For Loops

Principles of Computer Programming I

Spring/Fall 20XX

AUGUSTA UNIVERSITY

Outline

- for loop basics
- Limitations and pitfalls
- More advanced for loops
- for loops with arrays

While Loops with Counters

- Notice a pattern in counter-controlled loops:

Initialize counter

```
int i = 0;
while(i < 10)
{
   Console.WriteLine($"{i}");
   i++;
}
Console.WriteLine("Done");
```

Stop when counter reaches a value

Increment counter on each iteration

Initialize counter

```
int num = 1, total = 0;
while(num <= 25)
{
   total += num;
   num++;
}
Console.WriteLine($"The sum is {total}");
```

Stop when counter reaches a value

Increment counter on each iteration

For Loops: Shorthand for Counters

- for statement combines initialization, increment, and condition

```
int i = 0;
while(i < 10)
{
  Console.WriteLine($"{i}");
  i++;
}
Console.WriteLine("Done");
```

Initialize counter   Loop condition   Increment

```
for(int i = 0; i < 10; i++)
{
  Console.WriteLine($"{i}");
}
Console.WriteLine("Done");
```

- 3 statements in 1 line, separated by semicolons

AUGUSTA UNIVERSITY

Details of the 3 Parts

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

- Initialization statement: Executed once when loop starts
- Condition statement: Loop continues if true, stops if false
  - Evaluated **before** executing loop body, like a `while` loop
- Update statement: Executed every time loop body **ends**

For Loop Operation

- First, create `i` and initialize to `0`
- Evaluate condition `i < 10`
  - True, so execute loop body
- At end of loop body, execute `i++`
- Return to beginning and evaluate condition again
- Last iteration: “9” is printed, then `i` increments to 10
- Now `i < 10` is false, so skip loop body and print “Done”

```
for(int i = 0; i < 10; i++)
{
  Console.WriteLine($"{i}");
}
Console.WriteLine("Done");
```

Outline

- for loop basics
- **Limitations and pitfalls**
- More advanced for loops
- for loops with arrays

Variable Scope in `for` Loops

- Variable declared in `for` loop has scope *inside* that loop's body
- Cannot be used after loop ends

```
int total = 0;
for(int count = 0; count < 10; count++)
{
  total += count;
}
Console.WriteLine($"The average is {(double) total / count}");
```

Imagine count is declared here

Error! No variable named count in scope

Using the Counter After the Loop

- Solution: Declare counter before loop, outside body
- Loop initialization must assign to it, not declare it

```
int total = 0;
int count;
for(count = 0; count < 10; count++)
{
  total += count;
}
Console.WriteLine($"The average is {(double) total / count}");
```

Variable declaration determines scope

Set count to 0, don't create it

count is still in scope

Pitfall: Re-declaring a Variable

- Variable declared in `for` loop must not already exist

```
int total = 0;
int count;
for(int count = 0; count < 10; count++)
{
  total += count;
}
Console.WriteLine($"The average is {(double) total / count}");
```

Error! Name count is already used

- Warning: counter variables have common names

Pitfall: Re-declaring a Variable

```
int i = 0; total = 0;
while(i < 10)
{
  total += i;
  i++;
}
Console.WriteLine($"The average is {(double) total / i}");
//Many lines later...
for(int i = 0; i < 10; i++) ←—— Error! Name i is already used
{
  Console.WriteLine($"{i}");
}
```

Does This Work?

```
total = 0;
for(int i = 0; i < 25; i++)
{
  total += i;
}
Console.WriteLine($"The total is {total}");
//Many lines later...
for(int i = 0; i < 10; i++)
{
  Console.WriteLine($"{i}");
}
```

From While to For

- Pitfall: Leaving the increment in the loop body

```
int i = 0;
while(i < 10)
{
  Console.WriteLine($"{i}");
  i++;
}
Console.WriteLine("Done");
```

```
for(int i = 0; i < 10; i++)
{
  Console.WriteLine($"{i}");
  i++;
}
Console.WriteLine("Done");
```

Now i will be incremented twice per loop

Outline

- for loop basics
- Limitations and pitfalls
- **More advanced for loops**
- for loops with arrays

AUGUSTA UNIVERSITY

Conditions with Variables

- Like `while` loops, condition can use a variable or method result

```
Console.WriteLine("Enter a positive number.");
int numTimes = int.Parse(Console.ReadLine());
for(int c = 0; c < numTimes; c++)
{
  Console.WriteLine("**********");
}
```

Loop condition based on user input

```
for(int i = 1; i <= (int) myItem.GetPrice(); i++)
{
  Console.WriteLine($"${i}");
}
```

Get the price of an Item, convert it to `int`

Counts out whole dollars in price

Update Can Be Other Operations

- Print the even numbers:

```
for(int i = 0; i < 19; i += 2)
{
  Console.WriteLine($"{i}");
}
```

- Count down to 0:

```
for(int t = 10; t > 0; t--)
{
  Console.Write($"{t}...");
}
Console.WriteLine("Liftoff!");
```

AUGUSTA UNIVERSITY

Loops and Other Conditions

- If statements can be nested inside loops:

```
for(int i = 0; i < 8; i++)
{
  if(i % 2 == 0)
  {
    Console.WriteLine("It's my turn");
  }
  else
  {
    Console.WriteLine("It's your turn");
  }
  Console.WriteLine("Switching players...");
}
```

Nesting Loops

- Loops can contain other loops:

```
for(int r = 0; r < 11; r++)
{
  for(int c = 0; c < 11; c++)
  {
    Console.Write($"{r} x {c} = {r * c} \t");
  }
  Console.Write("\n");
}
```

This loops 10 times on each iteration of the outer loop

Print one line of multiplications, separated by tabs

End the line after 10 entries

Combining While and For

- Ask the user to enter a positive number repeatedly, until they enter “Q” to quit. If the user enters a positive number, display that number of “*” symbols on one line. Ignore invalid input.
- Sentinel value: “Q”
  - Need to use a `while` loop
- Printing “*” symbols: counter-controlled loop
- Ignore invalid input: Need to use TryParse

Combining While and For

```
string userInput;
do
{
  Console.WriteLine("Enter a positive number, or \"Q\" to stop");
  userInput = Console.ReadLine();
  int.TryParse(userInput, out int inputNum);
  if(inputNum > 0) ⟵ Check for valid input
  {
    for(int c = 0; c < inputNum; c++) ⟵ for loop: prints the number
    {                                     of *'s the user requested
      Console.Write("*");
    }
    Console.WriteLine();
  }
} while(userInput != "Q"); ⟵ while loop: Checks for sentinel value
```

Outline

- for loop basics
- Limitations and pitfalls
- More advanced for loops
- **for loops with arrays**

Iterating Over Arrays

- With a `while` loop:

```
int i = 0;  ← Counter
int sum = 0;
while(i < myArray.Length)
{
  sum += myArray[i];
  i++;
}
double average = (double) sum /
  myArray.Length;
```

Loop condition: `i < myArray.Length`

Increment: `i++;`

- With a `for` loop:

```
int sum = 0;
for(int i = 0; i < myArray.Length; i++)
{
  sum += myArray[i];
}
double average = (double) sum /
  myArray.Length;
```

Counter: `int i = 0`

Loop condition: `i < myArray.Length`

Increment: `i++`

AUGUSTA UNIVERSITY

Filling Arrays with For Loops

- User-provided array size doesn't change the loop condition
- What if we want to add user-input validation?

```
Console.WriteLine("How many homework grades are there?");
int numGrades = int.Parse(Console.ReadLine());
double[] homeworkGrades = new double[numGrades];
for(int i = 0; i < homeworkGrades.Length; i++)
{
  Console.WriteLine($"Enter the grade for homework {i+1}");
  homeworkGrades[i] = double.Parse(Console.ReadLine());
}
```

For Loop with Nested While

```
Console.WriteLine("How many homework grades are there?");
int numGrades = int.Parse(Console.ReadLine());
double[] homeworkGrades = new double[numGrades];
for(int i = 0; i < homeworkGrades.Length; i++)
{
  bool isNum;
  do
  {
    Console.WriteLine($"Enter the grade for homework {i+1}");
    isNum = double.TryParse(Console.ReadLine(), out homeworkGrades[i]);
  } while(!isNum || homeworkGrades[i] < 0);
}
```

Adding a TryParse loop here won't fit on the slide