Loops and Repetition

```
bool working = true;

while (working) {
    whistle();
}
```

Repetition at its Worst

Imagine that your assignment was to write a program that printed the complete "99 bottles of beer on the wall" song...

```
1   #include <cstdlib>
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6       cout << "99 bottles of beer on the wall, 99 bottles of beer!\n";
7       cout << "You take one down, pass it around; 98 bottles of beer on the wall";
8
9       cout << endl << "98 bottles of beer on the wall, 98 bottles of beer!\n";
10      cout << "Yo                                                    f beer on the wall";
11
12      cout << endl <<  97 bottles of beer on the wall, 97 bottles of beer!\n";
13      cout << "You take one down, pass it around; 96 bottles of beer on the wall";
14
15      // a whole lot more...
273
274     cout << endl << "1 bottles of beer on the wall, 1 bottles of beer!\n";
275     cout << "You take one down, pass it around; 0 bottles of beer on the wall";
276
277     system("PAUSE");
278     return 0;
279 }
280
```

15 kilobytes and 280 lines of code!

Repetition at its Best

(I don't mean the song… it still sucks)

SO much nicer to use a loop:

- only 2.5% the size of the other version!

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

int main() {
    int bottles = 99;

    while (bottles) {
        cout <<   bottles << " bottles of beer on the wall, "
             <<   bottles << " bottles of beer!\n";
        cout << "You take one down, pass it around; "
             << --bottles << " bottles of beer on the wall!\n";
    }

    system("PAUSE");
    return 0;
}

```

- loops are incredibly common in programming--and hopefully you see why!

Don’t Forget to Bring a Towel!

D.R.Y programming:

- Don’t Repeat Yourself!
- make the computer do the work!

Which would you rather do?

- Write the entire “99 bottles of beer on the wall” song, or
- write one line of it and have the computer do the rest?

Using loops lets us do more while writing less.

- Hooray for towels! Er, I mean ‘loops’...

The while loop

General syntax:

```
while (condition) {
    // code to repeat until the condition is false
}
```

Notice the 3 components:

- the while keyword, followed by
- a condition or expression inside of a pair of parentheses, and
- the block of code to repeat until the condition evaluates to false

The while loop

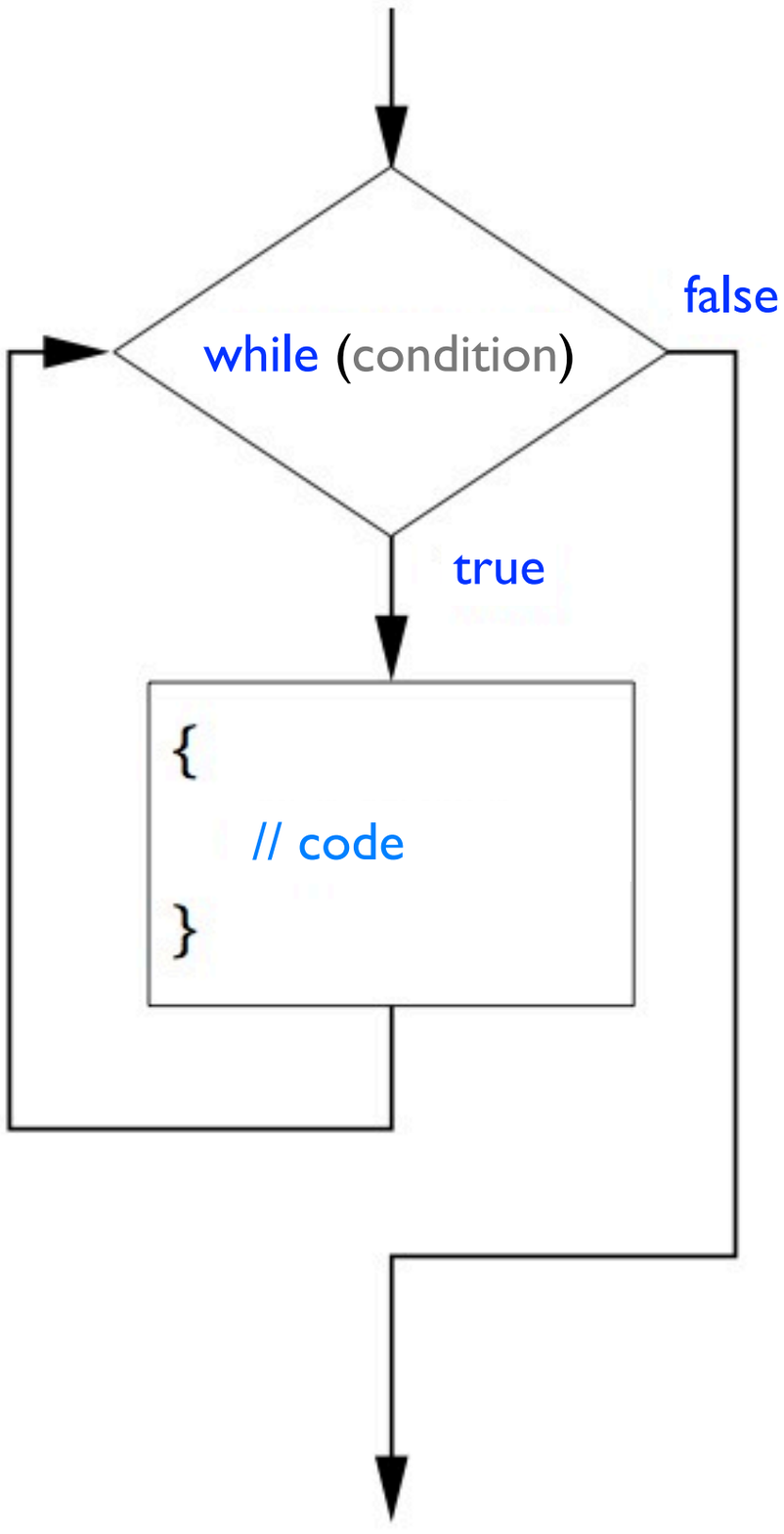
Common features:

```
// specify a starting condition
while (condition) {
    // code to repeat until the condition is false
    // modify the condition somehow
}
```

Loops (generally) share three common features:

- a starting condition (such as a counting or boolean flag variable)
- a condition check that occurs before every iteration of the loop
- some way to modify the variables used in the condition (increment a counting variable, change the value of a flag variable on a certain event, etc.)

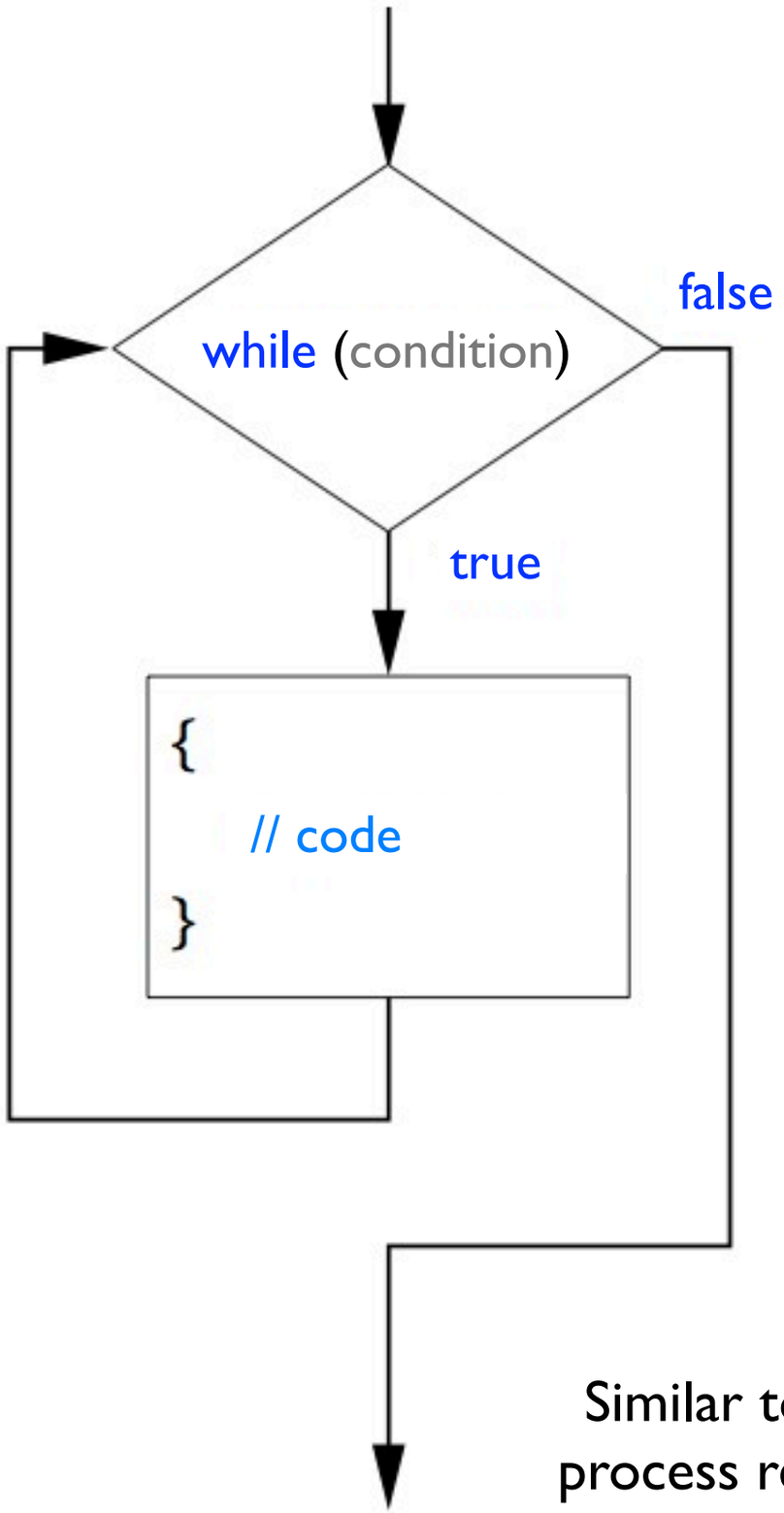
The while loop

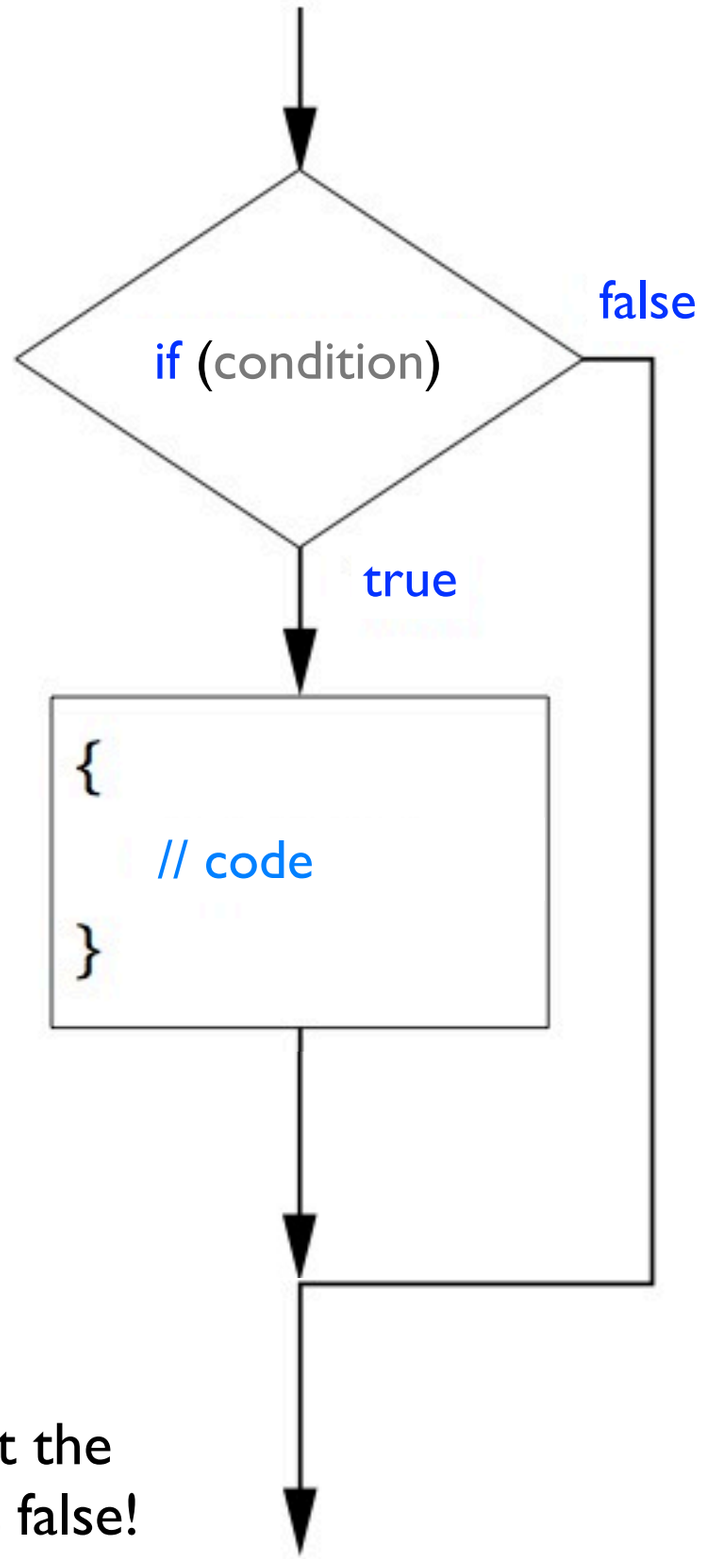


For each *iteration* of the while loop:

- the condition gets evaluated
- if true, the block of code associated with the while loop gets executed and the loop proceeds to the next iteration
- if false, the loop is finished

The while loop





Similar to a lone if statement, except the process repeats until the condition is false!

Common Loop Patterns

Repeat a block of code a *specific number of times*:

```
// declare an integer counting variable
int counter = 1;


while (counter <= 10) {
    cout << “Behold, the number ” << counter << “!” << endl;
    counter++; // modify the counter variable
}
```

Common Loop Patterns

Repeat a block of code until a *specific condition* is met

```
// set to a value that initially satisfies condition (why?)
char letter = '\0'; // \0 is called the 'null' character

// keep prompting the user until he enters a valid value
while (letter < 'A' || letter > 'Z') {
    cout << "Please enter an uppercase letter: ";
    cin >> letter;
}

// congratulate the user on his intellectual prowess
cout << "Thanks! " << letter << " is an uppercase letter." << endl;
```

Common Loop Patterns

Repeat a block of code until a *specific value* is entered:

```
cout << “Let’s play random number generator!” << endl;
char playAgain = ‘Y’;

while (playAgain != ‘N’) {
    cout << “Random number: 3” << endl;

    // ask the user if he wants to play again
    cout << “Play again? (enter N to quit) ”;
    cin >> playAgain;
}
```

Infinite Loops

If the condition on which a loop is predicated never happens, then the loop will repeat without end.

Infinite loops are frequently caused by logic errors

- the condition is logically impossible
- the values used in the condition are never modified
- assignment is done instead of check for equality
- two doubles are compared using the equality operator (doubles have limited precision!)

Sometimes you might intentionally create an infinite loop

- … but we'll wait to cover a case like this until we talk about the break keyword

Practice

Answer me these questions three!

The code:

```
int bottles = 99;


while (bottles) {
    cout <<   bottles << " bottles of beer on the wall, "
         <<   bottles << " bottles of beer!\n";
    cout << "You take one down and pass it around; "
         << --bottles << " bottles of beer on the wall!\n";
}
```

The questions:

- how many times will the loop execute?
- where is the condition modified?
- what... is your favorite color?!?

And three more for good measure!

Loop 1:

```
int number = 10;


while (--number) {
    cout << number << endl;
}
```

Loop 2:

```
int number = 10;


while (number) {
    cout << --number << endl;
}
```

The questions:

- is the output produced by these loops the same, or different?
- what numbers will be output in each case?
- what if we had used the postfix decrement operator instead?

Fat-Finger Dialing

If the President (mistakenly) called you and asked you to save the Western world by writing a while loop to print out each of the letters of the alphabet, could you do it? COULD YOU???

Imagine that, in a heroic display of patriotism and self-sacrifice, you accept the challenge… Print out each of the letters of the alphabet (either upper- or lowercase), on the same line and without spaces!

Bad Timing

In an alternate future, Earth is still under threat from the Buggers, and Ender is on vacation… Scientists have recently discovered that Buggers are allergic to sequences of perfect squares (1, 4, 9, 16, etc…).

Make the Buggers slightly uncomfortable by writing a loop that prints the squares of the first `100` positive integers!

Not every situation is life-threatening...

Pretend you're really bored right now (don't worry, I know you're not *actually* bored).

Initialize variables x to `1` and y to `1,000,000,000`. Write a loop that multiplies x by `2` and divides y by `3` until x is bigger than y. After the loop ends, print out the number of iterations required.