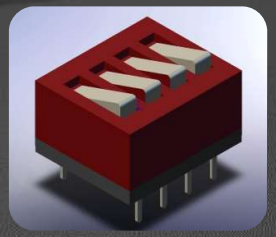
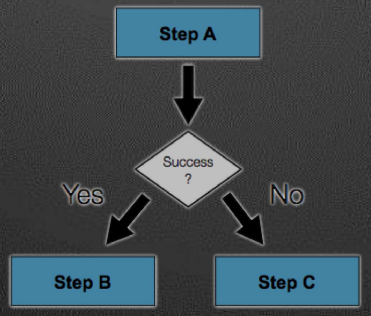
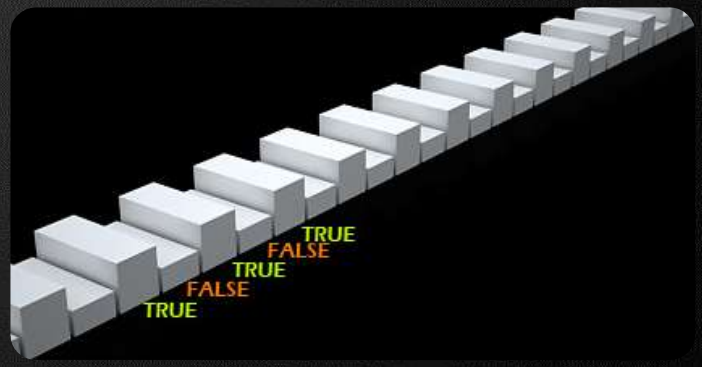
Telerik Academy



Conditional Statements

Implementing Control Logic in C#

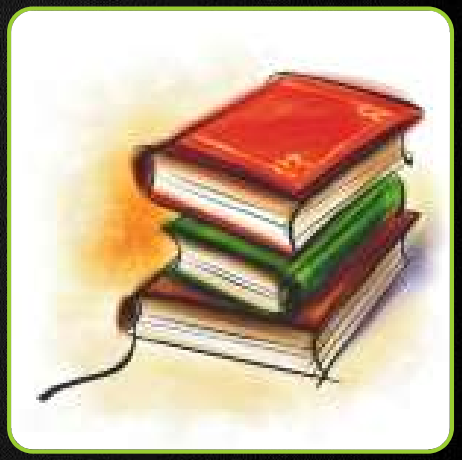




Follow us

Telerik Academy

Table of Contents



Follow us

Telerik Academy

Comparison and Logical Operators

Follow us

Telerik Academy

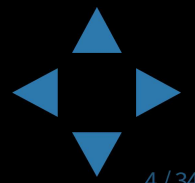
Comparison Operators

Operator	C# Notation
Equals	==
Not Equals	!=
Greater than	>
Greater than or equal	>=
Less than	<
Less than or equal	<=

- *Example*:

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

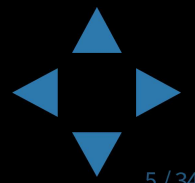
Follow us



Telerik Academy

Logical Operators

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



Follow us

Telerik Academy

De Morgan Laws

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

Follow us

Telerik Academy

if and if-else

Implementing Conditional Logic

Follow us

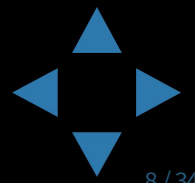
Telerik Academy

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;
}
```

Follow us



Telerik Academy

Condition and Statement

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

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

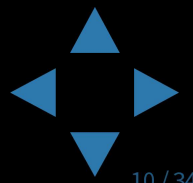
Follow us

Telerik Academy

How It Works?

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

Follow us



Telerik Academy

The if Statement – *Example*

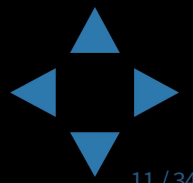
```
static void Main()
{
    Console.WriteLine("Enter two numbers.");

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

    // condition
    if (smallerNumber > biggerNumber)
    {
        // statement
        biggerNumber = smallerNumber;
    }

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

Follow us



Telerik Academy

The `if` Statement

Demo

Follow us

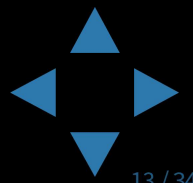
Telerik Academy

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;
}
```

Follow us

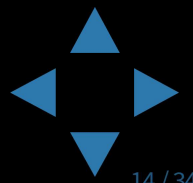


Telerik Academy

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

Follow us



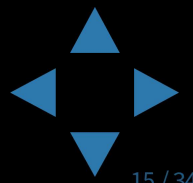
Telerik Academy `if-else` Statement – *Example*

- 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.");
}
```

Follow us



Telerik Academy

The `if-else` Statement

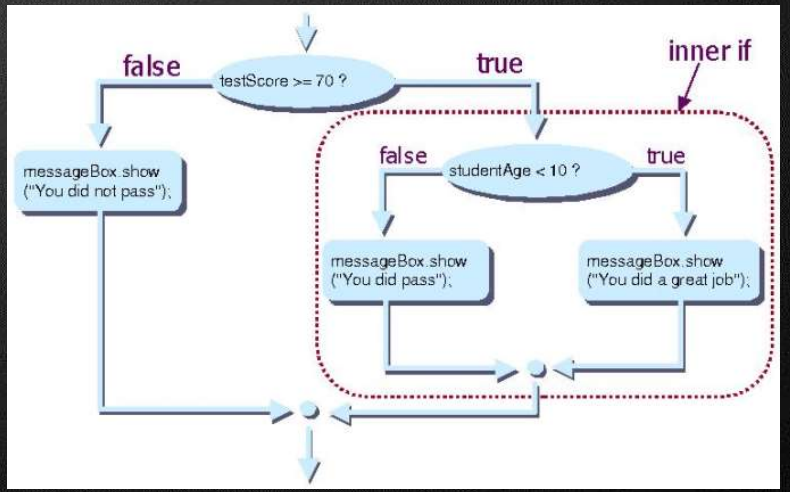
Demo

Follow us

Telerik Academy

Nested if Statements

Creating More Complex Logic



Follow us

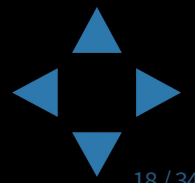
Telerik Academy

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;
```

Follow us

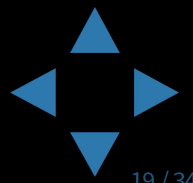


Telerik Academy

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

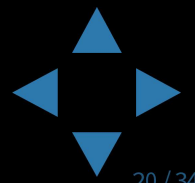
Follow us



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.");
    }
}
```

Follow us



Telerik Academy

Nested `if` Statements

Demo

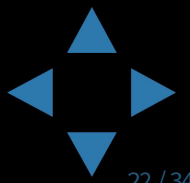
Follow us

Telerik Academy 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 …
```

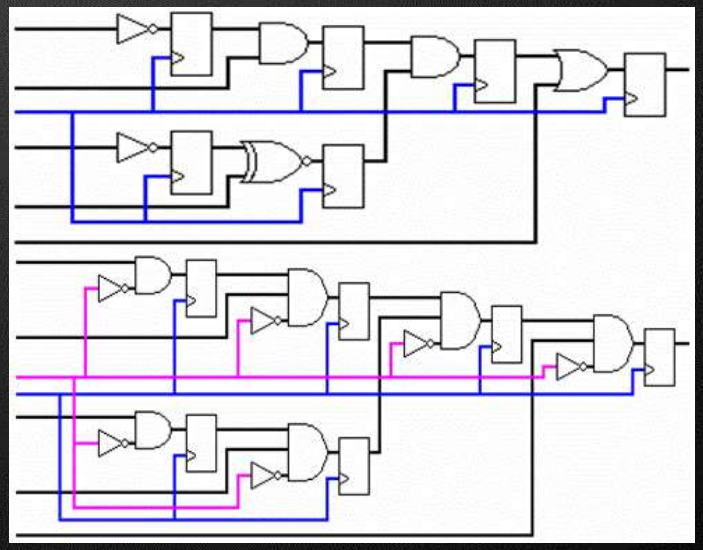
Follow us



Telerik Academy

Multiple `if-else` Statements

Demo



Follow us

Telerik Academy

Switch-case

Making Several Comparisons at Once

Follow us

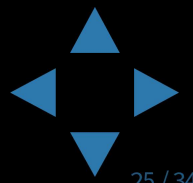
Telerik Academy

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;
}
```

Follow us

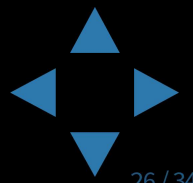


Telerik Academy

How `switch-case` Works?

- The expression is evaluated
- When one of the constants specified in a case label is equal to the expression
  - The statement that corresponds to that `case` is executed
- 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

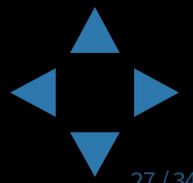
Follow us



Telerik Academy

The `switch-case` Statement

Demo



Follow us

Telerik Academy

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

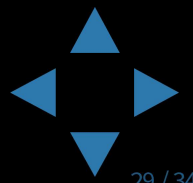
Follow us

Telerik Academy

Multiple Labels – *Example*

```
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;
}
```

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



Follow us

Telerik Academy

Multiple Labels in a `switch-case`

Demo

Follow us

Telerik Academy 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

Follow us

Telerik Academy

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

Follow us

Telerik Academy

Conditional Statements

Questions?

Follow us

Telerik Academy

Free Trainings @ Telerik Academy

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

Follow us