Iterative Constructs

SESSION 6

Determining Average Magnitude

- Suppose we want to calculate the average apparent brightness of a list of five star magnitude values
  - Can we do it
    - Yes, it would be easy

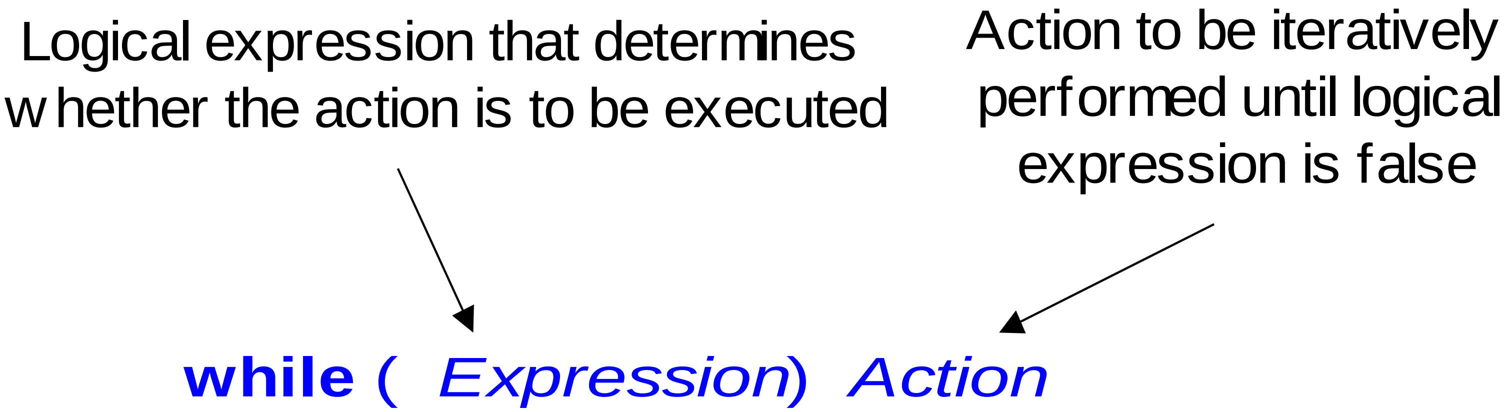
- Suppose we want to calculate the average apparent brightness of a list of 8,479 stars visible from earth
  - Can we do it
    - Yes, but it would be horrible without the use of iteration

C# Iterative Constructs

Four constructs

- while statement
- for statement
- do-while statement
- foreach statement

While Syntax

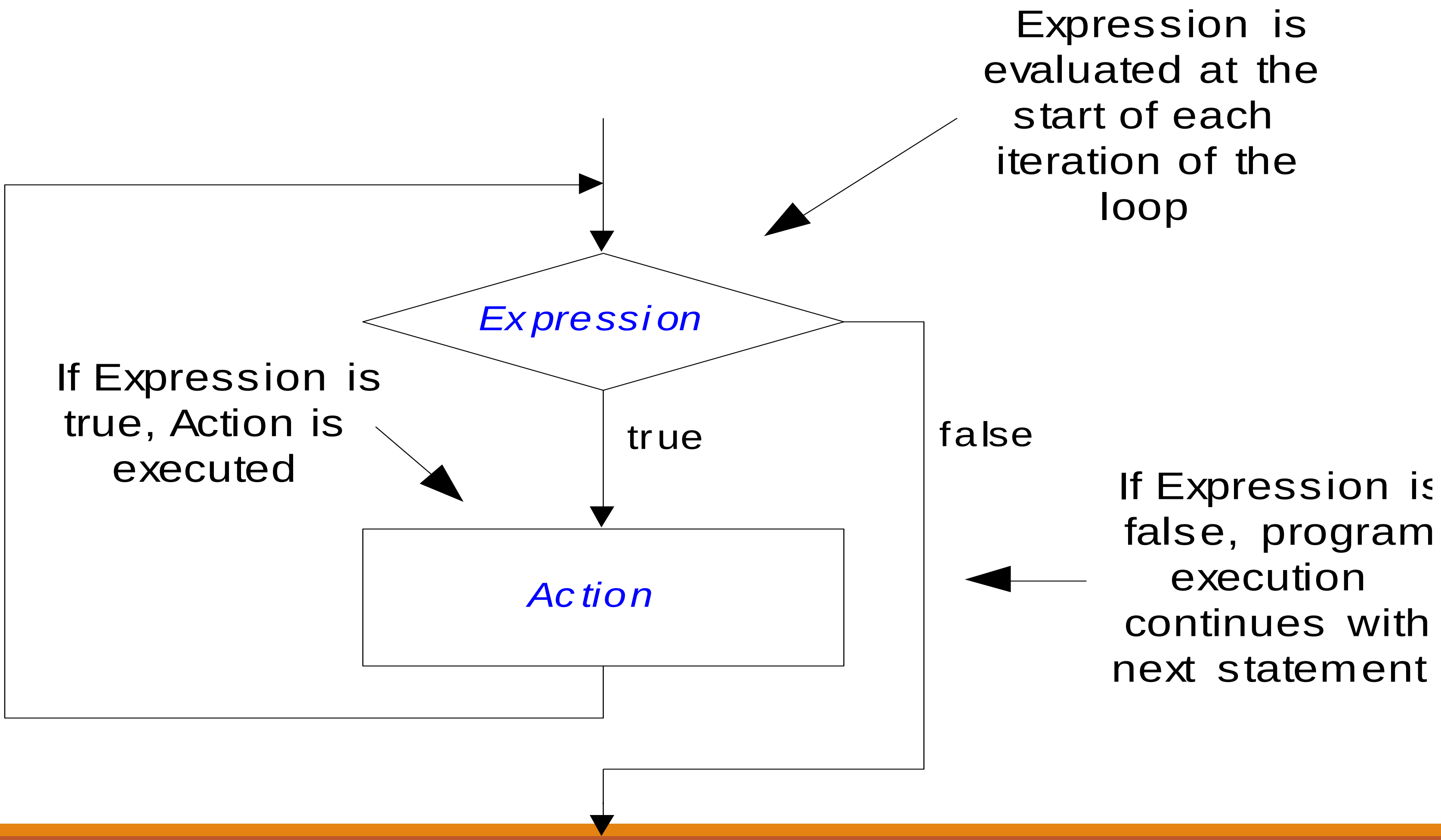


Logical expression that determines whether the action is to be executed

Action to be iteratively performed until logical expression is false

while (*Expression*) *Action*

While Semantics



Expression is evaluated at the start of each iteration of the loop

Expression

If Expression is true, Action is executed

true

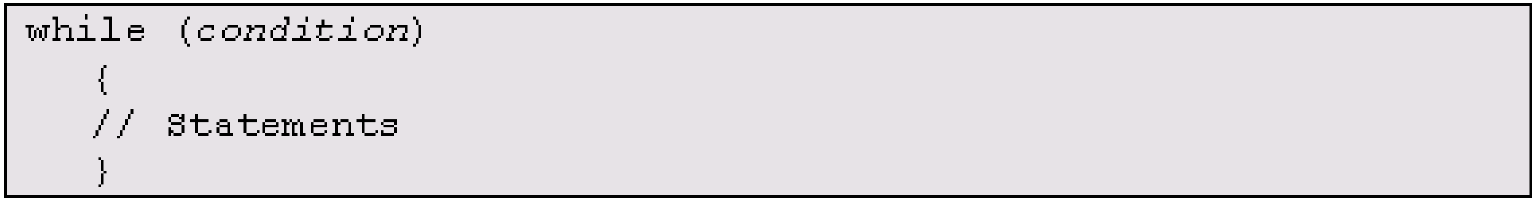
false

Action

If Expression is false, program execution continues with next statement

The while loop

- The while loop iterates through the specified statements till the specified condition is true.
- Syntax :



```
while (condition)
    {
    // Statements
    }
```

- The break statement breaks out of the loop at anytime.
- The continue statement skips the current iteration and begins with the next iteration.

Computing an Average

```
int listSize = 4;
int numberProcessed = 0;
int sum = 0;
while (numberProcessed < listSize) {
 int value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

listSize 4

Execution Trace

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	0

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

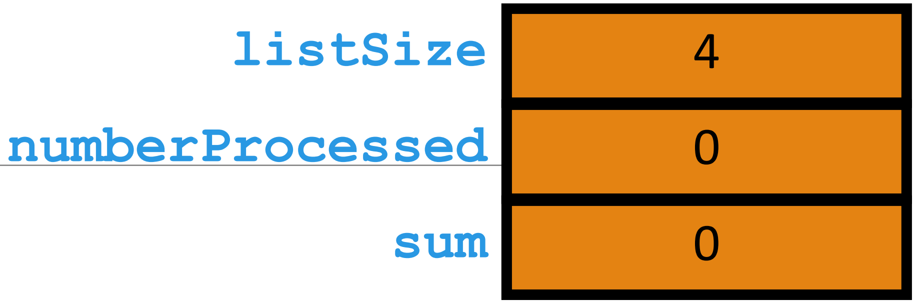
Execution Trace

listSize	4
numberProcessed	0
sum	0

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace


listSize 4

numberProcessed 0

sum 0

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	0
sum	0
value	--

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

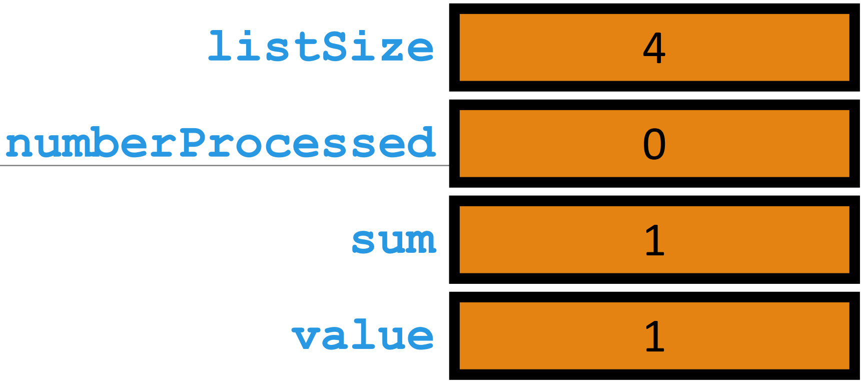
Execution Trace

listSize	4
numberProcessed	0
sum	0
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace



listSize	4
numberProcessed	0
sum	1
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

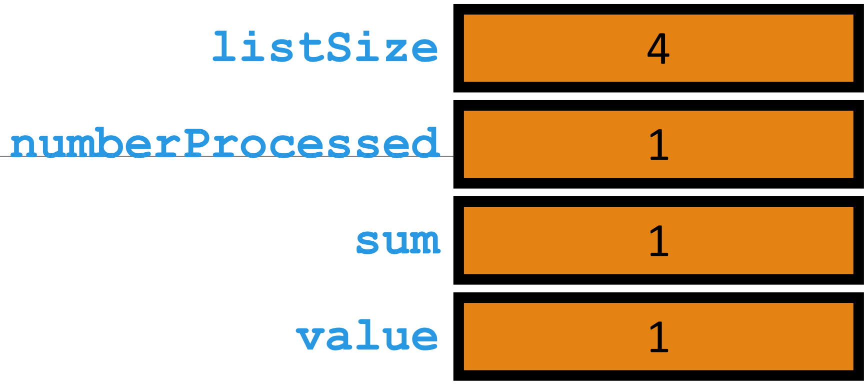
Execution Trace

listSize	4
numberProcessed	1
sum	1
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace



listSize 4

numberProcessed 1

sum 1

value 1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

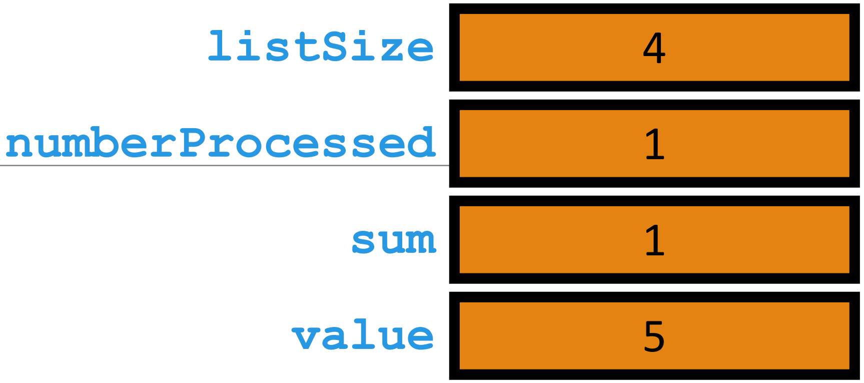
Execution Trace

listSize	4
numberProcessed	1
sum	1
value	--

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace



listSize 4

numberProcessed 1

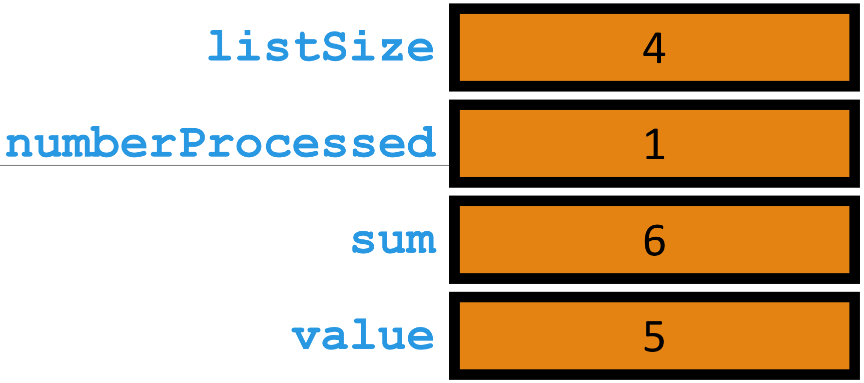
sum 1

value 5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace



listSize	4
numberProcessed	1
sum	6
value	5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	2
sum	6
value	5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}", average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	2
sum	6
value	5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

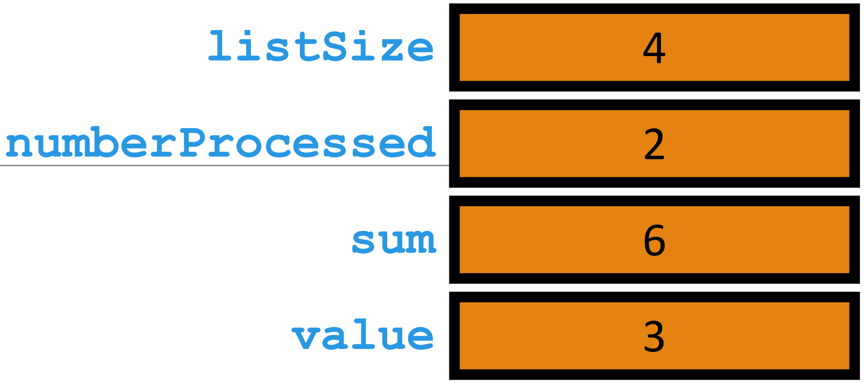
Execution Trace

listSize	4
numberProcessed	2
sum	6
value	--

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}", average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace



listSize 4

numberProcessed 2

sum 6

value 3

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

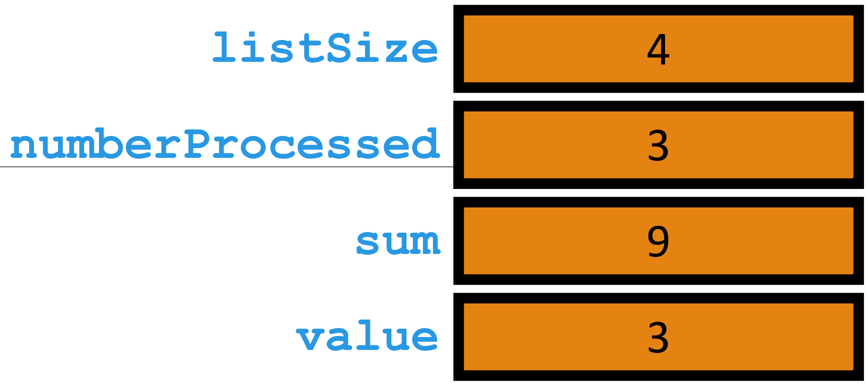
Execution Trace

listSize	4
numberProcessed	2
sum	9
value	3

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace



listSize 4

numberProcessed 3

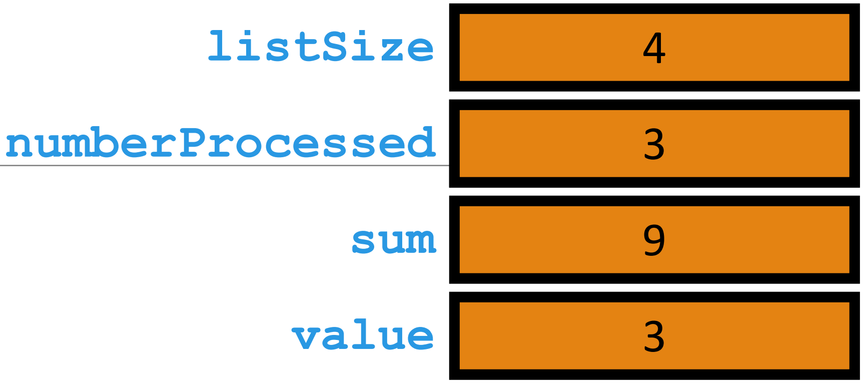
sum 9

value 3

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace



```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	3
sum	9
value	--

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}", average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	3
sum	9
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}", average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	3
sum	10
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	4
sum	10
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}", average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	4
sum	10
value	1

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	4
sum	10
average	2.5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
       sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

listSize	4
numberProcessed	4
sum	10
average	2.5

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
 value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Suppose input contains: 1 5 3 1 6

Execution Trace

Stays in stream until extracted

```
int listSize = 4;
int numberProcessed = 0;
double sum = 0;
while (numberProcessed < listSize) {
 double value;
  value = Convert.ToInt32(Console.ReadLine());
 sum += value;
 ++numberProcessed;
}
double average = sum / numberProcessed ;
Console.WriteLine("Average: {0}”, average);
```

Power of Two Table

```
const int TableSize = 20;


int i = 0;

long Entry = 1;


Console.WriteLine(“i \t\t 2 ** i“);


while (i < TableSize) {
 Console.WriteLine(“{0} \t\t {1}“,i, Entry);
 Entry = 2 * Entry;
 ++i;

}
```

Better Way of Averaging

```
int numberProcessed = 0;

double sum = 0;

double value;

while (value = Convert.ToInt32(Console.ReadLine())    {

 sum += value;

 ++numberProcessed;

}

double average = sum / numberProcessed ;

Console.WriteLine("Average: {0}”, average);
```

The value of the input operation corresponds to true only if a successful extraction was made

What if list is empty?

Even Better Way of Averaging

```
int numberProcessed = 0;
double sum = 0;
double value;
while (value = Convert.ToInt32(Console.ReadLine())  {
 sum += value;
 ++numberProcessed;
}
if ( numberProcessed > 0 ) {
 double average = sum / numberProcessed ;
 Console.WriteLine("Average: {0}”, average);
}
else {
 Console.WriteLine("No list to average“);
}
```

The For Statement

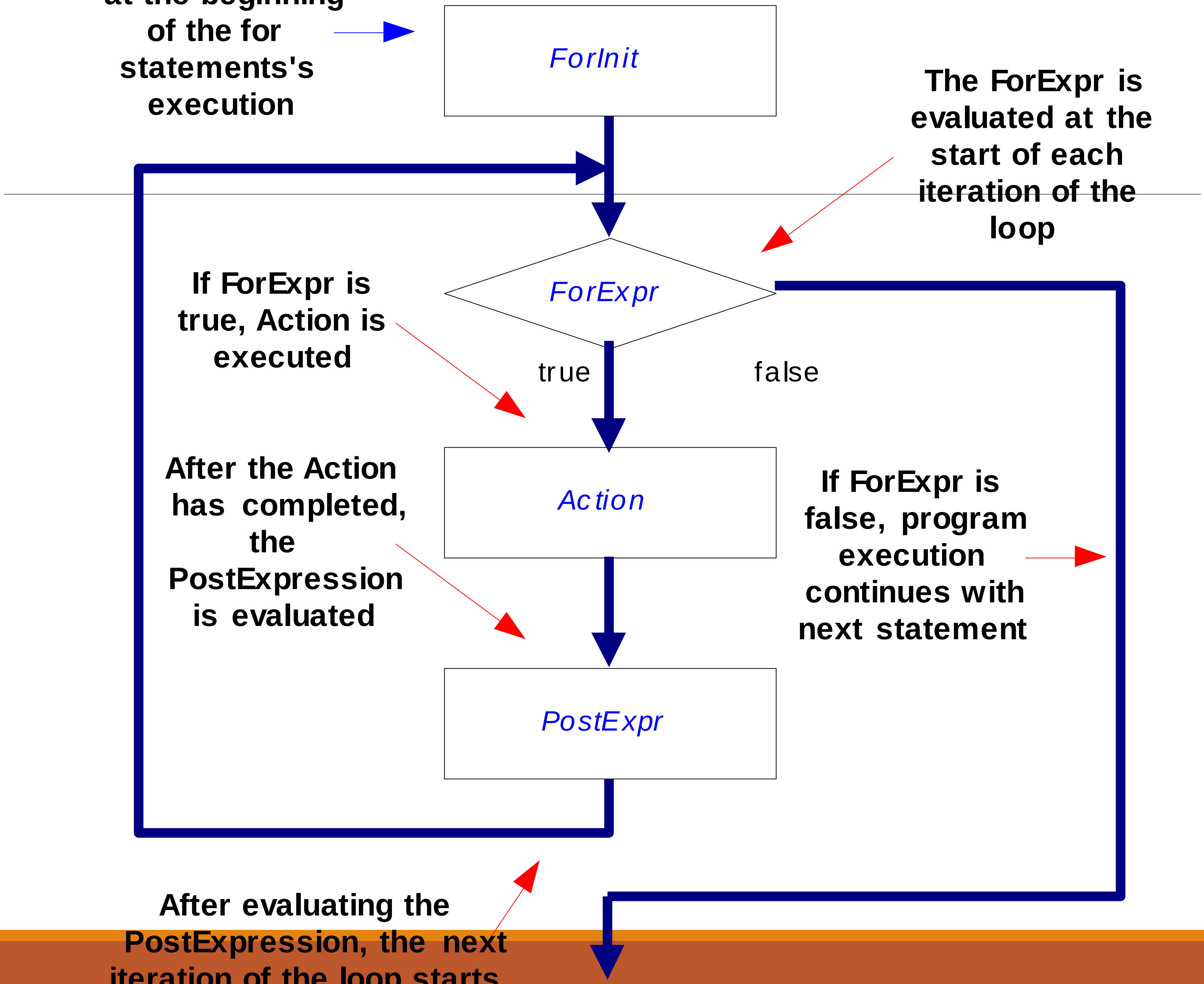
Syntax

`for` (*ForInit* ; *ForExpression*; *PostExpression*)

Action

Example

```
for (int i = 0; i < 3; ++i) {
   Console.WriteLine("i is {0}‟, i);

}
```



Evaluated once at the beginning of the for statements's execution

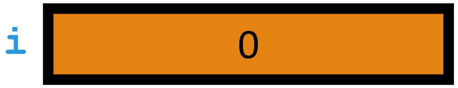
The ForExpr is evaluated at the start of each iteration of the loop

If ForExpr is true, Action is executed

After the Action has completed, the PostExpression is evaluated

If ForExpr is false, program execution continues with next statement

After evaluating the PostExpression, the next iteration of the loop starts


i

0

Execution Trace

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}“, i);

}

Console.WriteLine("all done“);
```

Execution Trace

i 0

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}“, i);

}

Console.WriteLine("all done“);
```

Execution Trace

i 0

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}“, i);
}
Console.WriteLine("all done“);
```

```
i is 0
```

Execution Trace

i 0

```
for (int i = 0; i < 3; ++i) {
  Console.WriteLine("i is {0}“, i);

}

Console.WriteLine("all done“);
```

```
i is 0
```

Execution Trace

i 1

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}‟, i);

}

Console.WriteLine("all done‟);
```

```
i is 0
```

Execution Trace

i 1

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}“, i);

}

Console.WriteLine("all done“);
```

Execution Trace

i 1

```
for (int i = 0; i < 3; ++i) {
  Console.WriteLine("i is {0}“, i);

}

Console.WriteLine("all done“);
```

```
i is 0

i is 1
```

Execution Trace

i 1

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}“, i);

}

Console.WriteLine("all done“);
```

```
i is 0

i is 1
```

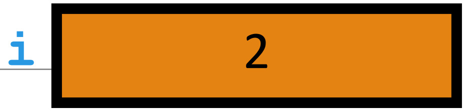
Execution Trace

i 2

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}“, i);
}
Console.WriteLine("all done“);
```

```
i is 0
i is 1
```

Execution Trace

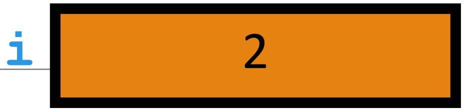

i

2

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}“, i);

}

Console.WriteLine("all done“);
```

```
i is 0

i is 1
```

Execution Trace



```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}“, i);
}
Console.WriteLine("all done“);
```

```
i is 0
i is 1
i is 2
```

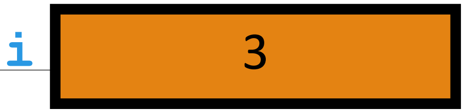
Execution Trace

i 2

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}“, i);

}

Console.WriteLine("all done“);
```

```
i is 0

i is 1

i is 2
```

Execution Trace


i

3

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}‟, i);
}
Console.WriteLine("all done‟);
```

```
i is 0
i is 1
i is 2
```

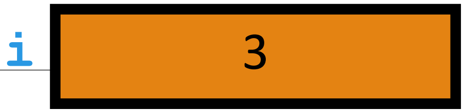
Execution Trace

i 3

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}“, i);
}
Console.WriteLine("all done“);
```

```
i is 0
i is 1
i is 2
```

Execution Trace


i

3

```
for (int i = 0; i < 3; ++i) {
 Console.WriteLine("i is {0}“, i);

}

Console.WriteLine("all done“);
```

```
i is 0

i is 1

i is 2

all done
```

Table Revisiting

```
const int TableSize = 20;


long Entry = 1;


Console.WriteLine(“i \t\t 2**i“);


for (int i = 0; i <= TableSize; ++i) {
 Console.WriteLine(“{0} \t\t {1}“,i,Entry);
 Entry *= 2;

}
```

Table Revisiting

```
const int TableSize = 20;


long Entry = 1;


Console.WriteLine(“i \t\t 2**i“);



for (int i = 0; i <= TableSize; ++i) {
 Console.WriteLine(“{0} \t\t {1}“,i,Entry);
 Entry *= 2;

}

Console.WriteLine(“i is {0}“,i); // illegal
```

The scope of i is limited to the loop!

Nested Loop

```
int Counter1 = 0;
int Counter2 = 0;
int Counter3 = 0;
int Counter4 = 0;
int Counter5 = 0;

++Counter1;

for (int i = 1; i <= 10; ++i) {

         ++Counter2;

         for (int j = 1; j <= 20; ++j) {
             ++Counter3;
         }

         ++Counter4;
}

++Counter5;

Console.WriteLine(“{0} {1} {2} {3} {4}”, Counter1, Counter2, Counter3, Counter4,
Counter5);
```

Output:

```
1 10 200 10 1
```

Iteration Do’s

Key Points

- Make sure there is a statement that will eventually terminate the iteration criterion
  - The loop must stop!
- Make sure that initialization of loop counters or iterators is properly performed
- Have a clear purpose for the loop
  - Document the purpose of the loop
  - Document how the body of the loop advances the purpose of the loop

The Do-While Statement

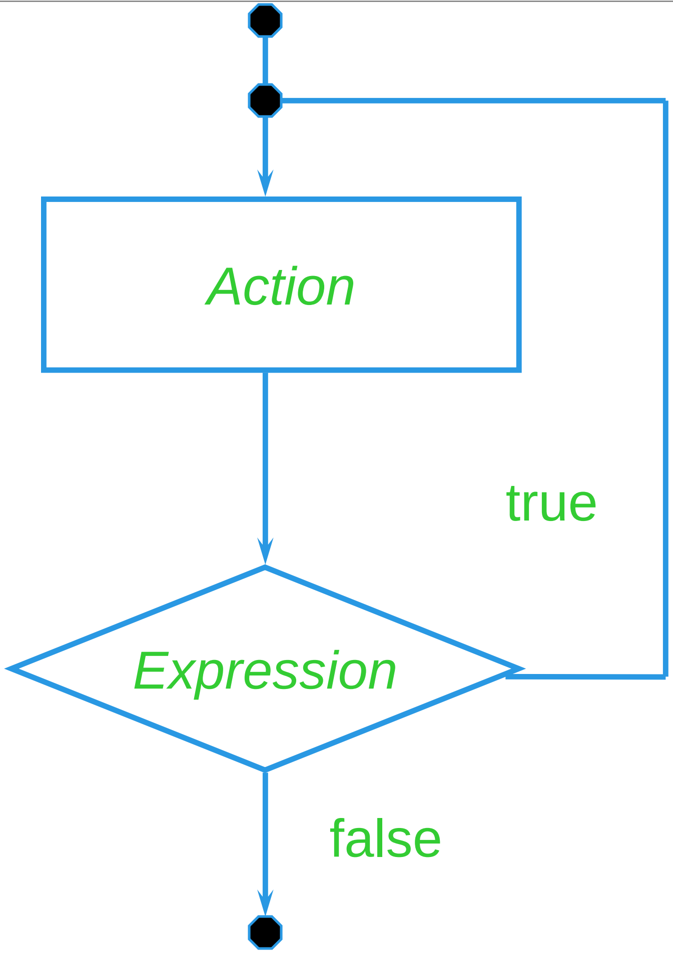
Syntax

```
    do Action
   while (Expression)
```

Semantics

- Execute *Action*
- If *Expression* is true then execute *Action* again
- Repeat this process until *Expression* evaluates to false

Action is either a single statement or a group of statements within braces

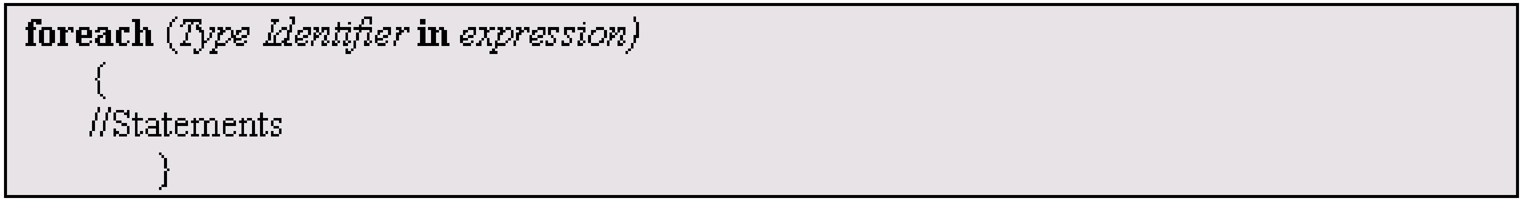


Waiting for a Proper Reply

```
char Reply;
do {
   Console.WriteLine("Decision (y, n): “);
     Reply =
 Convert.ToChar(Console.ReadLine());
} while ((Reply == 'y') || (Reply == ‘Y'));
```

The foreach loop (1)

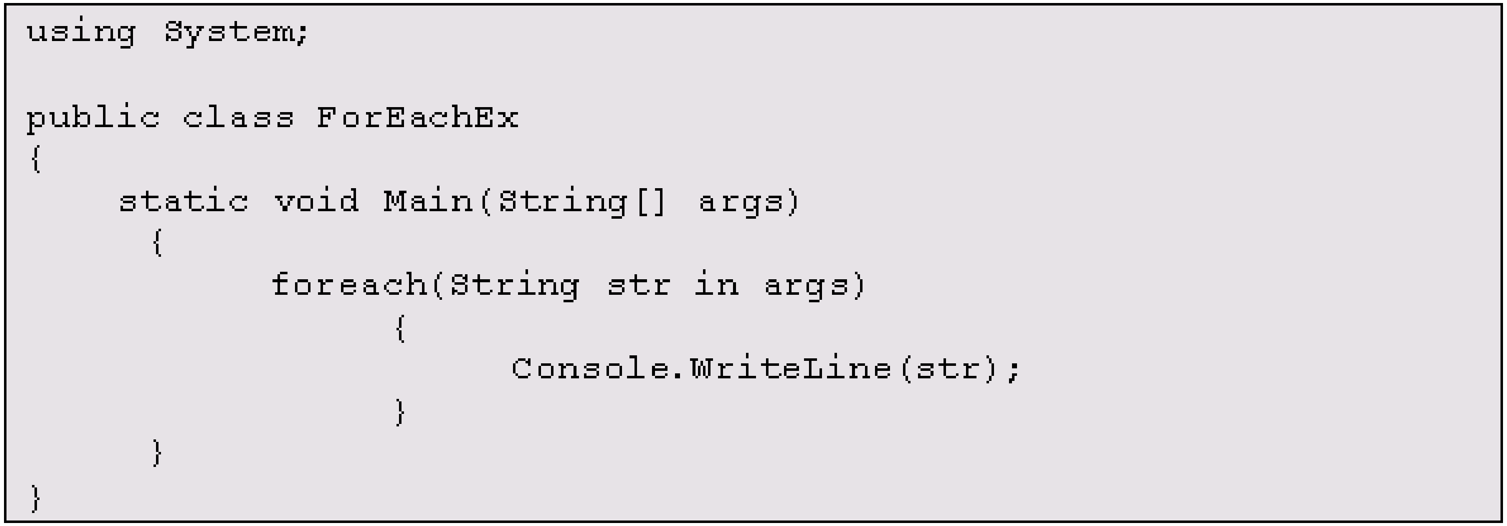
- The foreach loop is used to iterate through a collection or an array.
- Syntax:



```
foreach (Type Identifier in expression)
      {
      //Statements
            }
```

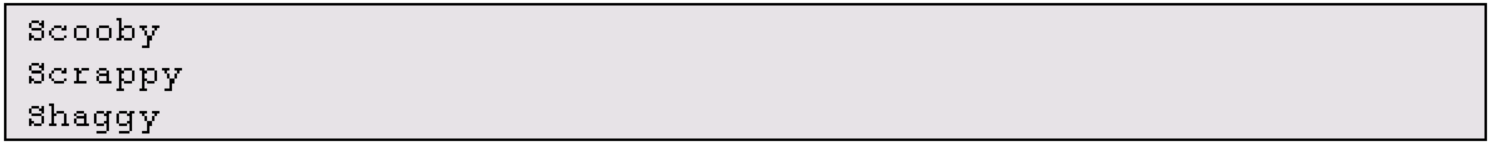
The foreach loop (2)

- Example:



```
using System;

public class ForEachEx
{
    static void Main(String[] args)
     {
           foreach(String str in args)
                {
                     Console.WriteLine(str);
                }
     }
}
```

- Output:



```
Scooby
Scrappy
Shaggy
```