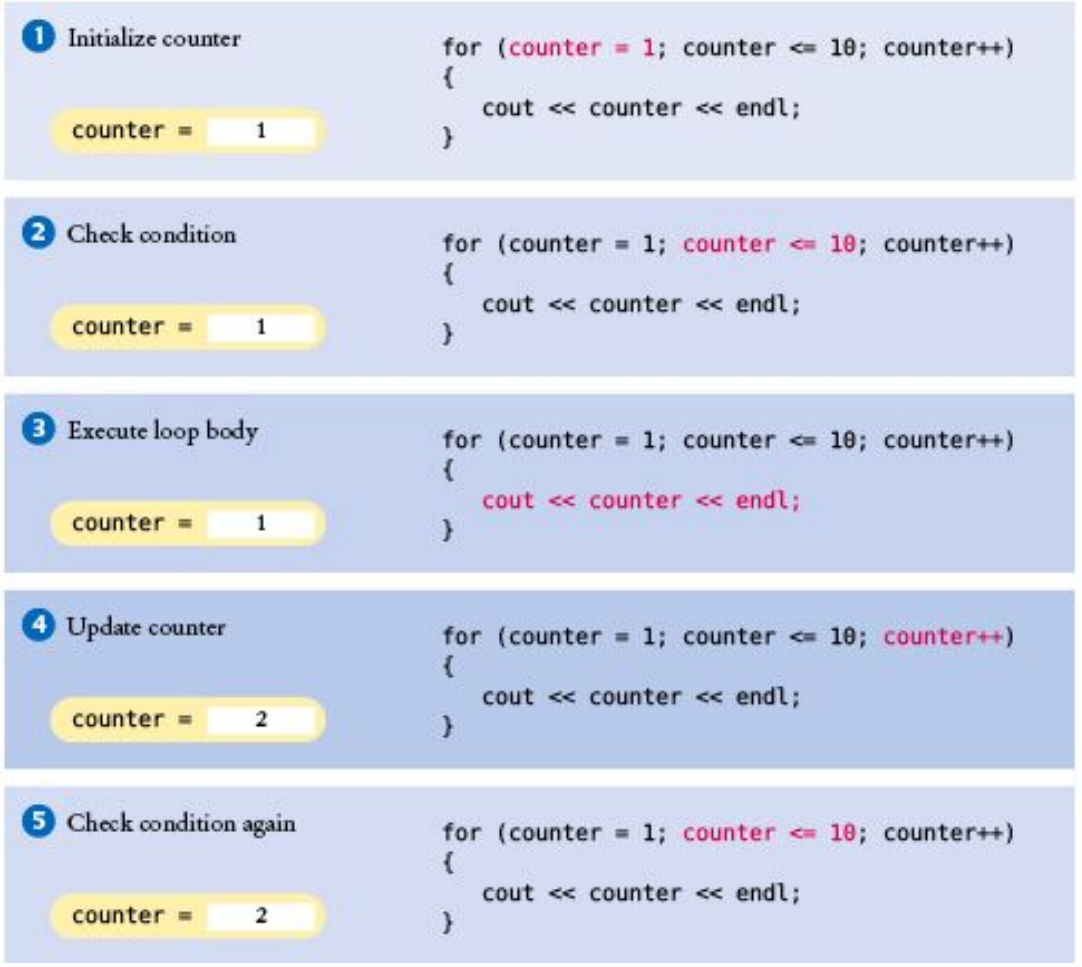
Let’s unfold a for loop

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

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

`for()` loop execution

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



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!).

Practice

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

A. 0 -1 -2 -3 -4 -5
B. 0 -1 -2 -3 -4 -5 -6
C. 0 -1 -2 -3 -4

Practice

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

A. 0 -2 -4
B. 0 -1 -2 -3 -4
C. -1 -3 -5

Practice

```
for (int i=1; i<10; i*=2) {
   cout << i << " ";
}
```

A. 1 2 4 8
B. 1 3 5 7 9
C. 1 2 4 6 8

Practice

```
for (int i=10; i>1; i=i/2)
{
   cout << i << " ";
}
```

A. 10 5 2.5 1.25
B. 10 5 2.5 1.25 1
C. 10 5 2 1
D. Infinite Loop

`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.

Common Errors #1

- Count down from 10 till 1

```
for (int i=10; i>0; i++) {
   cout << i << " ";
}
```

Common Errors #1 - Infinite Loop

- Count down from 10 till 1

```
                          i--
for (int i=10; i>0; i++) {
   cout << i << " ";
}
```

Ensure you are mentioning the update statement correctly. Whether to increase or decrease could be the confusing part.

Common Errors #2

- Suppose, you wish to print the first 100 numbers

```
for (int i=1; i<100; i++) {
   cout << i << " ";
}
```

Common Errors #2 - Off by one

- Suppose, you wish to print the first 100 numbers

```
                  i<=100
for (int i=1; i<100; i++) {
   cout << i << " ";
}
```

Ensure the condition is accurate. Use the < or <=, and > or >= relational operators correctly and accordingly

Common Errors #3

- Count down from 10 till 1

```
for (int i=10; i<0; i=i-1) {
   cout << i << " ";
}
```

Common Errors #3 - Loop not executing

- Count down from 10 till 1

```
                  i>0
for (int i=10; i<0; i=i-1) {
   cout << i << " ";
}
```

Ensure the condition is accurate. Check if the condition is true at the first iteration, to see if even the loop executes.

Common Errors #4

- Suppose, you wish to print the first 100 numbers

```
for (int i=1; i<100; i+1) {
   cout << i << " ";
}
```

Common Errors #4 - Incorrect update

- Suppose, you wish to print the first 100 numbers

```
                          i=i+1
for (int i=1; i<100; i+1) {
   cout << i << " ";
}
```

Ensure the update is correct. You need a statement that changes the value of the **control variable**

Application with Strings

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
T	h	e		b	i	g		b	a	n	g		t	h	e	o	r	y

```
string s = “The big bang theory”;

// print number of words and characters
```