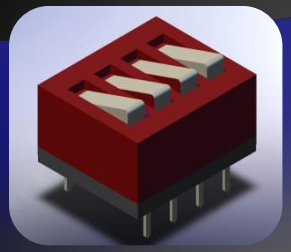
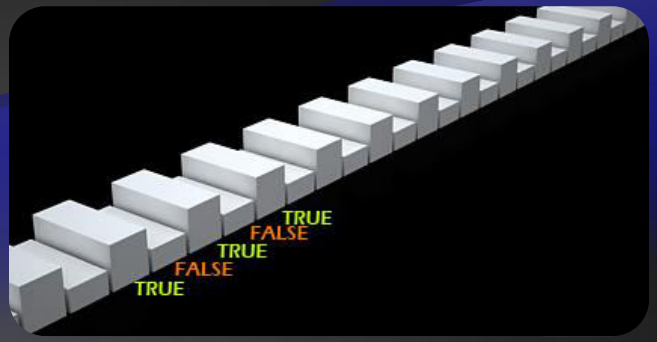
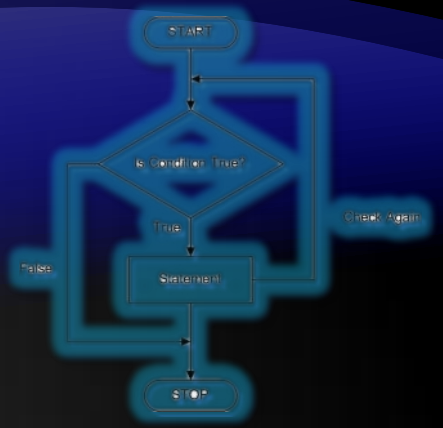
telerik

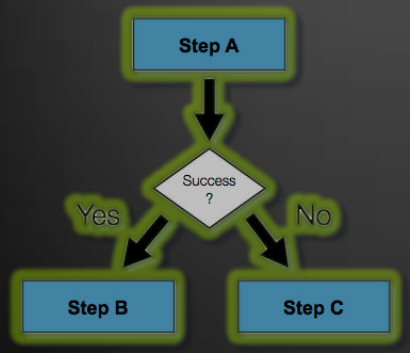





# Conditional Statements

## Implementing Control Logic in C#




**Svetlin Nakov**

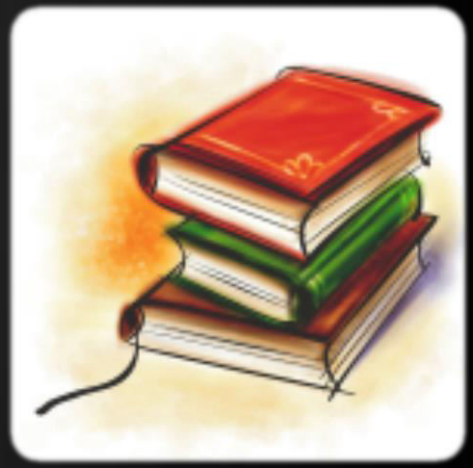
Telerik Corporation

www.telerik.com

telerik

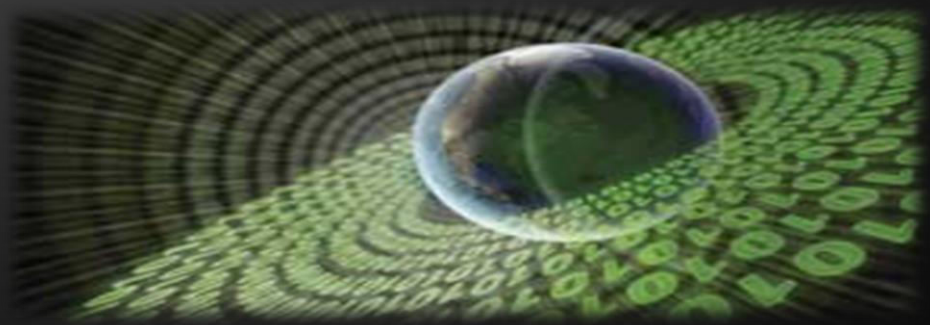
# Table of Contents





telerik

# Comparison and Logical Operators



telerik

# Comparison Operators

| Operator | Notation in C# |
|---|---|
| Equals | == |
| Not Equals | != |
| Greater Than | > |
| Greater Than or Equals | >= |
| Less Than | < |
| Less Than or Equals | <= |

- Example:

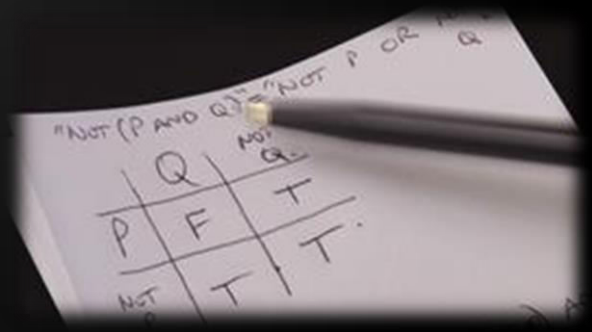
```
bool result = 5 <= 6;
Console.WriteLine(result); // True
```

telerik

# Logical Operators

| Operator | Notation in C# |
|---|---|
| Logical NOT | ! |
| Logical AND | && |
| Logical OR | \|\| |
| Logical Exclusive OR (XOR) | ^ |

- De Morgan laws
  - `!!A ⇔ A`
  - `!(A || B) ⇔ !A && !B`
  - `!(A && B) ⇔ !A || !B`



telerik

# if and if-else

## Implementing Conditional Logic

telerik

# The if Statement

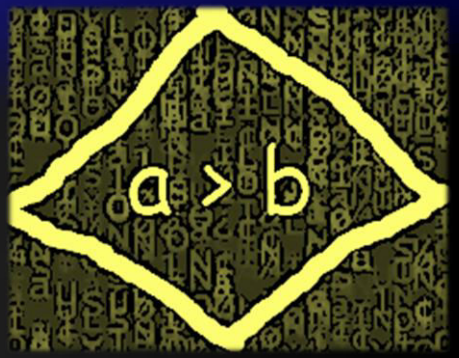
- The most simple conditional statement
- Enables you to test for a condition
- Branch to different parts of the code depending on the result
- The simplest form of an `if` statement:

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

telerik

# Condition and Statement

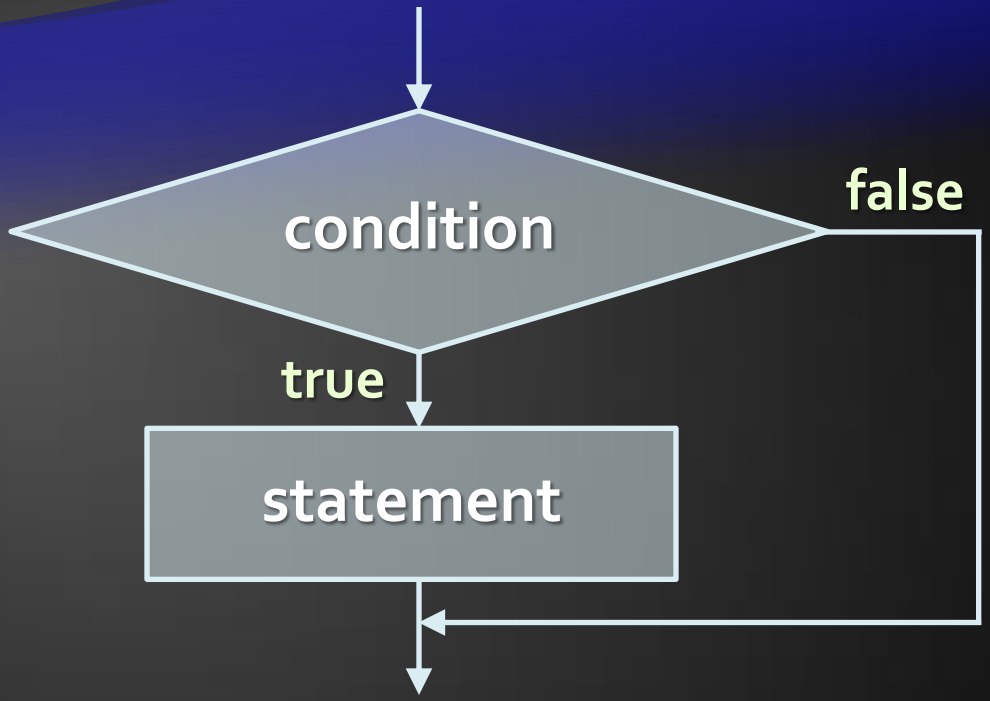
- The condition can be:
  - Boolean variable
  - Boolean logical expression
  - Comparison expression




- The condition cannot be integer variable (like in C / C++)
- The statement can be:
  - Single statement ending with a semicolon
  - Block enclosed in braces

telerik

# How It Works?




- The condition is evaluated
  - If it is true, the statement is executed
  - If it is false, the statement is skipped

telerik

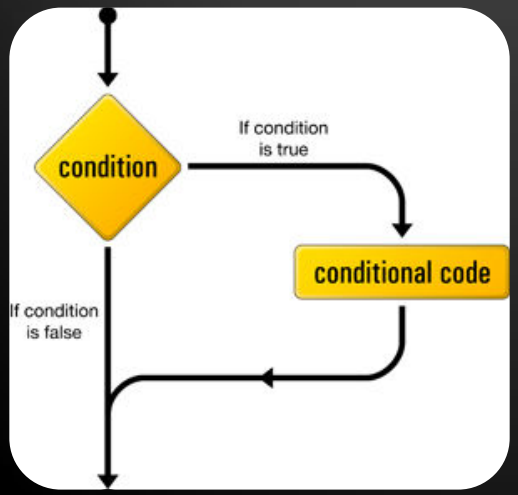
# The if Statement – Example

```
static void Main()
{
    Console.WriteLine("Enter two numbers.");

    int biggerNumber = int.Parse(Console.ReadLine());
    int smallerNumber = int.Parse(Console.ReadLine());

    if (smallerNumber > biggerNumber)
    {
        biggerNumber = smallerNumber;
    }

    Console.WriteLine("The greater number is: {0}",
        biggerNumber);
}
```

telerik

# The if Statement

## Live Demo





telerik

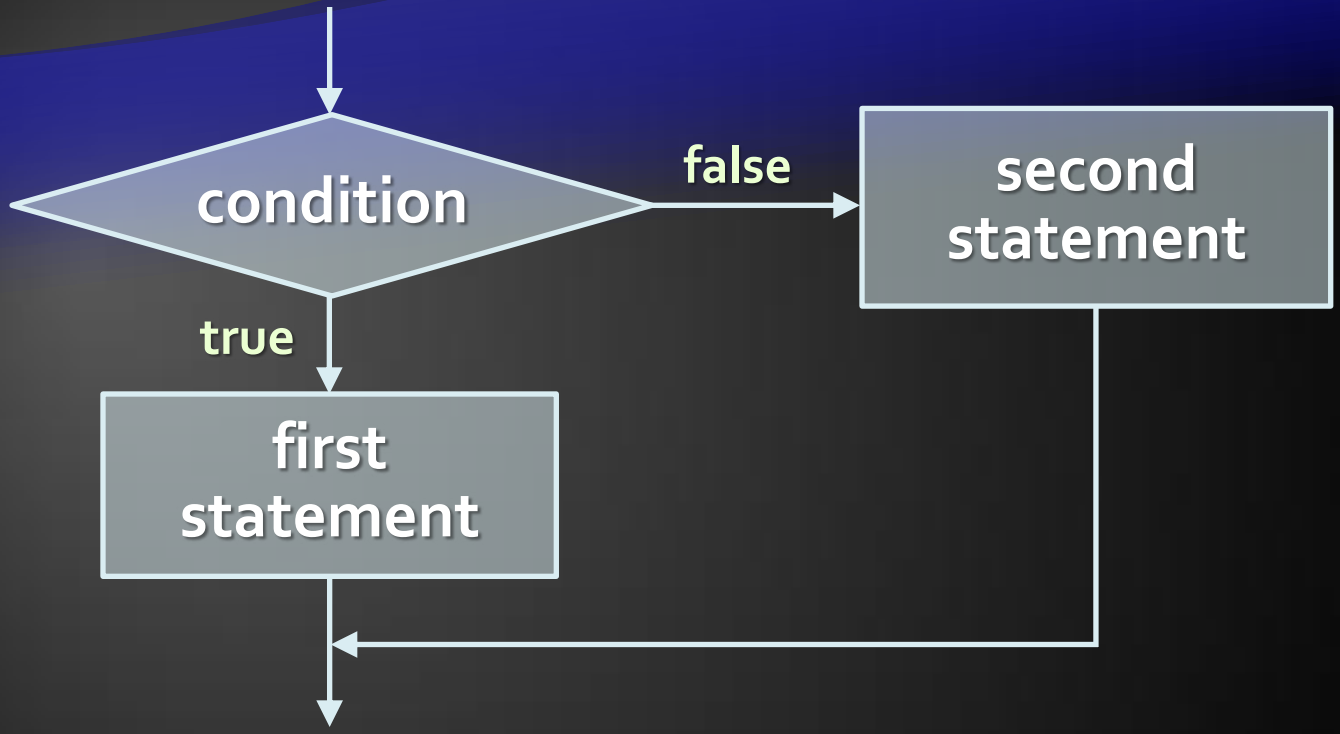
# The if-else Statement

- More complex and useful conditional statement
- Executes one branch if the condition is true, and another if it is false
- The simplest form of an `if-else` statement:

```
if (expression)
{
    statement1;
}
else
{
    statement2;
}
```

telerik

# How It Works ?




- The condition is evaluated
  - If it is true, the first statement is executed
  - If it is false, the second statement is executed

# if-else Statement – Example

- Checking a number if it is odd or even

```
string s = Console.ReadLine();
int number = int.Parse(s);

if (number % 2 == 0)
{
    Console.WriteLine("This number is even.");
}
else
{
    Console.WriteLine("This number is odd.");
}
```

telerik

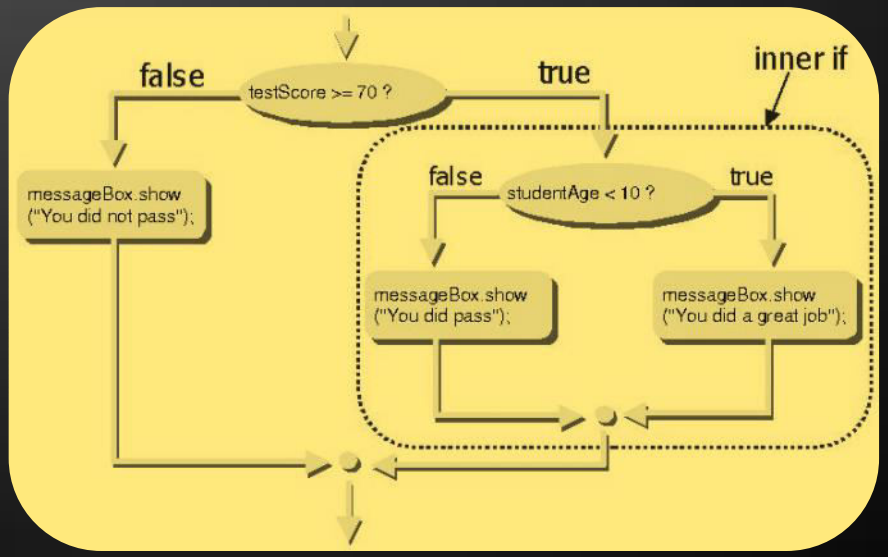
# The if-else Statement

Live Demo

telerik

# Nested if Statements

## Creating More Complex Logic



telerik

# Nested if Statements

- if and if-else statements can be nested, i.e. used inside another if or else statement
- Every else corresponds to its closest preceding if

```
if (expression)
{
    if (expression)
    {
        statement;
    }
    else
    {
        statement;
    }
}
else
    statement;
```

telerik

# Nested if – Good Practices

- Always use { … } blocks to avoid ambiguity
  - Even when a single statement follows
- Avoid using more than three levels of nested `if` statements
- Put the case you normally expect to process first, then write the unusual cases
- Arrange the code to make it more readable

# Nested if Statements – Example

```
if (first == second)
{
    Console.WriteLine(
        "These two numbers are equal.");
}
else
{
    if (first > second)
    {
        Console.WriteLine(
            "The first number is bigger.");
    }
    else
    {
        Console.WriteLine("The second is bigger.");
    }
}
```

telerik


# Nested if Statements

Live Demo

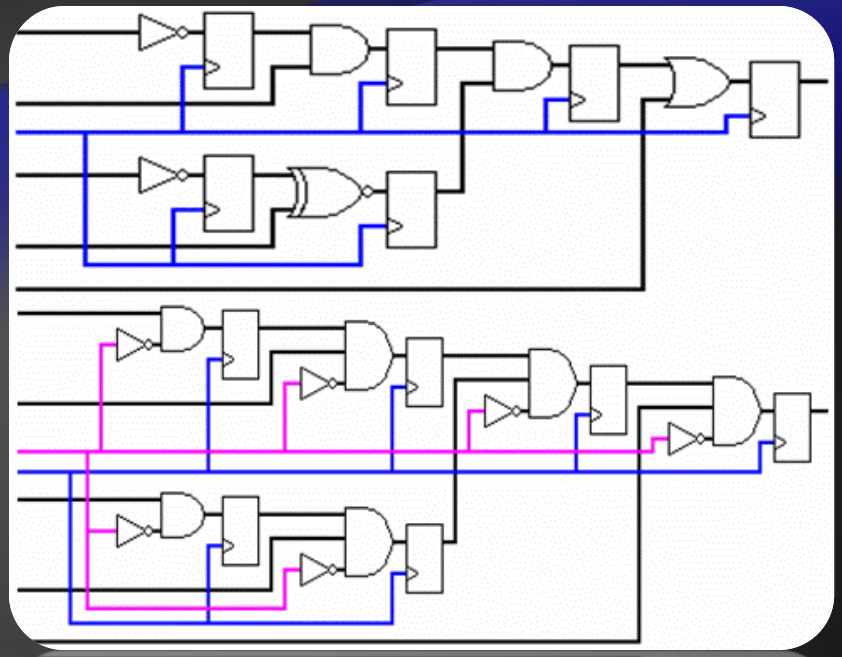
telerik

# Multiple if-else-if-else-…

- Sometimes we need to use another `if`-construction in the `else` block
  - Thus `else if` can be used:

```
int ch = 'X';
if (ch == 'A' || ch == 'a')
{
    Console.WriteLine("Vowel [ei]");
}
else if (ch == 'E' || ch == 'e')
{
    Console.WriteLine("Vowel [i:]");
}
else if  …
else …
```

telerik



# Multiple if-else Statements

Live Demo

telerik

# switch-case

## Making Several Comparisons at Once

telerik

# The switch-case Statement

- Selects for execution a statement from a list depending on the value of the switch expression

```
switch (day)
{
    case 1: Console.WriteLine("Monday"); break;
    case 2: Console.WriteLine("Tuesday"); break;
    case 3: Console.WriteLine("Wednesday"); break;
    case 4: Console.WriteLine("Thursday"); break;
    case 5: Console.WriteLine("Friday"); break;
    case 6: Console.WriteLine("Saturday"); break;
    case 7: Console.WriteLine("Sunday"); break;
    default: Console.WriteLine("Error!"); break;
}
```

telerik

# How switch-case Works?

1. The expression is evaluated
2. When one of the constants specified in a case label is equal to the expression
   - The statement that corresponds to that case is executed
3. If no case is equal to the expression
   - If there is default case, it is executed
   - Otherwise the control is transferred to the end point of the switch statement

telerik

# The switch-case Statement

Live Demo

telerik

# Using switch: Rules

- Variables types like `string`, `enum` and integral types can be used for `switch` expression
- The value `null` is permitted as a case label constant
- The keyword `break` exits the switch statement
- "No fall through" rule – you are obligated to use `break` after each case
- Multiple labels that correspond to the same statement are permitted

telerik

# Multiple Labels – Example

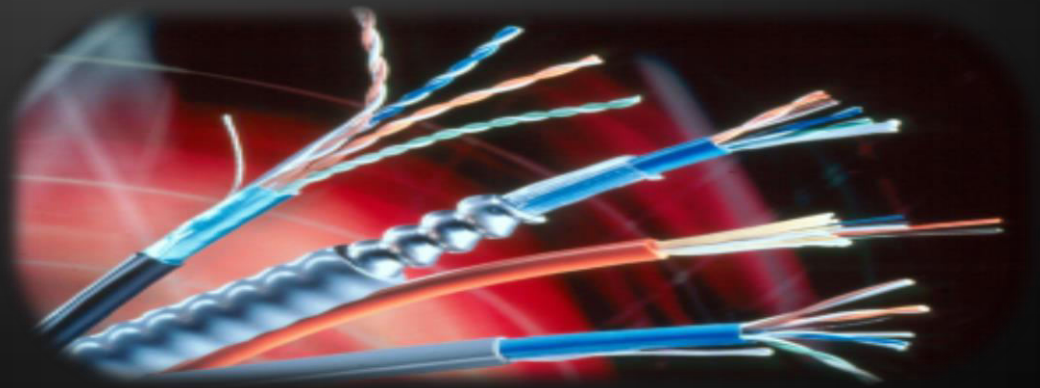
- You can use multiple labels to execute the same statement in more than one case

```
switch (animal)
{
    case "dog" :
        Console.WriteLine("MAMMAL");
        break;
    case "crocodile" :
    case "tortoise" :
    case "snake" :
        Console.WriteLine("REPTILE");
        break;
    default :
        Console.WriteLine("There is no such animal!");
        break;
}
```

telerik

# Multiple Labels in a switch-case

Live Demo



telerik

# Using switch – Good Practices

- There must be a separate `case` for every normal situation
- Put the normal case first
  - Put the most frequently executed cases first and the least frequently executed last
- Order cases alphabetically or numerically
- In `default` use case that cannot be reached under normal circumstances

telerik

# Summary

- Comparison and logical operators are used to compose logical conditions
- The conditional statements `if` and `if-else` provide conditional execution of blocks of code
  - Constantly used in computer programming
  - Conditional statements can be nested
- The `switch` statement easily and elegantly checks an expression for a sequence of values

telerik

# Conditional Statements

# Questions?

telerik

# Exercises

1. Write an `if` statement that examines two integer variables and exchanges their values if the first one is greater than the second one.
2. Write a program that shows the sign (+ or -) of the product of three real numbers without calculating it. Use a sequence of if statements.
3. Write a program that finds the biggest of three integers using nested if statements.
4. Sort 3 real values in descending order using nested if statements.

telerik

# Exercises (2)

5. Write program that asks for a digit and depending on the input shows the name of that digit (in English) using a switch statement.
6. Write a program that enters the coefficients `a`, `b` and `c` of a quadratic equation

   $a*x^2 + b*x + c = 0$

   and calculates and prints its real roots. Note that quadratic equations may have `0`, `1` or `2` real roots.
7. Write a program that finds the greatest of given `5` variables.

telerik

# Exercises (3)

8. Write a program that, depending on the user's choice inputs `int`, `double` or `string` variable. If the variable is integer or double, increases it with `1`. If the variable is string, appends "`*`" at its end. The program must show the value of that variable as a console output. Use `switch` statement.
9. We are given `5` integer numbers. Write a program that checks if the sum of some subset of them is `0`. Example: `3, -2, 1, 1, 8` → `1+1-2=0`.

telerik

# Exercises (4)

10. Write a program that applies bonus scores to given scores in the range [1..9]. The program reads a digit as an input. If the digit is between 1 and 3, the program multiplies it by 10; if it is between 4 and 6, multiplies it by 100; if it is between 7 and 9, multiplies it by 1000. If it is zero or if the value is not a digit, the program must report an error.

Use a `switch` statement and at the end print the calculated new value in the console.

telerik

# Exercises (5)

11. * Write a program that converts a number in the range [0…999] to a text corresponding to its English pronunciation. Examples:

```
0 → "Zero"
273 → "Two hundred seventy three"
400 → "Four hundred"
501 → "Five hundred and one"
711 → "Seven hundred and eleven"
```