Loops

Execute Blocks of Code Multiple Times

Svetlin Nakov

Telerik Corporation

www.telerik.com

- **What is a Loop?**
- **Loops in C#**
    - **while loops**
    - **do ... while loops**
    - **for loops**
    - **foreach loops**
- **Nested loops**

- 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

Using `while(…)` Loop
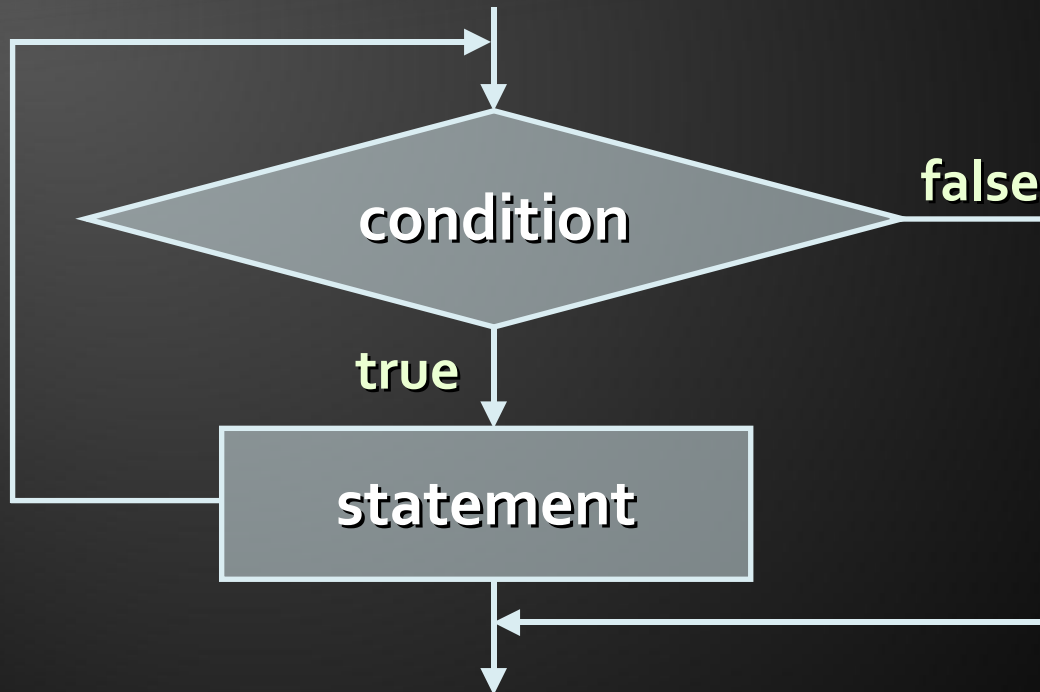
Repeating a Statement While Given Condition Holds
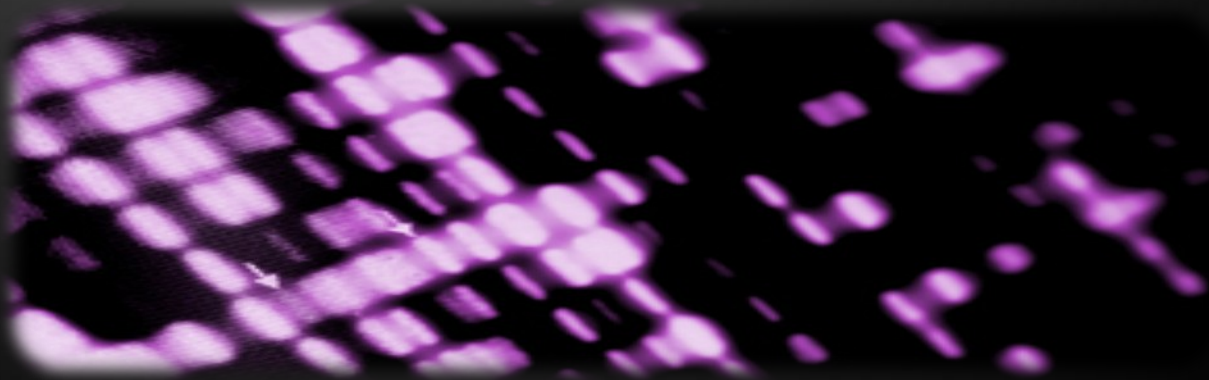
- **The simplest and most frequently used loop**

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

- **The repeat condition**

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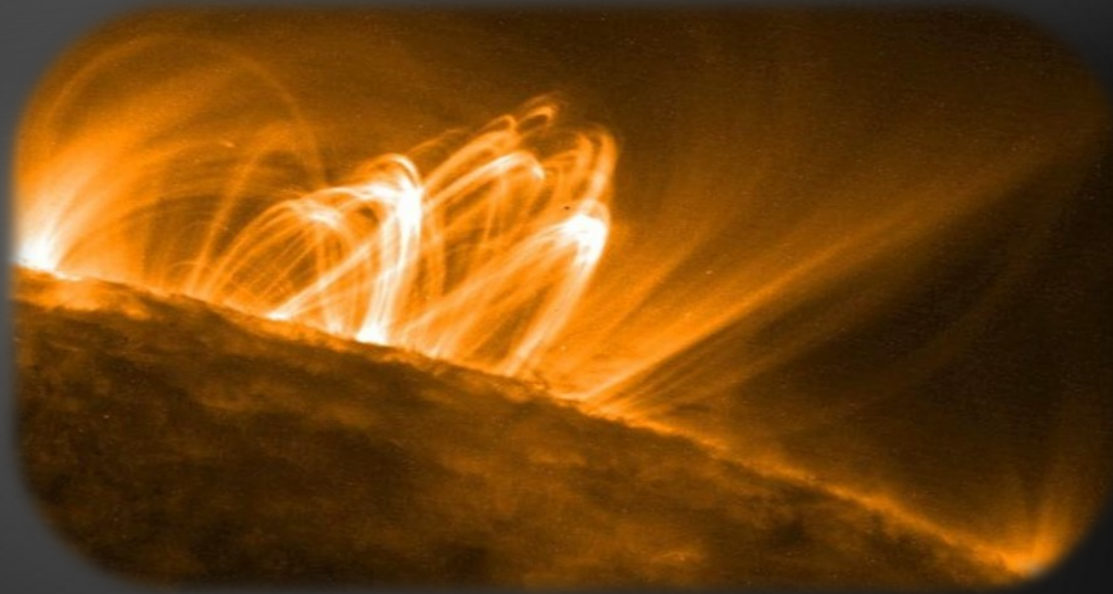
  - **Also called loop condition**

```
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_
```

while(…)

Examples

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

```csharp
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);
```

Calculating Sum 1..N

Live Demo

- **Checking whether a number is prime or not**

```
Console.Write("Enter a positive integer number: ");
uint number = uint.Parse(Console.ReadLine());
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);
```

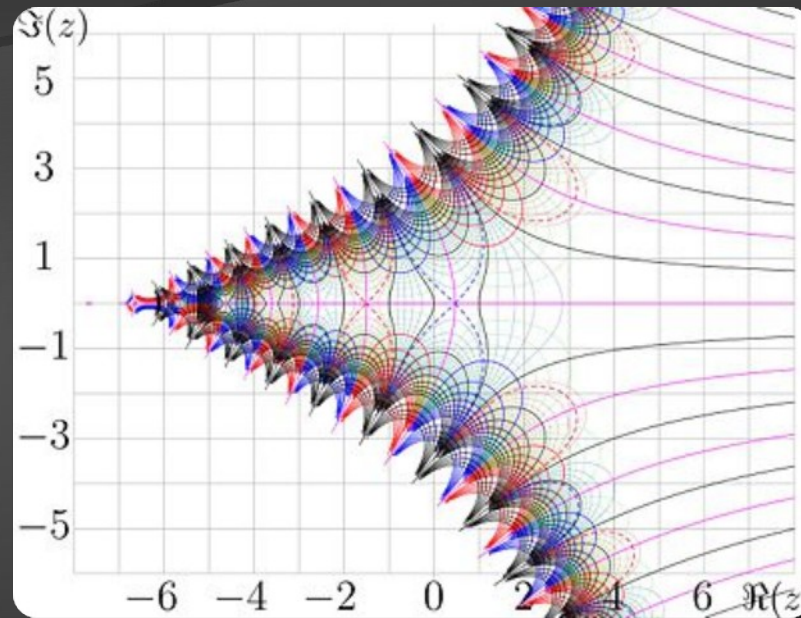# Checking Whether a Number Is Prime

Live Demo

- **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);
}
```

Calculating Factorial

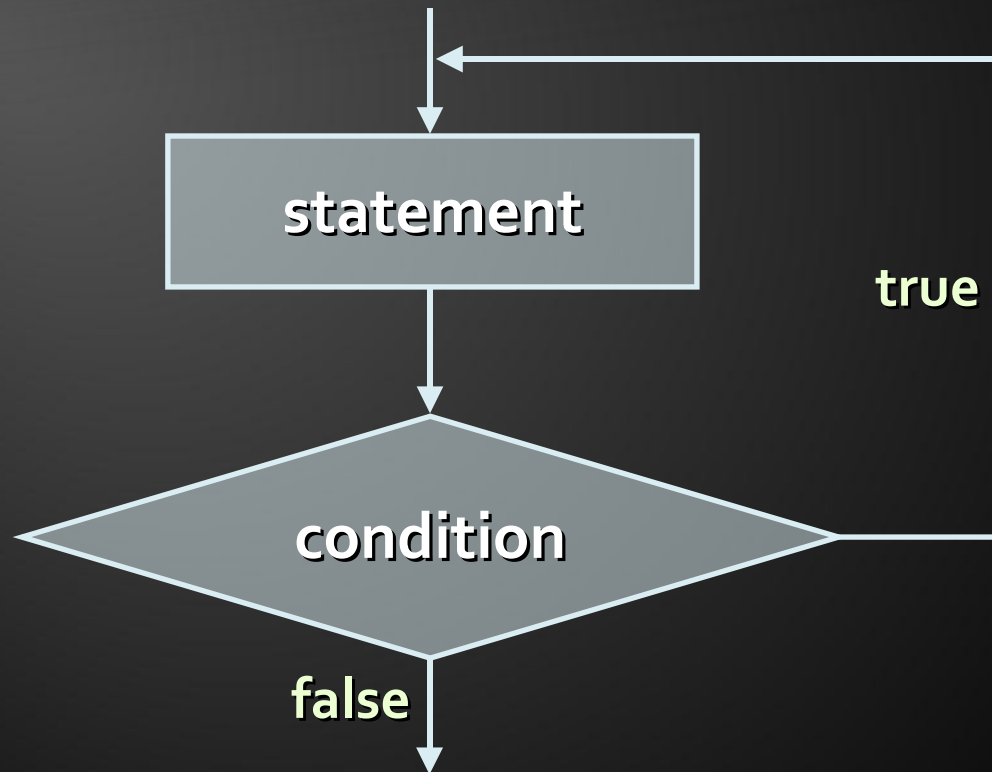## Live Demo

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**

do { … }
while (…)
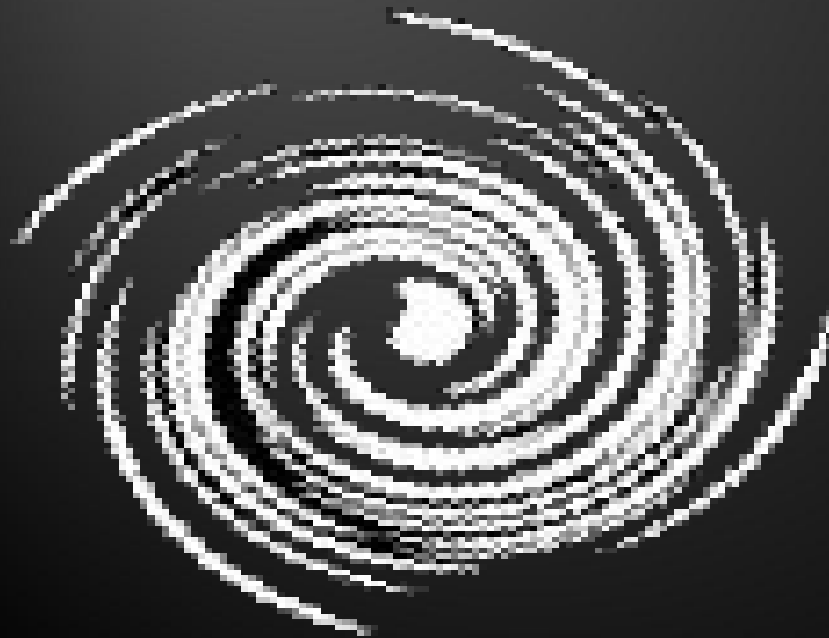
Examples

- **Calculating N factorial**

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

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

    Console.WriteLine("n! = " + factorial);
}
```
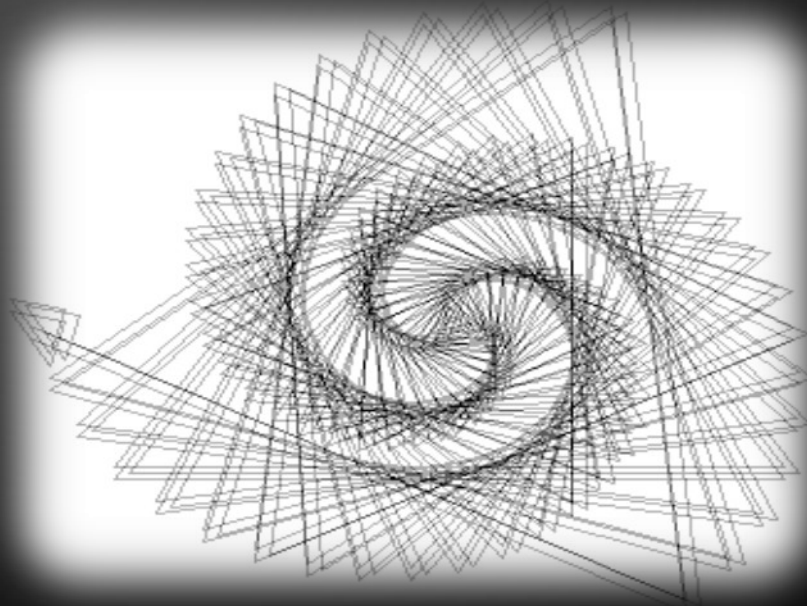
- **Calculating N factorial with `BigInteger`**

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

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

Factorial (do ... while)

Live Demo

- 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);
```

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

Live Demo

for Loops

telerik
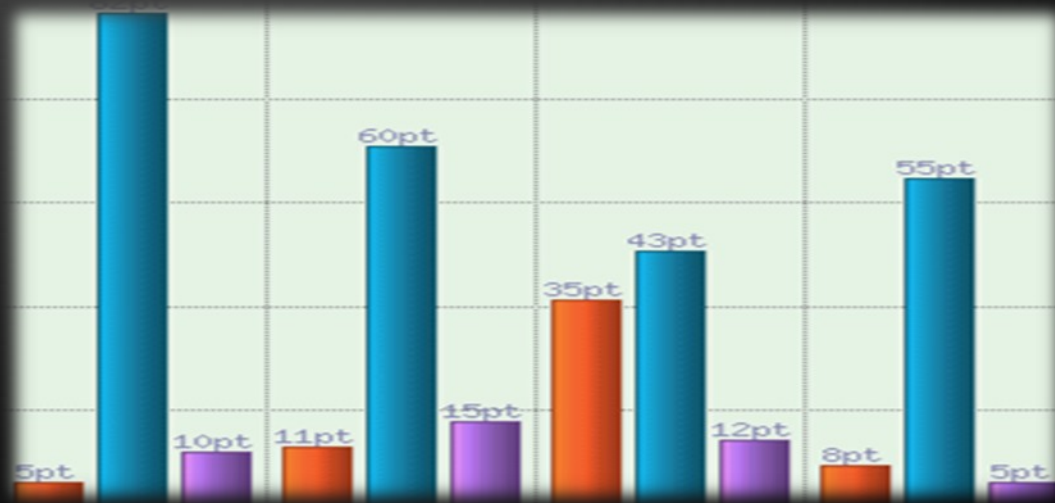
- **The typical for loop syntax is:**

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

- **Consists of**

  - **Initialization statement**

  - **Boolean test expression**

  - **Update statement**

  - **Loop body block**

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

```
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**

```
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

for Loop

Examples

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
...
```

For Loops

Live Demo

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

```csharp
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);
}
```

Calculating N^M

Live Demo

- **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);
```

foreach Loop

Iteration over a Collection
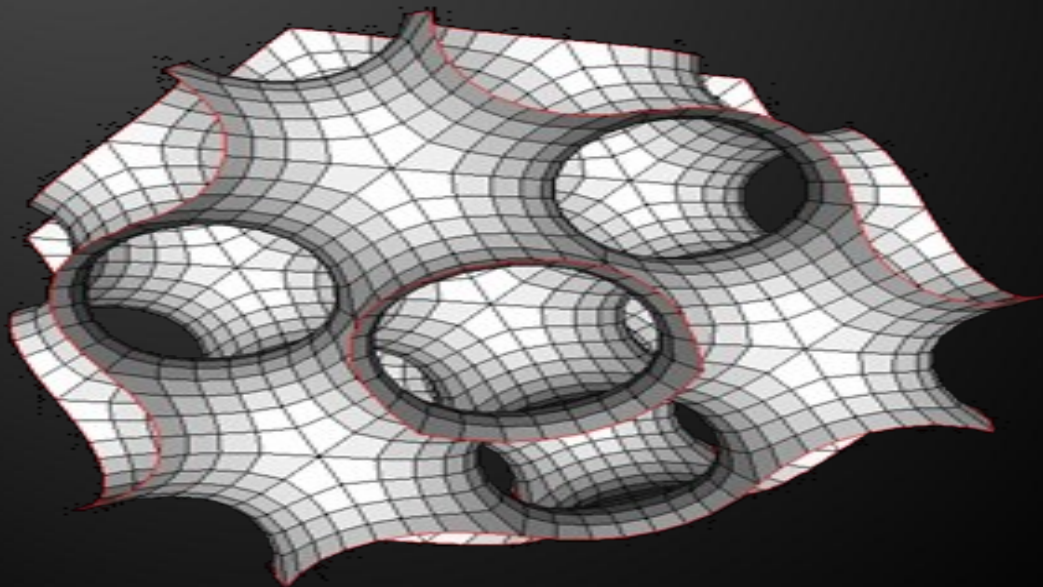
* **The typical `foreach` loop syntax is:**

```
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**

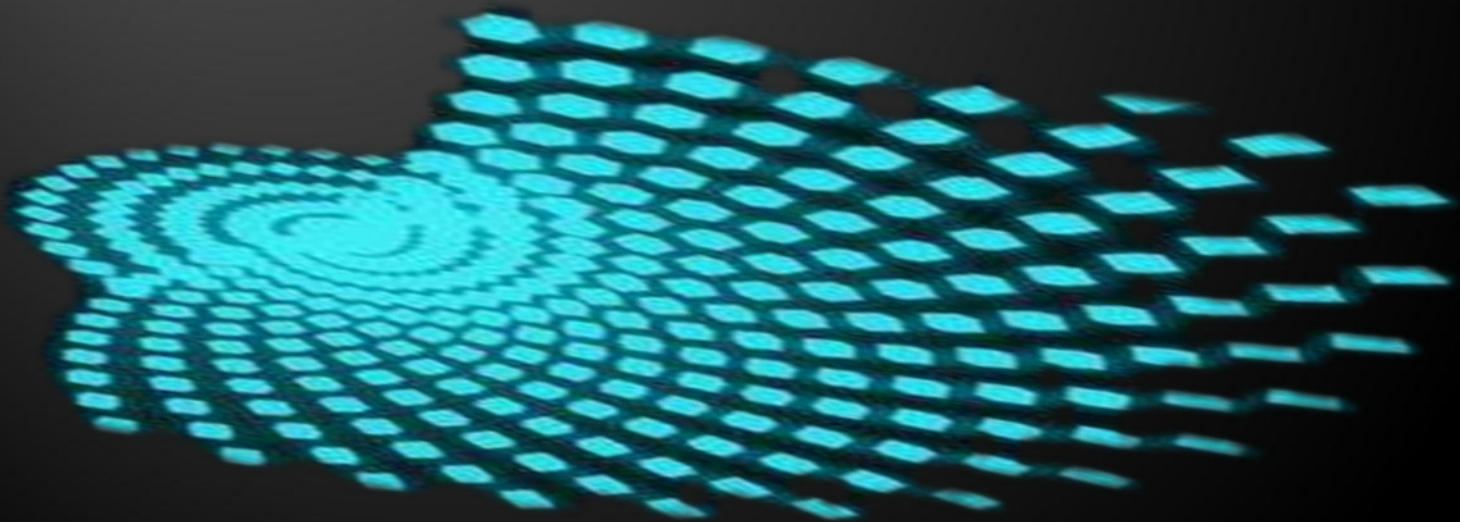- **Example of foreach loop:**

```
string[] days = new string[] {
   "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**

foreach Loop

Live Demo

Nested Loops

Using Loops Inside a 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;
    }
    …
}
```

Nested Loops

Examples

- **Print the following triangle:**

```
1

1 2

…

1 2 3 ... n
```

```csharp
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();
}
```

Triangle

Live Demo

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

```csharp
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);
    }
}
```
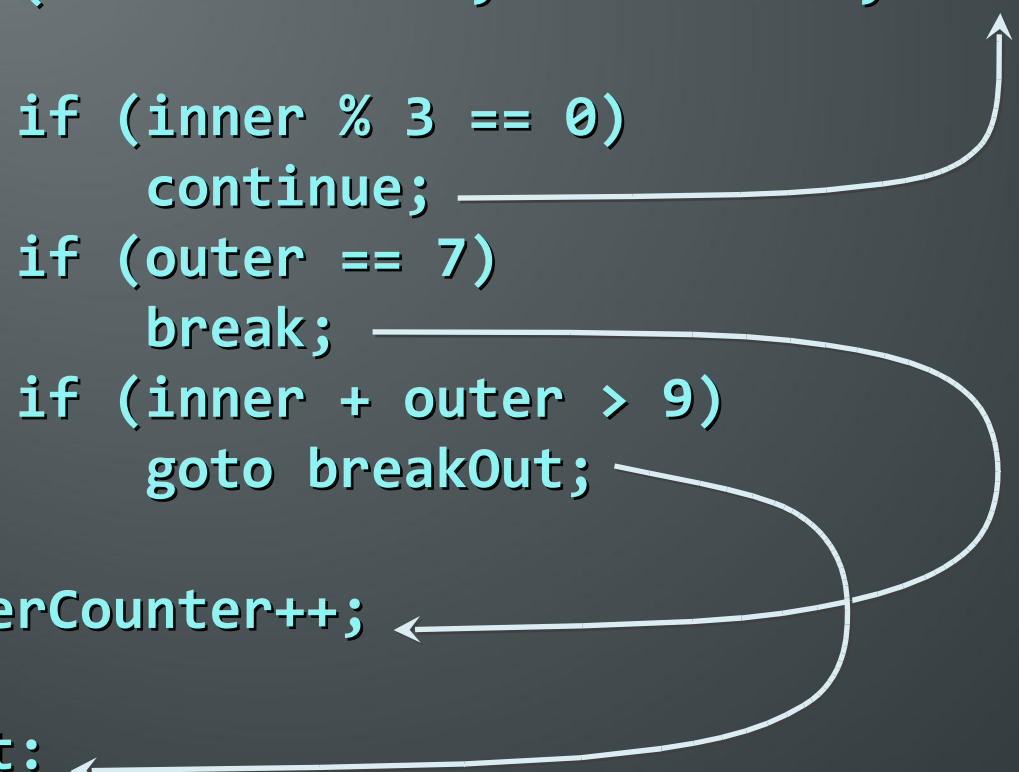
Primes in Range [n, m]

Live Demo

telerik

- Jump statements are:
  - `break`, `continue`, `goto`
- How `continue` woks?
  - 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)
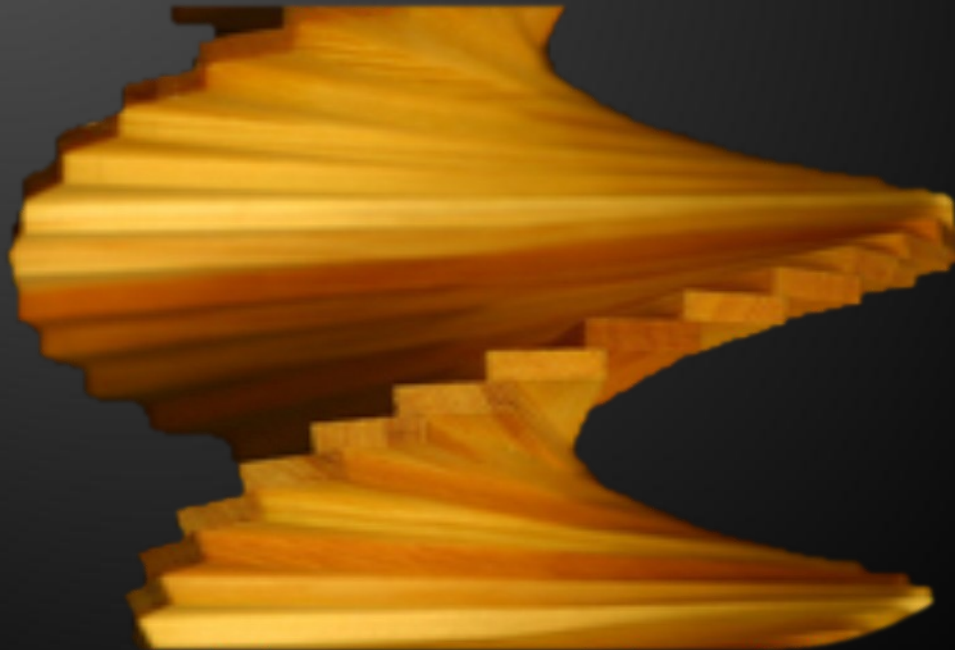
C# Jump Statements – Example

```
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

Loops – More Examples

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

```csharp
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?

- 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.
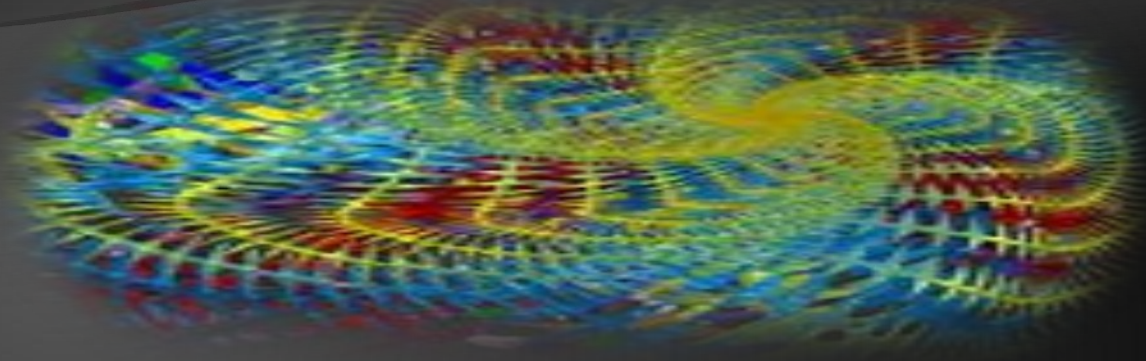
TOTO 6/49

Live Demo

* 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

Questions?

1. Write a program that prints all the numbers from 1 to N.

2. Write a program that prints all the numbers from 1 to N, that are not divisible by 3 and 7 at the same time.

3. Write a program that reads from the console a sequence of N integer numbers and returns the minimal and maximal of them.

4. Write a program that calculates N!/K! for given N and K (1<N<K).

5. Write a program that calculates N!*K! / (K-N)! for given N and K (1<N<K).

1. Write a program that, for a given two integer numbers N and X, calculates the sum
$$S = 1 + 1!/X + 2!/X^2 + \ldots + N!/X^N$$

2. Write a program that reads a number N and calculates the sum of the first N members of the sequence of Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, …

Each member of the Fibonacci sequence (except the first two) is a sum of the previous two members.

4. Write a program that calculates the greatest common divisor (GCD) of given two numbers. Use the Euclidean algorithm (find it in Internet).

1. **In the combinatorial mathematics, the Catalan numbers are calculated by the following formula:**

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!\,n!} \qquad \text{for } n \geq 0.$$

   Write a program to calculate the Nth Catalan number by given N.

5. **Write a program that prints all possible cards from a standard deck of 52 cards (without jokers). The cards should be printed with their English names. Use nested `for` loops and `switch-case`.**

𝕏telerik

1. **Write a program that reads from the console a positive integer number N (N < 20) and outputs a matrix like the following:**

N = 3

1	2	3
2	3	4
3	4	5

N = 4

1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7

telerik

1. *** Write a program that calculates for given N how many trailing zeros present at the end of the number N!. Examples:**

   `N = 10 → N! = 3628800 → 2`

   `N = 20 → N! = 2432902008176640000 → 4`

   Does your program work for N = 50 000?

   Hint: The trailing zeros in N! are equal to the number of its prime divisors of value 5. Think why!

1. * Write a program that reads a positive integer number N (N < 20) from console and outputs in the console the numbers 1 … N numbers arranged as a spiral.

   Example for N = 4

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7