Lesson 7-1: For Loops

Computer Science 46A: Introduction to Programming

San José State University

Announcements

- Homework #7 will be posted
- Lab this Friday 3/14
- Midterm 3/27

Learning Objectives

By the end of this lesson, you should be able to:

1. Use a for loop to iterate a condition several times
2. Interpret when to use while or for loops, depending on the case
3. Implement code blocks within for loops to leverage their power

Recall: Loops

- A loop is a code block that that is repeated many times
- Its one of the most powerful components of programming
- There are a few implementation of loops in different languages
- In Java, there are three different implementations of loops:
  - While loop – if a condition is met, keep doing something
  - For loop – do something a certain amount of times
  - Do loop – do something, and then keep doing it if a condition is met

Recall: The `while` loop

- A “while loop” is a code block that repeats a command *while* a particular condition is true
- Syntax:

```
while (condition)
{
    // do this command
}
```

as long as this is true, do this

While loops are nice tools when you do not know when something will occur

The `for` loop

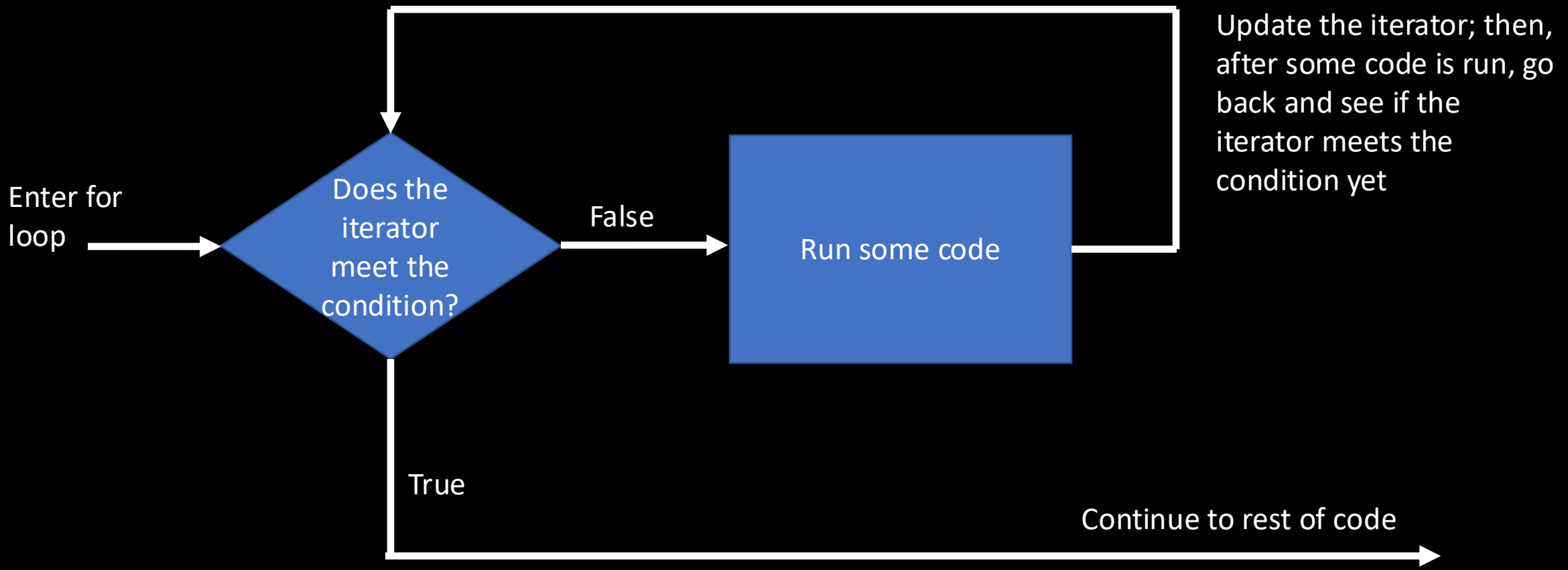
- A “for loop” is a code block that repeats a command *for* a certain amount times
- Syntax:

```
for (iterator; condition; update)
{
    // do this command
}
```

Update iterator each time - as long as it has not yet met the condition, run the code

`for` loops are nice tools when you know how many times you need to do something

The `for` loop flow chart



Example: EvenNumbers

Goal: Print all of the even numbers up to a certain number

```
Enter a positive integer: 14
The even numbers less than 14 are:
0, 2, 4, 6, 8, 10, 12,
```

Output of EvenNumbers

`while` and `for` loops are often interchangeable

- In many `while` loops, we can calculate how many times a particular iteration will occur
- A for loop is a better loop when:
  - it's possible to calculate the number of times something will occur
  - when there is a concrete limit for the iterator
- One major upshot is that it is harder to have an infinite loop

From Lecture 6-1: CountNumbers

What if we wanted to count all of the numbers between 1 and 100?

```
int number = 1;
int sum = 0;
while (number <= 100)
{
    sum = sum + number;
    number ++;
}
```

Can we turn this while loop into a for loop?

```
Enter a positive integer: 100
The sum of numbers from 1 to 100 is 5050
```

Output of CountNumbers

Updated example: CountNumbers

What if we wanted to count all of the numbers between 1 and 100?

```
int sum = 0;
for (int number = 1; number <= 100; number ++)
{
    sum = sum + number;
}
```

```
Enter a positive integer: 100
The sum of numbers from 1 to 100 is 5050
```

Output of CountNumbers

Comparing `while` and `for` loops

while Loops

- Controlled by events
- Easy to have an infinite loop
- Use when you're not sure when something will occur

for Loops

- Controlled by counts
- Harder to have an infinite loop
- Use when you know how many times you need to do something

Poll Everywhere 1: Which is better?

- Problem: Given a roster of students in CS 46A, how many people in this classroom are named “Mike”?

A) `while` loop

B) `for` loop

PollEv.com/narayanbalasubramanian644

Poll Everywhere 2: Which is better?

- Problem: What's the next prime number after 4391?

A) `while` loop

B) `for` loop

PollEv.com/narayanbalasubramanian644

The `for` loop with other statement blocks

- Just as with `while` loops, we can unleash the power of `for` loops by implementing code blocks within them
- For example, we we can implement an if/else statement into the code block:

```
for (iterator, condition, update)
{
    if (another condition)
    {
        // do this command
    }
}
```

Example: CountDivisors

What if we wanted to count all of the numbers in range which are divisible by a given number?

```
for (int number = lowerLimit; number <= upperLimit; number ++)
    {
        if (number%divisor==0)
        {
            sum++;
        }
    }
```

```
Enter the lower limit: 7
Enter the upper limit: 21
Enter the divisor: 7
In the range 7 to 21, there are 3 numbers divisible by 7
```

Output of CountDivisors

Participation Exercise 7-1a: `OddIndexChars`

Goal: Given a string, print all of the characters places at odd indices

```
nrdcint rgamn
```

Output of `OddIndexChars` for the string "`Introduction to Programming`"

Codecheck Link: HERE and on Canvas

Participation Exercise 7-1b: NumberOfDays

Goal: Count the number of days in the years between two given years

```
Enter the first year: 2000
Enter the second year: 2022
The number of days between 2000 and 2022 is 8401
```

Output of NumberOfDays

Hint: A normal year has 365 days while a leap year has 366 days.
A leap year is a year which is divisible by 4.
Careful not to use magic numbers (except 0, 1, 4)!

Codecheck Link: HERE and on Canvas