Foreach, Break and Continue

Principles of Computer Programming I

Spring/Fall 20XX

AUGUSTA UNIVERSITY

Outline

- For loops and arrays
- Foreach loops
- Break and continue

Searching Through an Array

- E.g. Determine if a `char` array named `initials` contains ‘B’:

```
bool foundB = false;
for(int i = 0; i < initials.Length; i++)
{
  if(initials[i] == 'B')
    foundB = true;
}
if(foundB)
  Console.WriteLine("Array contains 'B'");
else
  Console.WriteLine("Array does not contain 'B'");
```

Variable to store result of search

Loop counts through each index in array

Check each array element

Update variable if desired value is found

Use variable to display result

Index of the the First Appearance

```
bool foundB = false;
int firstIndex = 0;
for(int i = 0; i < initials.Length; i++)
{
  if(initials[i] == 'B' && !foundB)
  {
    foundB = true;
    firstIndex = i;
  }
}
if(foundB)
  Console.WriteLine($"'B' first appears at index {firstIndex}");
```

Two components of search: Whether 'B' has been found, and index of 'B'

Condition: Current element is 'B', and 'B' has not yet been found

Save result of search in variable

Index of the First Appearance

- Can we do this without the foundB variable?

```
int firstIndex = -1;
for(int i = 0; i < initials.Length; i++)
{
  if(initials[i] == 'B' && firstIndex == -1)
  {
    firstIndex = i;
  }
}
if(firstIndex != -1)
  Console.WriteLine($"'B' first appears at index {firstIndex}");
```

Initialize to a value that means "result not found"

Do not update firstIndex if it already has a value

Working with Two Arrays

- One loop can iterate over two arrays at once
- Example: Compute the **dot product** of two vectors
  - Definition: $[a_1, a_2, a_3] \cdot [b_1, b_2, b_3] = a_1 b_1 + a_2 b_2 + a_3 b_3$

```
double[] vec1 = ...
double[] vec2 = ...
double dotProduct = 0;
for(int i = 0; i < vec1.Length; i++)
{
  dotProduct += vec1[i] * vec2[i];
}
```

Accumulator variable

Vectors must be the same length

Multiply elements in the same position

Nested Loops with Two Arrays

- For each value in `targets`, determine if it is in `myNums`
  - Example:
    ```
    targets = {4, 2, 6, 8}
    myNums = {6, 9, 3, 4, 5}
    ```
  - Result should be: 4 is in `myNums`, 2 is not in MyNums, 6 is in `myNums`, 8 is not in `myNums`
- Need 2 loops: One for `targets`, one for `myNums`

Nested Loops with Two Arrays

- For each value in `targets`, determine if it is in `myNums`

```
for(int i = 0; i < targets.Length; i++)
{
  bool found = false;
  for(int j = 0; j < myNums.Length; j++)
  {
    if(myNums[j] == targets[i])
      found = true;
  }
  if(found)
    Console.WriteLine($"myNums contains {targets[i]}");
  else
    Console.WriteLine($"{targets[i]} is not in myNums");
}
```

1 loop iteration = 1 element of `targets`

1 loop iteration = 1 element of `myNums`

Stays the same during inner loop

Outline

- For loops and arrays
- **Foreach loops**
- Break and continue

A Loop Shortcut

- for loops over arrays all look the same:

```
for(int i = 0; i < myArray.Length; i++)
{
  <do something with myArray[i]>;
}
```

- If you only need to **read** the array entries, a shorter form:

```
int sum = 0;
foreach(int grade in homeworkGrades)
{
  sum += grade;
}
```

Name of array

Variable to hold each value in array

Contains homeworkGrades[0], then homeworkGrades[1], etc.

foreach Rules

- General form:

```
foreach(type <variableName> in <arrayName>)
{
  <do something with variableName>;
}
```

Scope of variable = body of loop

- Type of variable must match type of array element

```
string[] days = {"Mon", "Tue", "Wed", "Thu", "Fri"};
foreach(string day in days)
{
  Console.WriteLine(day);
}
```

A Shorter "Find" Loop

Scope = body of outer foreach loop

```
foreach(int target in targets)
{
  bool found = false;
  foreach(int num in myNums)
  {
    if(num == target)
      found = true;
  }
  if(found)
    Console.WriteLine($"myNums contains {target}");
  else
    Console.WriteLine($"{target} is not in myNums");
}
```

No need for variables i and j

Keeps the same value from targets as num changes with each iteration

foreach Limitations

- Cannot be used to **change** values in array

```
int[] homeworkGrades = new int[5];
foreach(int grade in homeworkGrades)
{
  Console.WriteLine("Please enter the next grade");
  grade = int.Parse(Console.ReadLine());
}
```

Error! Can't assign to grade

foreach Limitations

- No counter variable, so can't save index of array value

```
int firstIndex = -1;
foreach(char letter in initials)
{
  if(letter == 'B' && firstIndex == -1)
  {
    firstIndex = ????
  }
}
if(firstIndex != -1)
  Console.WriteLine($"'B' first appears at index {firstIndex}");
```

What is the index of `letter`?

Outline

- For loops and arrays
- Foreach loops
- **Break and continue**

Conditional Loop Control

- What if you want to skip some iterations of the loop?
- Example: Only use even values from array, skip odd values

```
int sum = 0;
for(int i = 0; i < myArray.Length; i++)
{
  if(myArray[i] % 2 == 0)
  {
    Console.WriteLine(myArray[i]);
    sum += myArray[i];
  }
}
Console.WriteLine(sum);
```

Check if current value is even

Entire loop body inside an `if` block

Skipping Iterations

- `continue` keyword = "skip this loop iteration"
- Return to loop beginning, increment counter, check condition

```
int sum = 0;
for(int i = 0; i < myArray.Length; i++)
{
  if(myArray[i] % 2 != 0)
    continue;
  Console.WriteLine(myArray[i]);
  sum += myArray[i];
}
Console.WriteLine(sum);
```

Check if current value is **odd**

Immediately start next iteration

These are only executed if `myArray[i]` is even

Continue with `while` Loops

- `continue` *immediately* returns to beginning of loop
- In a `while` loop, the header does not increment a counter

```
int sum = 0, i = 0;
while(i < myArray.Length)
{
  if(myArray[i] % 2 != 0)
    continue;
  Console.WriteLine(myArray[i]);
  sum += myArray[i];
  i++;
}
```

Execution stops here

Counter increment

```
int sum = 0, i = 0;
while(i < myArray.Length)
{
  if(myArray[i] % 2 != 0)
  {
    i++;
    continue;
  }
  Console.WriteLine(myArray[i]);
  sum += myArray[i];
  i++;
}
```

Always increment counter before loop body ends

Multiple End Conditions

- Scenario: Loop should end when a sentinel value is encountered, or when input is invalid

```
int sum = 0, userNum = 0;
bool success = true;
while(success && userNum >= 0)
{
  sum += userNum;
  Console.WriteLine("Enter a positive number to add it. "
    + "Enter anything else to stop.");
  success = int.TryParse(Console.ReadLine(), out userNum);
}
```

Extra variable to store parsing success

0 is a valid input, doesn't indicate failure

Another Way to End the Loop

- break keyword = "stop execution here" – ends the loop

```
int sum = 0, userNum = 0;
while(userNum >= 0)
{
  sum += userNum;
  Console.WriteLine("Enter a positive number to add it. "
    + "Enter anything else to stop.");
  if(!int.TryParse(Console.ReadLine(), out userNum)
    break;
}
```

Simpler condition, no variable needed

If TryParse failed, end the loop

Using break in a for Loop

- Find the index of the first occurrence of the character ‘B’
- Two end conditions: Searched entire array, or ‘B’ is found

```
int firstIndex = -1;
for(int i = 0; i < initials.Length; i++)
{
  if(initials[i] == 'B')
  {
    firstIndex = i;
    break;
  }
}
if(firstIndex != -1)
  Console.WriteLine($"'B' first appears at index {firstIndex}");
```

“Normal” end condition

Instead of adding a condition, just stop the loop when ‘B’ is found

More break with for Loops

- Scenario: Array is partially filled in with numbers, but at some (unknown) point, all the rest are zeroes

34	2	18	80	12	0	0	0	0

- Find product of all "filled" elements in array
  - Can't use "unfilled" elements, since product would be 0

More break with for Loops

- Find product of all elements up to the first 0

34	2	18	80	12	0	0	0	0

```
int product = 1;
for(int i = 0; i < myArray.Length; i++)
{
  if(myArray[i] == 0)
    break;
  product *= myArray[i];
}
Console.WriteLine($"Product: {product}");
```

Normal loop end: after last value in array

"Special" loop end: Encounter a zero

Using break in a foreach Loop

- Previous loop could also be written with foreach
- break has exactly the same effect

```
int product = 1;
foreach(int value in myArray)
{
  if(value == 0)
    break;
  product *= value;
}
Console.WriteLine($"Product: {product}");
```

Loops once for each element of myArray

Stops loop early if a 0 is found

Arrays and Loops Problem

- Your program is given two `int` arrays, `limits` and nums. For each element in `limits`, find and display the first element of nums (searching left to right) that is less than that limit. If no number in nums is less than the limit, display "None" instead.
- Example:
  - `limits = {4, 2, 6, 8}`
  - `nums = {6, 9, 3, 5, 4}`
  - Results should be: "3 None 3 6"

One Possible Solution

```
foreach(int limit in limits)
{
  int firstSmaller = limit;
  foreach(int num in nums)
  {
    if(num < limit)
    {
      firstSmaller = num;
      break;
    }
  }
  if(firstSmaller < limit)
    Console.WriteLine(firstSmaller);
  else
    Console.WriteLine("None");
}
```