Loops; While Loop

ITP 165 – Fall 2015
Week 3, Lecture 1

Problem: 99 Bottles of Beer Song

99 bottles of beer on the wall, 99 bottles of beer.

Take one down, pass it around, 98 bottles of beer on the wall!

Writing a program to “sing” the song

- Naïve approach would be something like this:

```
std::cout << "99 bottles of beer..." << std::endl;
std::cout << "98 bottles of beer..." << std::endl;
std::cout << "97 bottles of beer..." << std::endl;
// Write out 96 more verses...
```

Rephrasing the Problem

- Let's think about a better algorithm…

1. Start with bottles = 99
2. Sing the verse for the current number of bottles
3. Subtract one bottle
4. Repeat steps 2 and 3 until there are 0 bottles left

- What's the key word in this algorithm that makes it so concise?

Rephrasing the Problem, Cont'd

1. Start with bottles = 99
2. Sing the verse for the current number of bottles
3. Subtract one bottle

4. Repeat steps 2 and 3 until there are 0 bottles left

Repetition in a program

- If your algorithm has the word “repeat” in it, it likely means you want to use a loop

- A ***loop*** is a type of statement that allows any number of statements to continue to execute so long as a particular condition is true

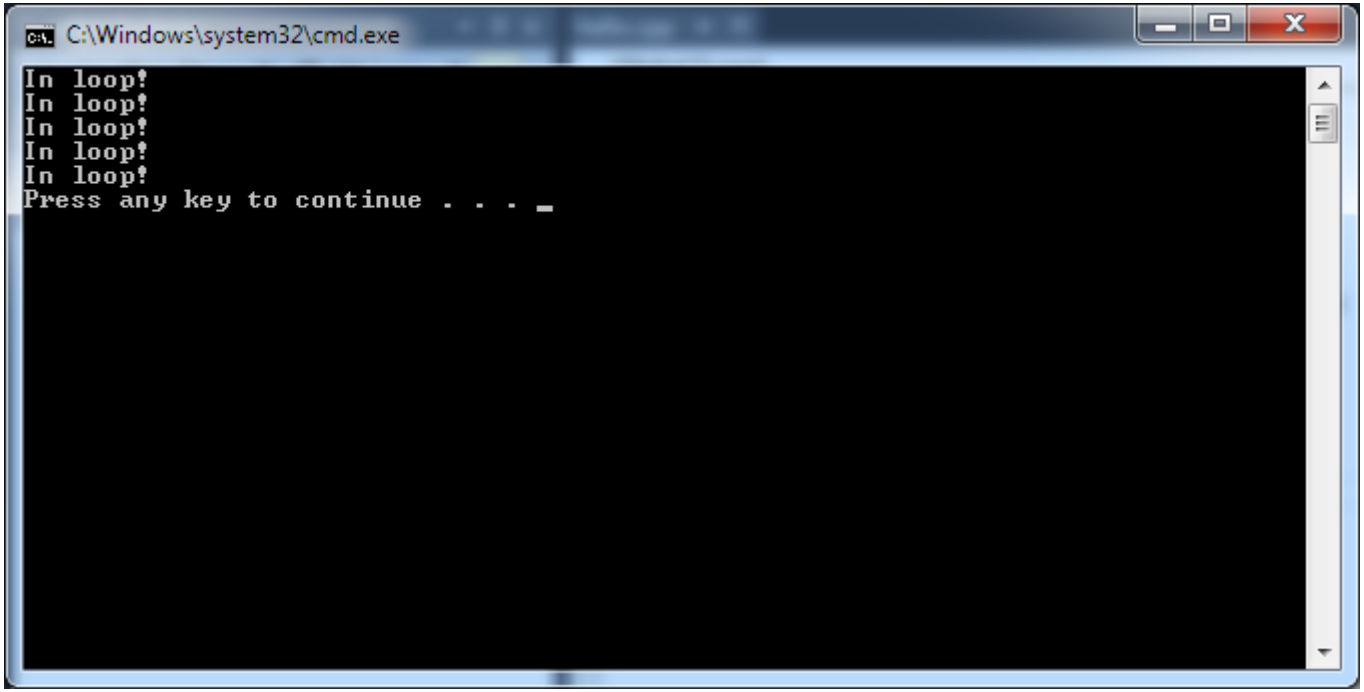
- Each time the statements in the body of a loop execute is an ***iteration***

A simple program with a loop

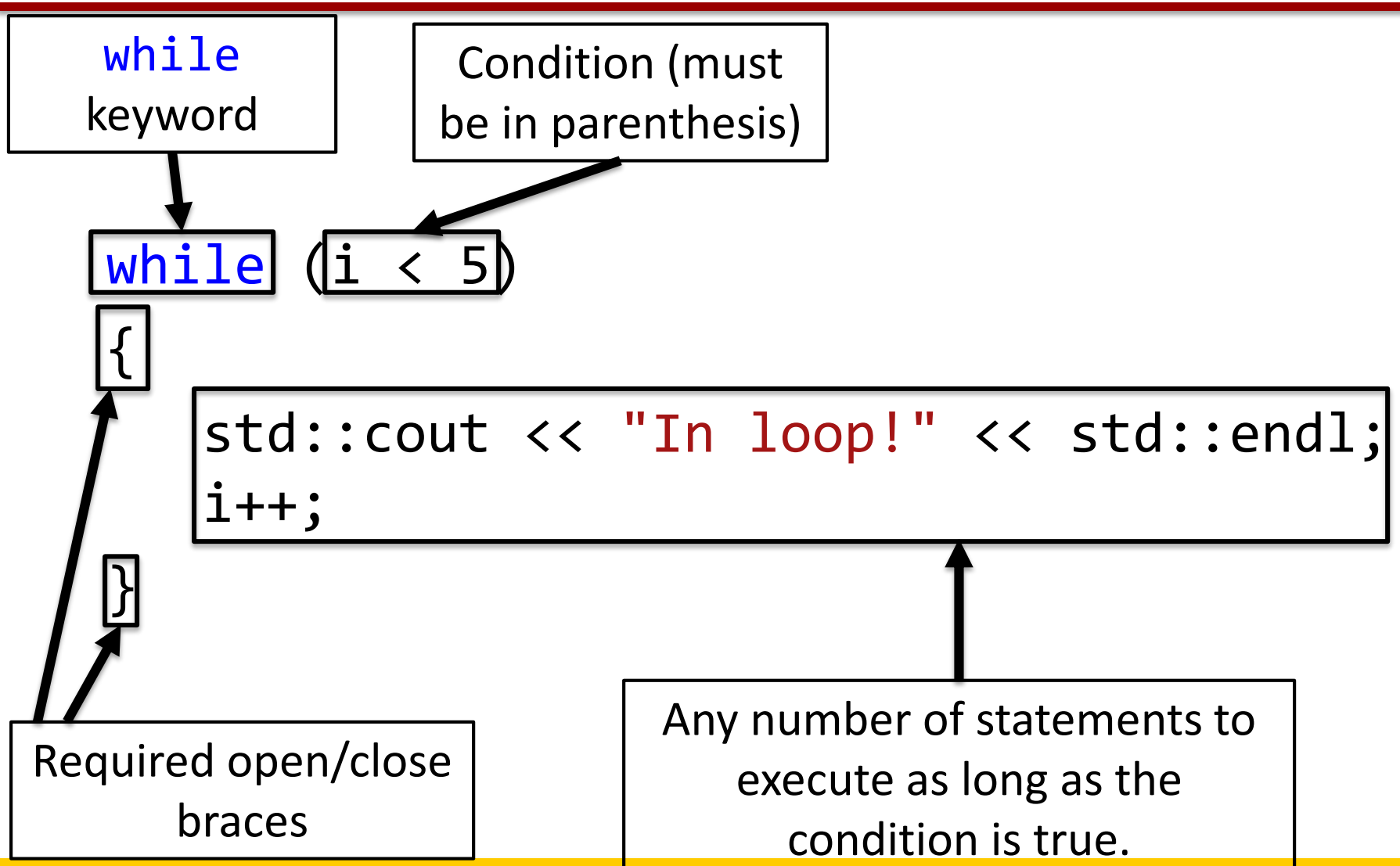
```
#include <iostream>
int main()
{
    int i = 0;
    while (i < 5)
    {
        std::cout << "In loop!" << std::endl;
        i++;
    }

    return 0;
}
```

- How many iterations does this loop have?

A simple program with a loop, Cont'd



While Loop Syntax



A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

The process the program goes through when running a while loop:

1. Check the condition. If the condition is false, exit the loop. Otherwise continue to step 2.
2. Execute the statements inside the braces.
3. Goto step 1

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	0

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	0

1. Check the condition. If the condition is false, exit the loop. Otherwise continue to step 2.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	1

2. Execute the statements inside the braces.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	1

3. Goto step 1.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	1

1. Check the condition. If the condition is false, exit the loop. Otherwise continue to step 2.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	2

2. Execute the statements inside the braces.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	2

3. Goto step 1.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	2

1. Check the condition. If the condition is false, exit the loop. Otherwise continue to step 2.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	3

2. Execute the statements inside the braces.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	3

3. Goto step 1.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	3

1. Check the condition. If the condition is false, exit the loop. Otherwise continue to step 2.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	4

2. Execute the statements inside the braces.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	4

3. Goto step 1.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	4

1. Check the condition. If the condition is false, exit the loop. Otherwise continue to step 2.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	5

2. Execute the statements inside the braces.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	5

3. Goto step 1.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}
```

Variable	Value
i	5

1. Check the condition. If the condition is false, exit the loop. Otherwise continue to step 2.

A simple program with a loop – step by step

```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
    i++;
}

return 0;
```

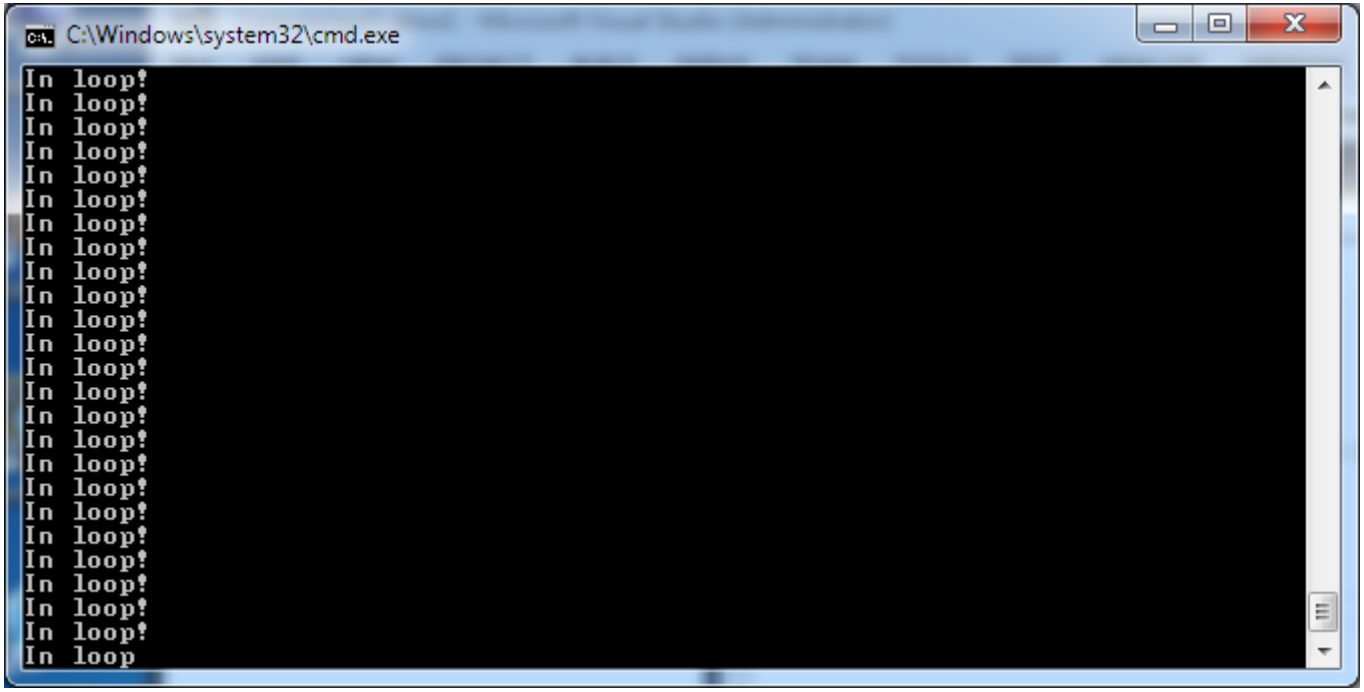
Variable	Value
i	5

A slight modification

- What happens if we remove the i++ line?

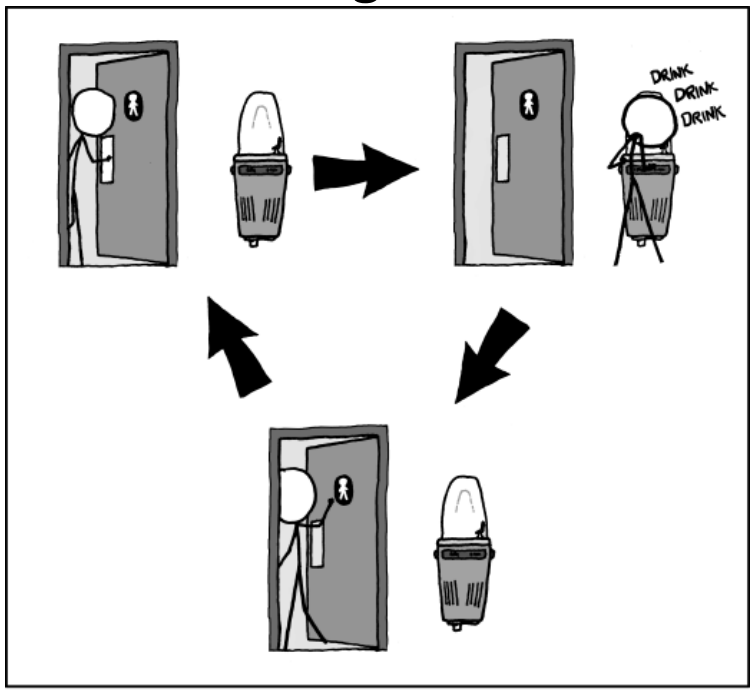
```
int i = 0;
while (i < 5)
{
    std::cout << "In loop!" << std::endl;
}
```

A slight modification, cont'd



Infinite Loop

- A loop is considered ***infinite*** if there is no way the condition can ever be false
- Generically, it can also just mean the algorithm will get stuck doing the same thing over and over again:



I AVOID DRINKING FOUNTAINS OUTSIDE BATHROOMS BECAUSE I'M AFRAID OF GETTING TRAPPED IN A LOOP.

99 Bottles of Beer, Revisted

- Recall that our better algorithm was:

1. Start with bottles = 99
2. Sing the verse for the current number of bottles
3. Subtract one bottle
4. Repeat steps 2 and 3 until there are 0 bottles left

- How can we rewrite this as a loop?

99 Bottles of Beer, Code

```
int bottles = 99;
while (bottles > 0)
{
    std::cout << bottles << " bottles of beer..." << std::endl;
    bottles--;
}
```

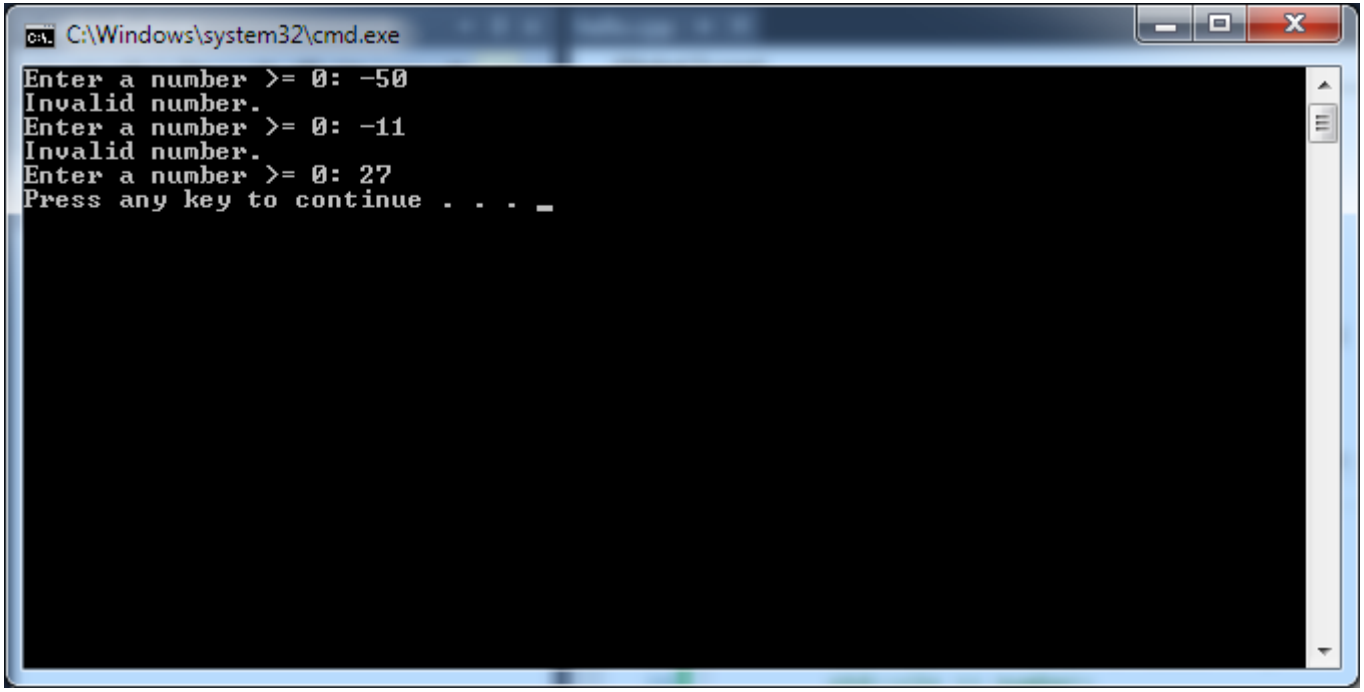
99 Bottles of Beer, In Action

Using loops to validate input

- This loop will not exit until you enter a number >= 0:

```
int number = -1;
while (number < 0)
{
    std::cout << "Enter a number >= 0: ";
    std::cin >> number;
    if (number < 0)
    {
        std::cout << "Invalid number." << std::endl;
    }
}
```

Using loops to validate input, cont'd



Using loops to validate input, another way

- We could also have written that code as:

```
bool validNumber = false;
int number = -1;
while (!validNumber)
{
   std::cout << "Enter a number >= 0: ";
   std::cin >> number;
   if (number < 0)
   {
      std::cout << "Invalid number." << std::endl;
   }
   else
   {
      validNumber = true;
   }
}
```

break Statement

- The `break` statement immediately exits the current loop

- For example:

```
while (true)
{
    break;
}
```

Using loops to validate input, a third way

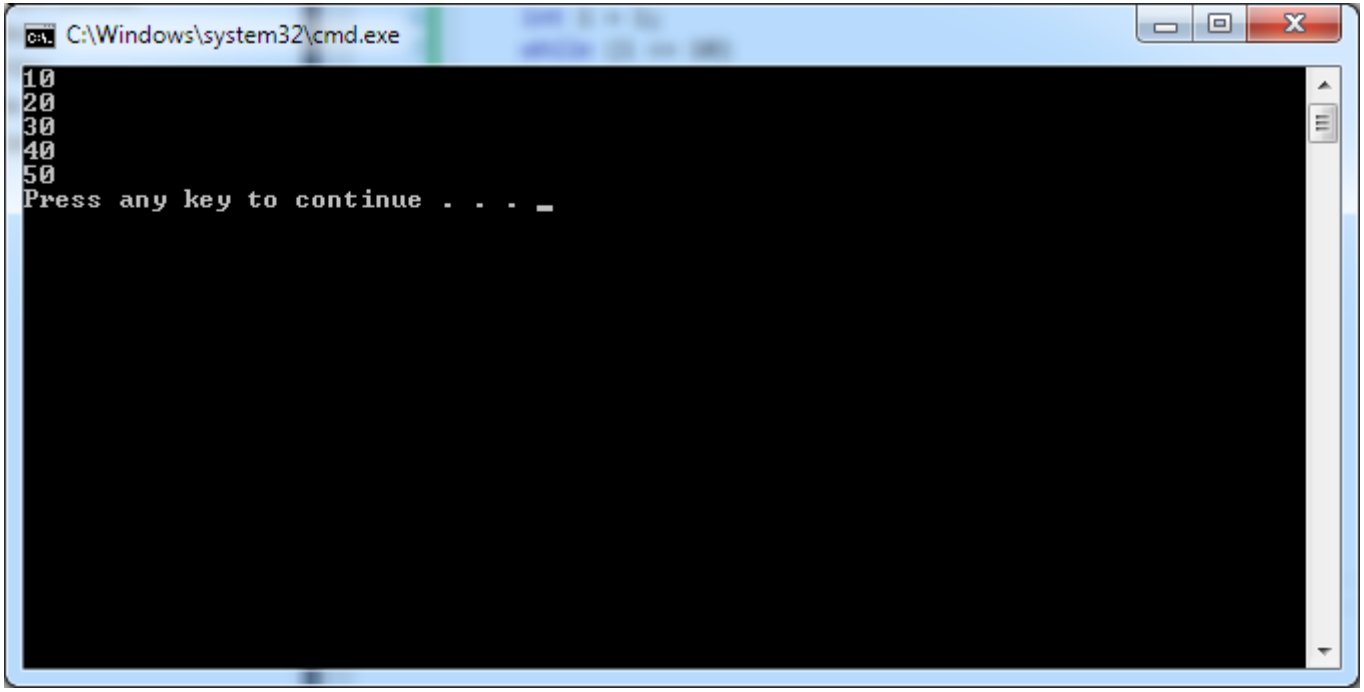
```
int number = -1;
while (true)
{
    std::cout << "Enter a number >= 0: ";
    std::cin >> number;
    if (number >= 0)
    {
        break;
    }

    std::cout << "Invalid number." << std::endl;
}
```

continue Statement

- The continue statement skips the remainder of the body for the current iteration

```
// Output even numbers from 1-10, multiplied by 5
int i = 1;
while (i <= 10)
{
   // If this is true, it means
   // the number is not divisible by 2
   // (aka it's odd)
   if (i % 2 != 0)
   {
      i++;
      continue;
   }

   std::cout << i * 5 << std::endl;
   i++;
}
```

Previous Code in Action



A caveat on continue/break

- Oftentimes, it is cleaner to write code that ***does not*** use `continue` or `break`. For example, the code for the previous loop could be rewritten as:

```
// Output even numbers from 1-10, multiplied by 5
int i = 1;
while (i <= 10)
{
   // i % 2 is 0 when even
   if (i % 2 == 0)
   {
      std::cout << i * 5 << std::endl;
   }

   i++;
}
```

Nesting Loops

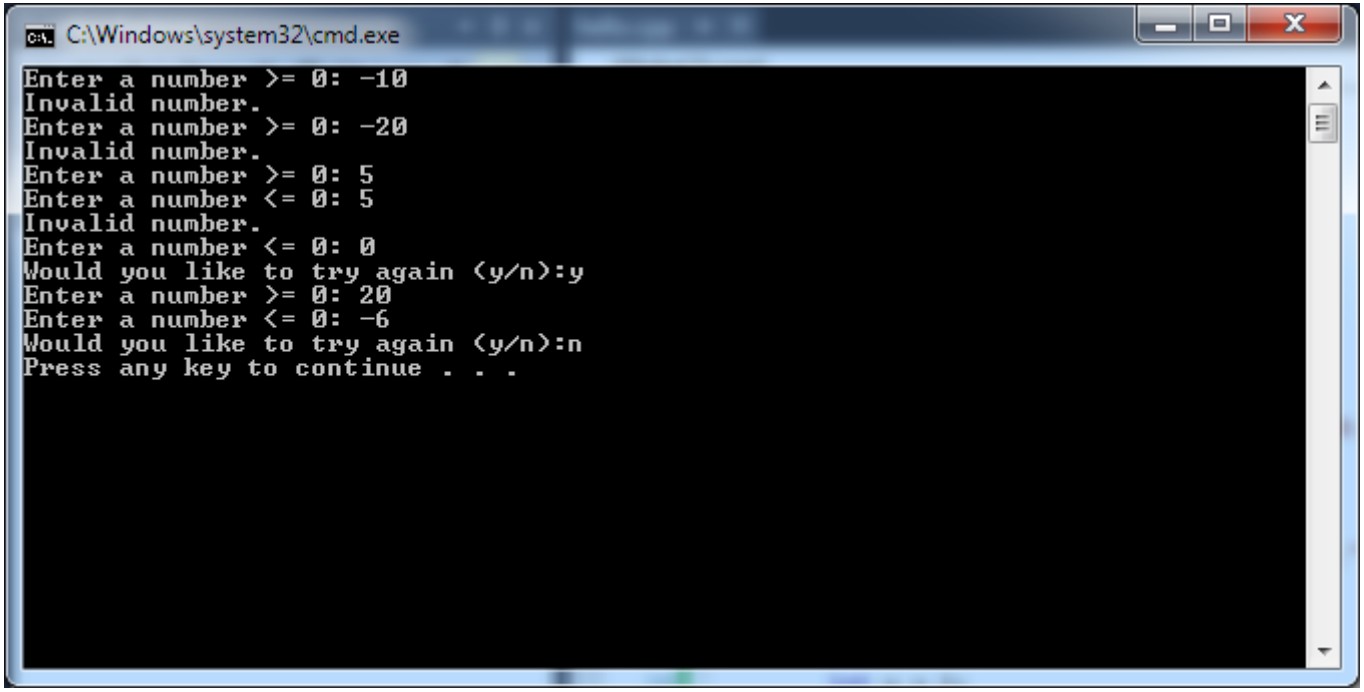
- Since a loop can contain any statements inside of its body…

- You can nest loops, just like you can nest conditionals.

Nesting Loops – A complex example

```
bool tryAgain = true;
while (tryAgain) {
   int x = -1;
   while (x < 0) {
      std::cout << "Enter a number >= 0: ";
      std::cin >> x;
      if (x < 0) {
         std::cout << "Invalid number." << std::endl;
      }
   }

   int y = 1;
   while (y > 0) {
      std::cout << "Enter a number <= 0: ";
      std::cin >> y;
      if (y > 0) {
         std::cout << "Invalid number." << std::endl;
      }
   }

   std::string select;
   std::cout << "Would you like to try again (y/n):";
   std::cin >> select;
   if (select == "n" || select == "no") {
      tryAgain = false;
   }
}
```

The complex example in action



Nesting Loops and continue/break

- Remember that continue and break only operate on the current loop

- So for example, in the code below, the outer loop will continue to run:

```
while (true)
{
    while (true)
    {
        break;
    }
}
```

Lab Practical #4