Telerik Academy

# Loops

## Execute Blocks of Code Multiple Times

**C# Fundamentals**

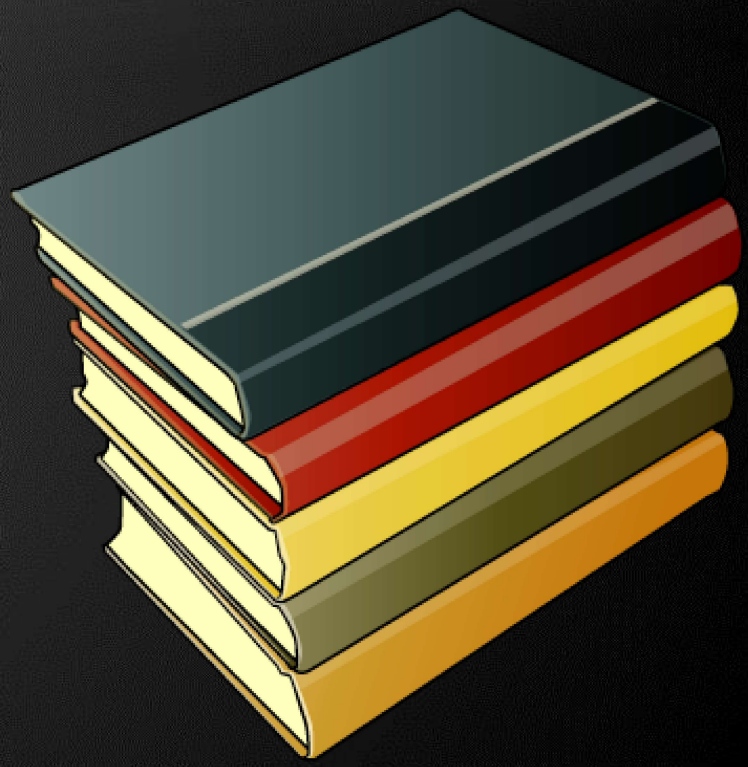
Telerik Software Academy

https://telerikacademy.com

Telerik Academy

# Table of Contents





Telerik Academy

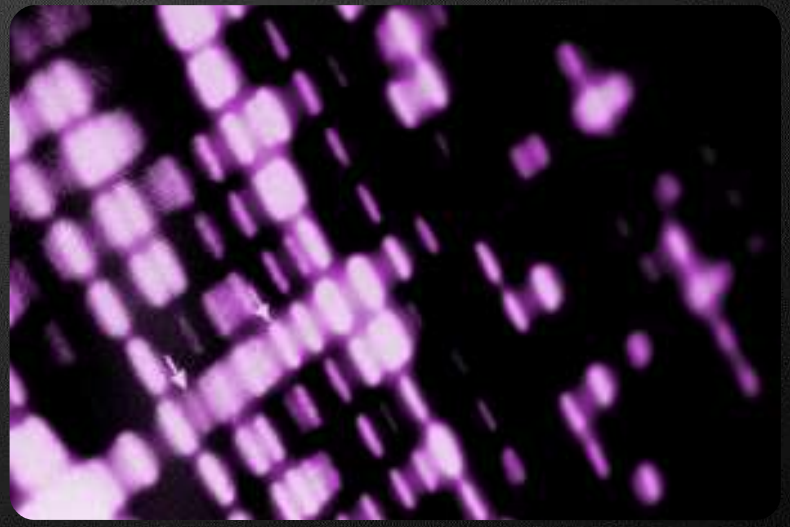
# What Is Loop?

- A `loop` is a control statement that allows repeating execution of a block of statements
  - May execute a code block fixed number of times
  - May execute a code block while given condition holds
  - May execute a code block for each member of a collection
- Loops that never end are called an `infinite loops`



Telerik Academy

# Using while(…) Loop



Telerik Academy

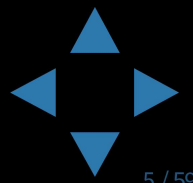
# How to use `while` loop?

- The simplest and most frequently used loop

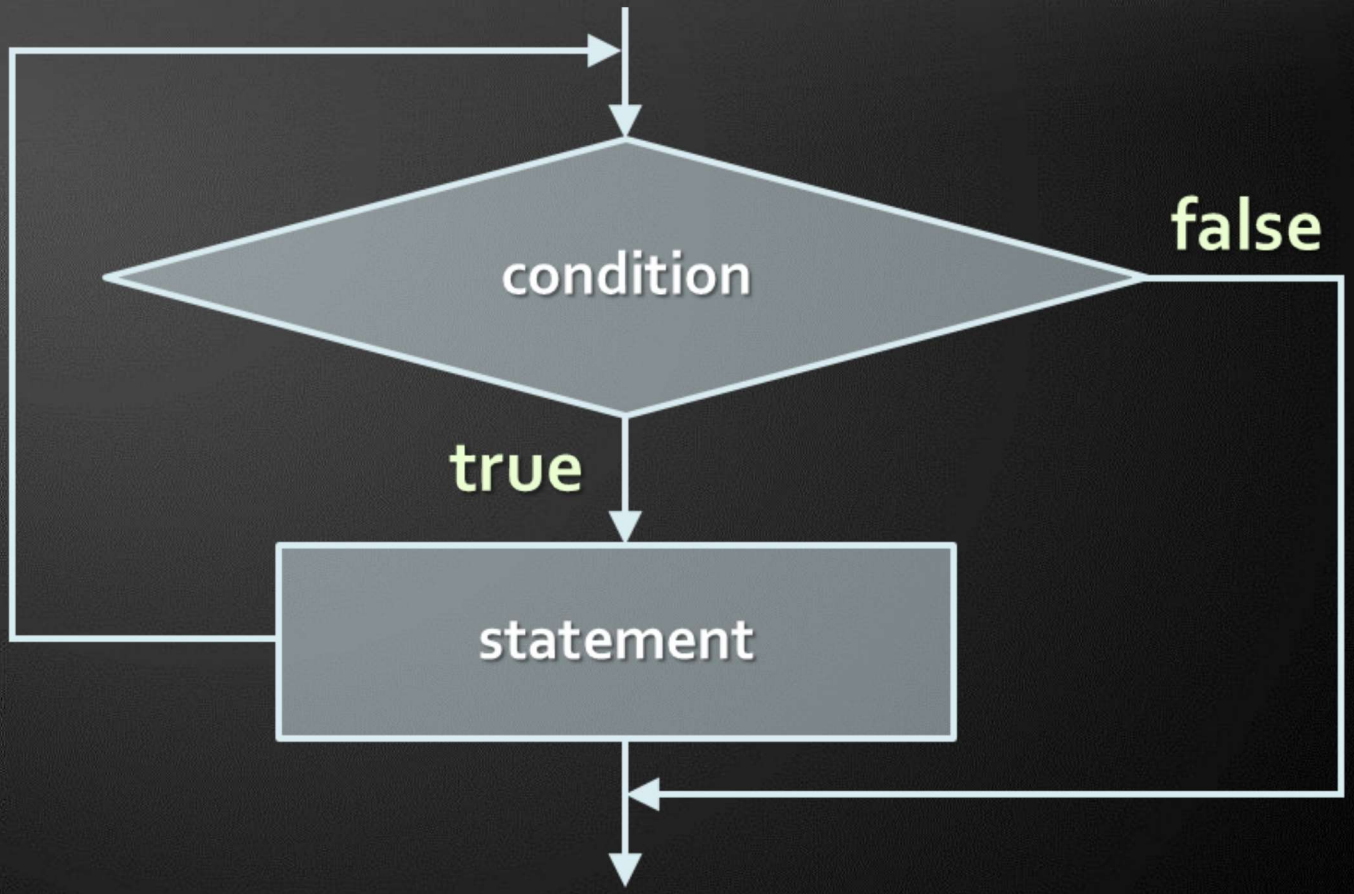
```
while (condition)
{
    statements;
}
```

- The repeat condition
  - Returns a boolean result of `true` or `false`
  - Also called `loop condition`





# Telerik Academy `while` loop – How it works?



Telerik Academy

# while loop – *Example*

```
int counter = 0;
while (counter < 10)
{
    Console.WriteLine("Number : {0}", counter);
    counter++;
}
```

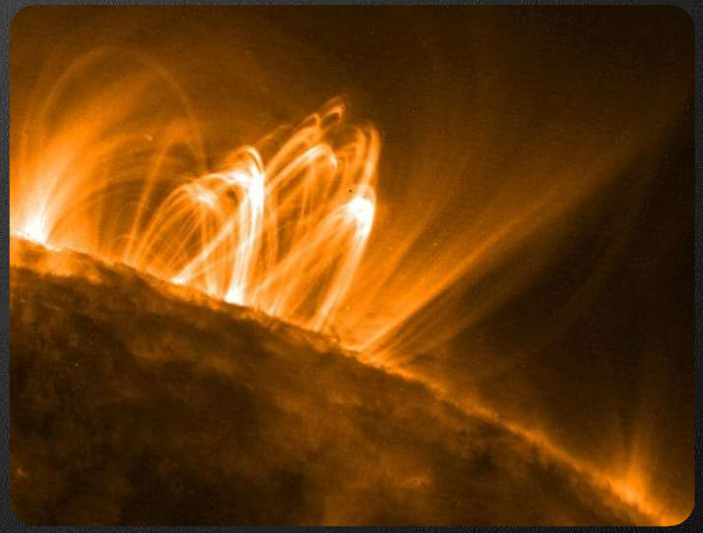
```
Number : 0
Number : 1
Number : 2
Number : 3
Number : 4
Number : 5
Number : 6
Number : 7
Number : 8
Number : 9
Press any key to continue_
```





Telerik Academy

# While

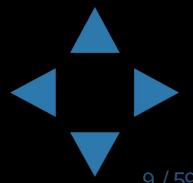


Telerik Academy

# Sum 1..N – *Example*

- Calculate and print the sum of the first N natural numbers

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
int number = 1;
int sum = 1;
Console.Write("The sum 1");
while (number < n)
{
    number++;
    sum += number ;
    Console.Write("+{0}", number);
}
Console.WriteLine(" = {0}", sum);
```







Telerik Academy

# Calculating Sum 1..N

## Demo

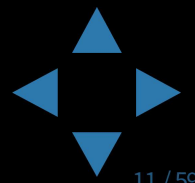
Telerik Academy

# Prime Number – *Example*

- Checking whether a number is prime or not

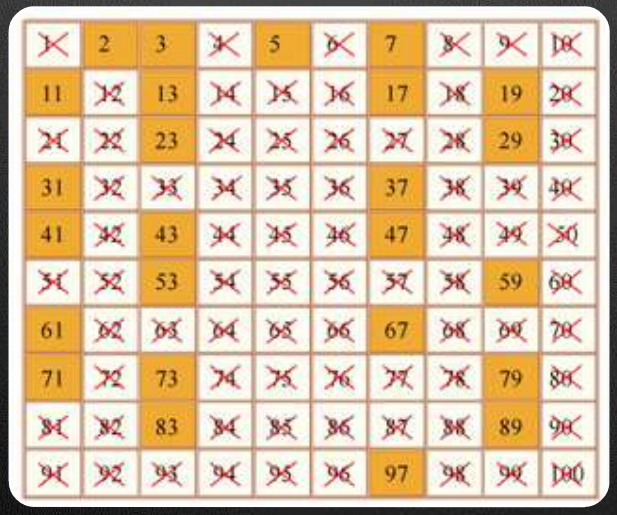
```
Console.Write("Enter a positive integer number: ");
string consoleArgument=Console.ReadLine();
uint number = uint.Parse(consoleArgument);
uint divider = 2;
uint maxDivider = (uint) Math.Sqrt(number);
bool prime = true;
while (prime && (divider <= maxDivider))
{
    if (number % divider == 0)
    {
        prime = false;
    }
    divider++;
}
Console.WriteLine("Prime? {0}", prime);
```





Telerik Academy

# Checking Whether a Number Is Prime




Demo

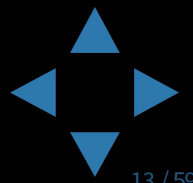
Telerik Academy

# Using break operator

- break operator exits the inner-most loop

```
static void Main()
{
    int n = Convert.ToInt32(Console.ReadLine());
    // Calculate n! = 1 * 2 * ... * n
    int result = 1;
    while (true)
    {
        if (n == 1)
            break;
        result *= n;
        n--;
    }
    Console.WriteLine("n! = " + result);
}
```

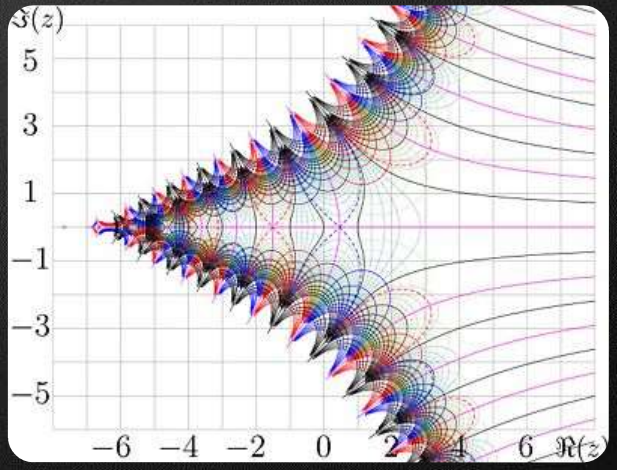




Telerik Academy

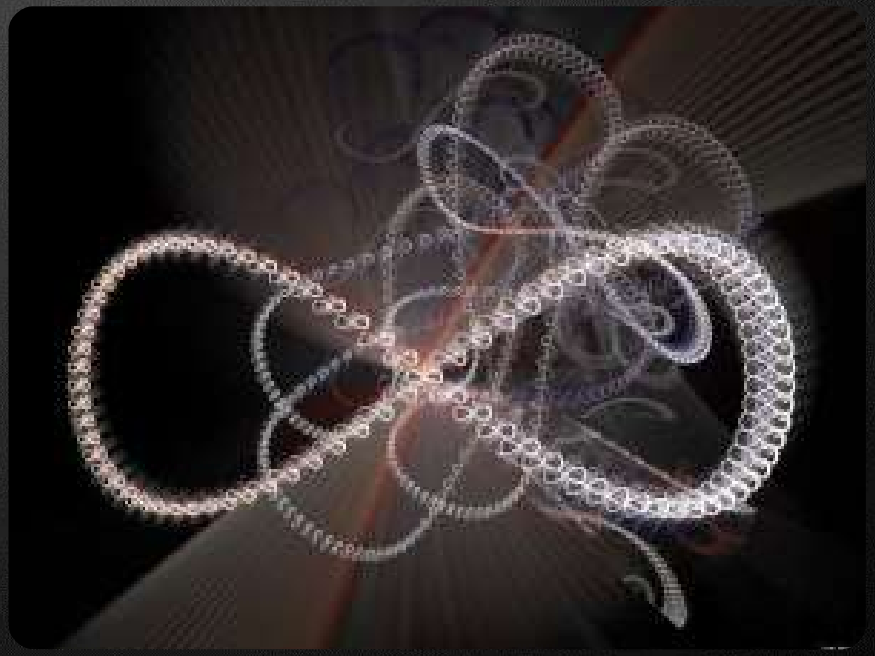
# Calculating Factorial

## Demo



Telerik Academy

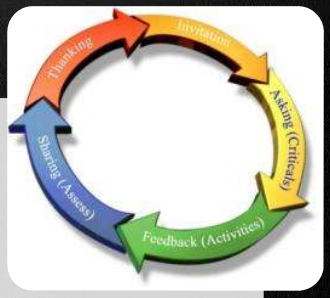
# do { … } while (…) loop



Telerik Academy

# Using do-while loop

- Another loop structure is:

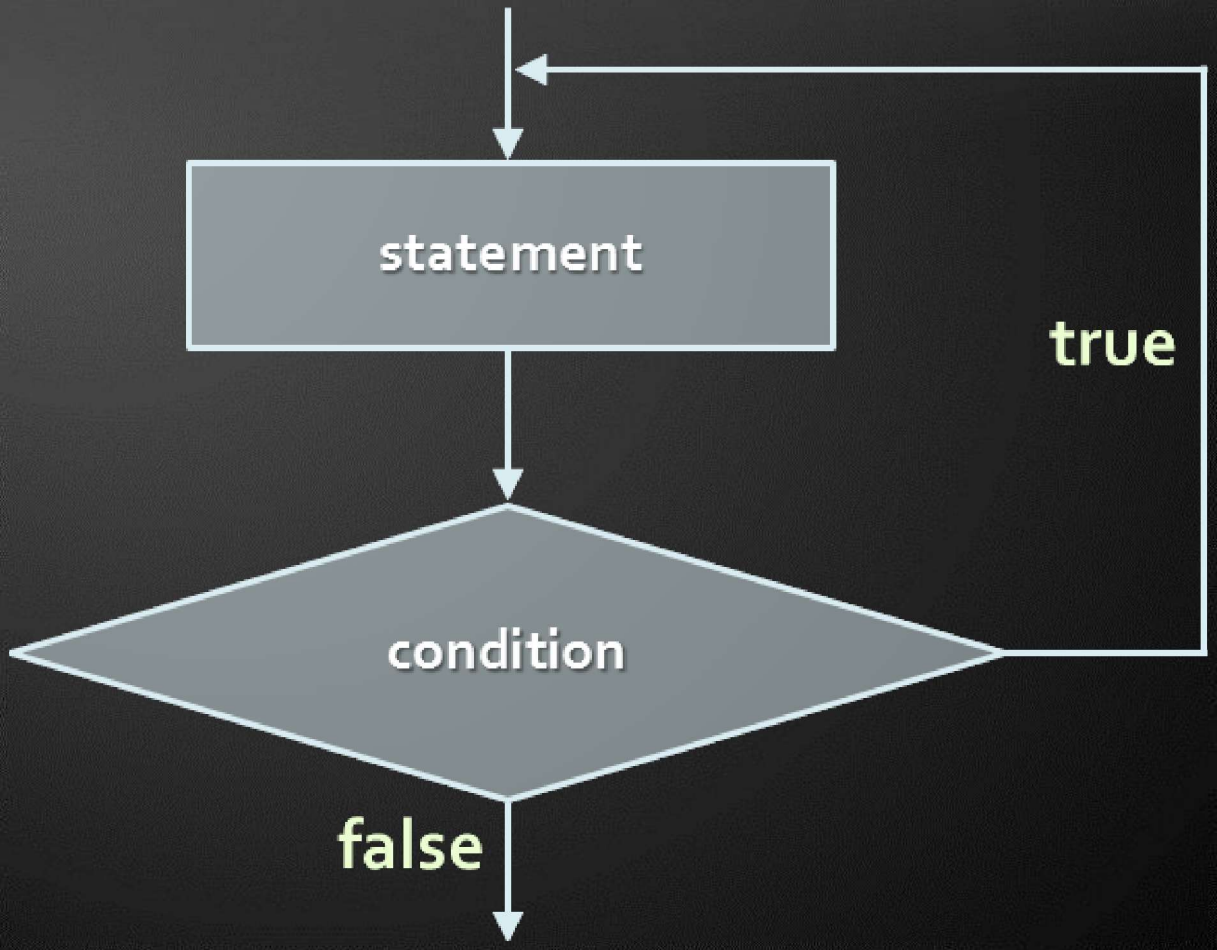



```
do
{
    statements;
}
while (condition);
```

- The block of statements is repeated
  - While the boolean loop condition holds
- The loop is executed at least once

Telerik Academy

# do-while statement



Telerik Academy

# do { … } while

*Examples*

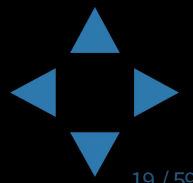
Telerik Academy

# Factorial – *Example*

- Calculating N factorial

```
static void Main()
{
    string numberAsString = Console.ReadLine();
    int n = Convert.ToInt32(numberAsString);
    int factorial = 1;

    do
    {
        factorial *= n;
        n--;
    }
    while (n > 0);

    Console.WriteLine("n! = " + factorial);
}
```





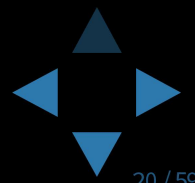
Telerik Academy

# Factorial with BigInteger – *Example*

- Calculating N factorial with `BigInteger`

Don't forget to add reference to System.Numerics.dll.

```
using System.Numerics;
static void Main()
{
    int n = 1000;
    BigInteger factorial = 1;
    do
    {
        factorial *= n;
        n--;
    }
    while (n > 0);
    Console.WriteLine("n! = " + factorial);
}
```







Telerik Academy

# Factorial (do ... while)

Demo



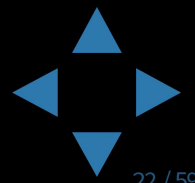
Telerik Academy

# Product[N..M] – *Example*

- Calculating the product of all numbers in the interval [n..m]:

```
int n = int.Parse(Console.ReadLine());
int m = int.Parse(Console.ReadLine());
int number = n;
decimal product = 1;

do
{
    product *= number;
    number++;
}
while (number <= m);

Console.WriteLine("product[n..m] = " + product);
```





Telerik Academy

# Product of the Numbers in the Interval [n..m]

Demo


Telerik Academy

# for Loops

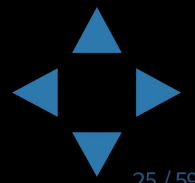
Telerik Academy

# for Loops

- The typical `for` loop syntax is:

```
for (initialization; test; update)
{
    statements;
}
```

- Consists of
  - `Initialization` statement
  - `Boolean test` expression
  - `Update` statement
  - Loop body block



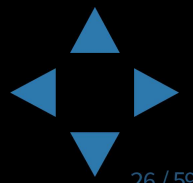
Telerik Academy

# The Initialization Expression

```
for (int number = 0; ...; ...)
{
    // Can use number here
}
// Cannot use number here
```

- Executed once, just before the loop is entered
  - Like it is out of the loop, before it
- Usually used to declare a counter variable





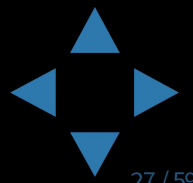
Telerik Academy

# The Test Expression

```
for (int number = 0; number < 10; ...)
{
    // Can use number here
}
// Cannot use number here
```

- Evaluated before each iteration of the loop
  - If `true`, the loop body is executed
  - If `false`, the loop body is skipped
- Used as a `loop condition`





Telerik Academy

# The Update Expression

```
for (int number = 0; number < 10; number++)
{
        // Can use number here
}
// Cannot use number here
```

- Executed at each iteration after the body of the loop is finished
- Usually used to update the counter

Telerik Academy

# for Loop

Telerik Academy

# Simple for Loop – *Example*

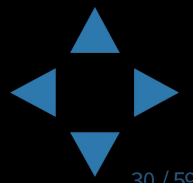
- A simple for-loop to print the numbers 0…9:

```
for (int number = 0; number < 10; number++)
{
    Console.Write(number + " ");
}
```

- A simple for-loop to calculate n!:

```
decimal factorial = 1;
for (int i = 1; i <= n; i++)
{
    factorial *= i;
}
```





# Complex for Loop – *Example*

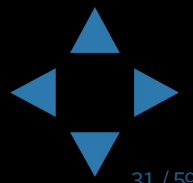
- Complex for-loops could have several counter variables:

```
for (int i=1, sum=1; i <= 128; i = i*2, sum += i)
{
    Console.WriteLine("i={0}, sum={1}", i, sum);
}
```

- Result:

```
i=1, sum=1
i=2, sum=3
i=4, sum=7
i=8, sum=15
...
```





Telerik Academy

# For Loops

## Demo

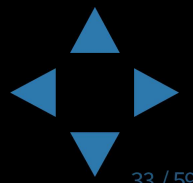
Telerik Academy

# N^M – *Example*

- Calculating n to power m (denoted as n^m):

```
static void Main()
{
    int n = int.Parse(Console.ReadLine());
    int m = int.Parse(Console.ReadLine());
    decimal result = 1;

    for (int i=0; i<m; i++)
    {
        result *= n;
    }

    Console.WriteLine("n^m = " + result);
}
```



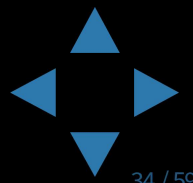


Telerik Academy

# Calculating N^M

Demo





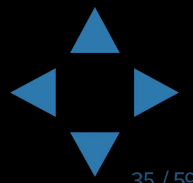
Telerik Academy

# Using a complex for loop – *Example*

- Drawing a diagonal line on the console

```
static void Main()
{
    for (int x = 0, y = 0; x < 10; x++, y++)
    {
        Console.SetCursorPosition(x, y);
        Console.Write('*');
    }
}
```



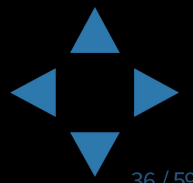


Telerik Academy

# Diagonal Line

Demo





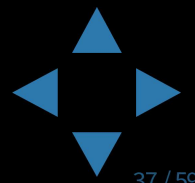
Telerik Academy

# Using continue Operator

- `continue` operator ends the iteration of the inner-most loop
- *Example*: sum all odd numbers in [1, n] that are not divisors of 7:

```
int n = int.Parse(Console.ReadLine());
int sum = 0;
for (int i = 1; i <= n; i += 2)
{
   if (i % 7 == 0)
   {
      continue;
   }
   sum += i;
}
Console.WriteLine("sum = {0}", sum);
```





Telerik Academy

# Using continue Operator

Demo

Telerik Academy

# foreach Loop

## iteretion over a collection of elements



Telerik Academy

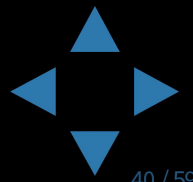
# Foreach Loops

- The typical `foreach` loop syntax is:

```
foreach (Type element in collection)
{
    statements;
}
```

- Iterates over all elements of a collection
  - The `element` is the loop variable that takes sequentially all collection values
  - The `collection` can be list, array or other group of elements of the same type





Telerik Academy

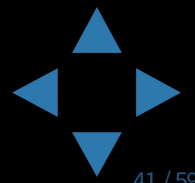
# foreach Loop – *Example*

- *Example* of `foreach` loop:

```
string[] days = {
  "Monday", "Tuesday", "Wednesday", "Thursday",
  "Friday", "Saturday", "Sunday" };
foreach (string day in days)
{
  Console.WriteLine(day);
}
```

- The above loop iterates of the array of days
  - The variable `day` takes all its values
- In the `foreach` loop we cannot set the value of the current item





Telerik Academy

# foreach Loop

Demo



Telerik Academy

# Nested Loops

## Using loops inside a loop

Telerik Academy

# What Is Nested Loop?

- A composition of loops is called a `nested loop`
  - A loop inside another loop
- *Example*:

```
for (initialization; test; update)
{
    for (initialization; test; update)
    {
        statements;
    }
    …
}
```



Telerik Academy

# Nested Loops

*Examples*

Telerik Academy

# Triangle – *Example*

- Print the following triangle:

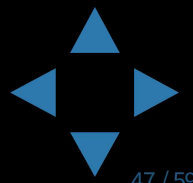
```
1

1 2

...

1 2 3 ... n
```

```
int n = int.Parse(Console.ReadLine());
for(int row = 1; row <= n; row++)
{
   for(int column = 1; column <= row; column++)
   {
      Console.Write("{0} ", column);
   }
   Console.WriteLine();
}
```

Telerik Academy

# Triangle

Demo

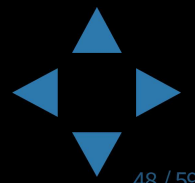


Telerik Academy

# Primes[N, M] – *Example*

- Print all prime numbers in the interval [n, m]:

```
int n = int.Parse(Console.ReadLine());
int m = int.Parse(Console.ReadLine());
for (int number = n; number <= m; number++)
{
   bool prime = true;
   int divider = 2;
   int maxDivider = Math.Sqrt(num);
   while (divider <= maxDivider)
   {
      if (number % divider == 0)
      {
         prime = false;
         break;
      }
      divider++;
   }
   if (prime)
   {
      Console.Write("{0} ", number);
   }
}
```





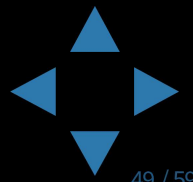


Telerik Academy

# Primes in Range [n, m]

Demo



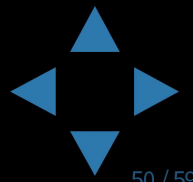


Telerik Academy

# C# Jump Statements

- Jump statements are:
  - `break`, `continue`, `goto`
- How `continue` works?
  - In `while` and `do-while` loops jumps to the test expression
  - In `for` loops jumps to the update expression
- To exit an inner loop use `break`
- To exit outer loops use `goto` with a label
  - Avoid using `goto`! (it is considered harmful)





Telerik Academy

# C# Jump Statements – *Example*

```
int outerCounter = 0;
for (int outer = 0; outer < 10; outer++)
{
    for (int inner = 0; inner < 10; inner++)
    {
        if (inner % 3 == 0)
        {
            continue;
        }
        if (outer == 7)
        {
            break;
        }
        if (inner + outer > 9)
        {
            goto breakOut;
        }
    }
    outerCounter++;
}
breakOut:
```

Label



Telerik Academy

# Loops – More *Examples*

Telerik Academy

# Nested Loops – *Examples*

- Print all four digit numbers in format ABCD such that A+B = C+D (known as happy numbers)

```
static void Main()
{
  for (int a =1 ; a <= 9; a++)
    for (int b = 0; b <= 9; b++)
      for (int c = 0; c <= 9; c++)
        for (int d = 0; d <= 9; d++)
          if (a + b == c + d)
            Console.WriteLine("{0}{1}{2}{3}",
              a, b, c, d);
}
```

Can you improve this algorithm to use 3 loops only?

Telerik Academy

# Happy Numbers

Demo

Telerik Academy

# Nested Loops – *Examples*

- Print all combinations from TOTO 6/49

```
static void Main()
{
  int i1, i2, i3, i4, i5, i6;
  for (i1 = 1; i1 <= 44; i1++)
    for (i2 = i1 + 1; i2 <= 45; i2++)
      for (i3 = i2 + 1; i3 <= 46; i3++)
        for (i4 = i3 + 1; i4 <= 47; i4++)
          for (i5 = i4 + 1; i5 <= 48; i5++)
            for (i6 = i5 + 1; i6 <= 49; i6++)
              Console.WriteLine("{0} {1} {2} {3} {4} {5}",
                i1, i2, i3, i4, i5, i6);
}
```

Warning: execution of this code could take too long time.

Telerik Academy

# TOTO 6/49

## Demo

Telerik Academy

# Summary

- C# supports four types of loops:
  - `while`
  - `do-while`
  - `for` loops
  - `foreach` loops
- Nested loops can be used to implement more complex logic
- The operators `continue`, `break` & `goto` can control the loop execution

Telerik Academy

# Loops

## Questions?



Telerik Academy

# Free Trainings @ Telerik Academy

- Fundamentals of C# ProgrammingTrack of Courses
  - csharpfundamentals.telerik.com
    - Telerik Software Academy
  - academy.telerik.com
    - Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy
    - Telerik Academy Learning System
  - telerikacademy.com