`for` Loops

Lecture 6

Outline

1. Motivation via `while` loops
2. `for` Loops

A Common `while` Loop

We often use `while` loops to repeat a task a fixed number of times, which leads to a similar structure based on a counter variable

```
int count;
count = 1;
while ( count <= 8 )
{
    cout << count << " squared is "
         << count*count << endl;
    count++;
}
```

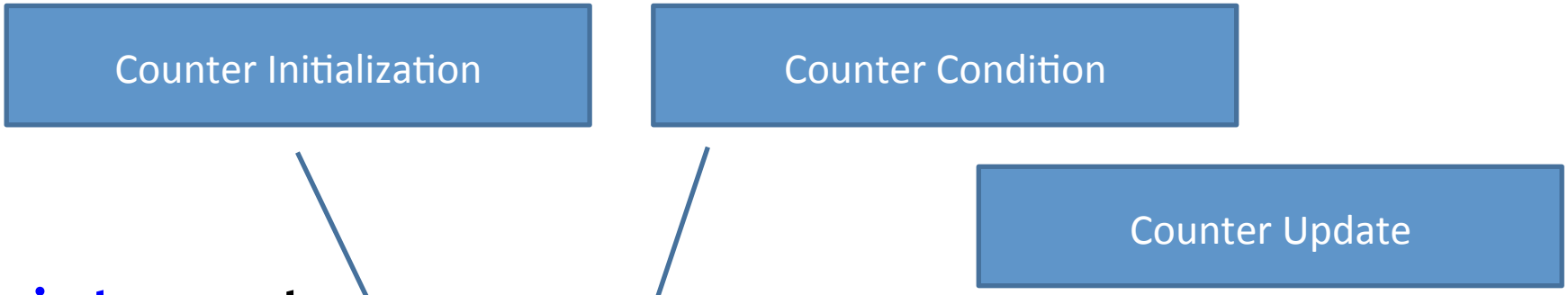
Counter Initialization

Counter Condition

Counter Update

for Loops

Specialized loops based on this counter structure




```
int count;
for ( count = 1; count <= 8; count++ )
{
   cout  << count << " squared is "
         << count*count << endl;
}
```

General Form

```
for ( INITIALIZATION; BOOLEAN_EXPRESSION; UPDATE )
{
   STATEMENT1;
   STATEMENT2;
   …
}
```

- INITIALIZATION is done one time, before the first loop iteration
- UPDATE is done every loop iteration after the first loop body statement
- BOOLEAN_EXPRESSION is checked every loop iteration, after UPDATE (and once after INITIALIZATION)

Another Example

```
#include <iostream>
#include <ctime>
using namespace std;

int main()
{
    int i;
    srand( time( 0 ) );
    for ( i=1; i<=5; i++ )
    {
        cout << rand() << endl;
    }
    return 0;
}
```

Set a "seed" for random number generator

Generate a random number

Pro Tips

- There are only two semicolons
  - Between the initialization step and the boolean expression
  - Between the boolean expression and the update step
- No semicolon after the update step
- No semicolon after the parentheses

- If you are doing an increment, be sure you do something like **`i++`**, not just **`i+1`**
  - That is, either **`i++`** or **`i=i+1`**
  - Just **`i+1`** doesn't do anything!

Exercise

Write a `for` loop that prints all the numbers between 100 and 200 (increasing order)

Answer

```
#include <iostream>
using namespace std;

int main()
{
   int number;
   for ( number=100; number<=200; number++ )
   {
      cout << number << endl;
   }
   return 0;
}
```

`for` and `while`

- Both kinds of loops work basically the same way
- The only difference is that the initialization and update pieces are part of the `for` syntax directly
- There is no particular benefit to using either loop, so you should use the one that makes the most sense to you in each situation

More Complex Updates

The update step can be more complex than a simple increment – it can be any assignment operation

- Usually updates the counter variable
- Watch out for infinite loops!

```
int x;
for ( x = 10; x >= 0; x-- )
{
   cout  << x << endl;
}
```

Another Example

```
double val;
double total = 0;

for ( val = 1; val <= 1000; val = val * 10 )
{
    total = total + val;
}
cout << total << endl;
```

	total	val
before for loop	0	?
after loop initialization	0	1
after first iteration	1	10
after second iteration	11	100
after third iteration	111	1000
after fourth iteration	1111	10000
after for loop	1111	10000

Exercise

- Write a `for` loop that prints all the powers of two between 1 and 1 billion
- Do not use the pow() function!
- Think about how to get from one power of two to the next
  - 1, 2, 4, 8, 16, 32, 64, …

Answer

```
#include <iostream>
using namespace std;

int main()
{
   int x;
   for ( x = 1; x <= 1000000000; x = x * 2 )
   {
      cout << x << endl;
   }
   return 0;
}
```

Wrap Up

- Use `for` loops when you need to repeat a task a certain number of times
- The counter/iteration variable is initialized, checked, and updated as part of the `for` loop syntax
- Always check your semicolons to be sure they are in the correct place, with no extras