Telerik Academy

Methods

Subroutines in Computer Programming

C# Advanced

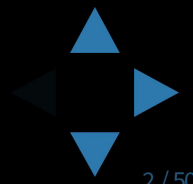
Telerik Software Academy

https://telerikacademy.com

Follow us

Telerik Academy

Table of Contents



Telerik Academy

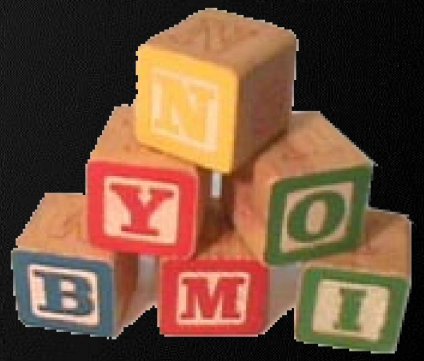
What is a Method?

- A method is a kind of building block that solves a small problem
  - A piece of code that has a name and can be called from the other code
  - Can take parameters and return a value
- Methods allow programmers to construct large programs from simple pieces
- Methods are also known as functions, procedures, and subroutines

Telerik Academy

Why to Use Methods?

- More manageable programming
  - Split large problems into small pieces
  - Better organization of the program
  - Improve code readability
  - Improve code understandability
- Avoiding repeating code
  - Improve code maintainability
- Code reusability
  - Using existing methods several times



Telerik Academy

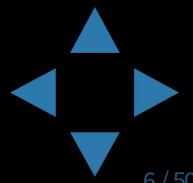
Declaring and Creating Methods

Telerik Academy

Declaring and Creating Methods

```
static void PrintLogo() // PrintLogo is the method's name
{
    Console.WriteLine("Telerik Corp.");
    Console.WriteLine("www.telerik.com");
}
```

- Each method has a name
  - It is used to call the method
  - Describes the method's purpose

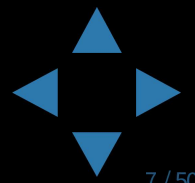


Telerik Academy

Declaring and Creating Methods

```
static void PrintLogo()
{
    Console.WriteLine("Telerik Corp.");
    Console.WriteLine("www.telerik.com");
}
```

- Methods declared `static` can be called by any other method (static or not)
  - This will be discussed later in details
- The keyword `void` means that the method does not return any result



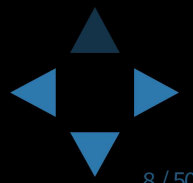
Telerik Academy

Declaring and Creating Methods

```
static void PrintLogo()
{
    Console.WriteLine("Telerik Corp.");
    Console.WriteLine("www.telerik.com");
}
```

Method body

- Each method has a body
  - It contains the programming code
  - Surrounded by { and }



Declaring and Creating Methods

```
using System;

class MethodExample
{
    static void PrintLogo()
    {
        Console.WriteLine("Telerik Corp.");
        Console.WriteLine("www.telerik.com");
    }

    static void Main()
    {
        // ...
    }
}
```

- Methods are always declared inside a `class`
- `Main()` is also a method like all others

Telerik Academy

Calling Methods

Telerik Academy

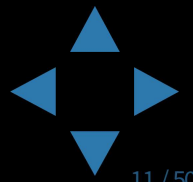
Calling Methods

- To call a method, simply use:
  1. The method's name
  2. Parentheses (don't forget them!)
  3. A semicolon (;)

```
PrintLogo();
```

- This will execute the code in the method's body and will result in printing the following:

```
Telerik Corp.
www.telerik.com
```

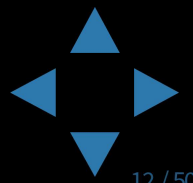



Telerik Academy

Calling Methods

- A method can be called from:
  - The `Main()` method
  - Any other method
  - Itself (process known as recursion)

```
static void Main()
{
    // ...
    PrintLogo();
    // ...
}
```

Telerik Academy

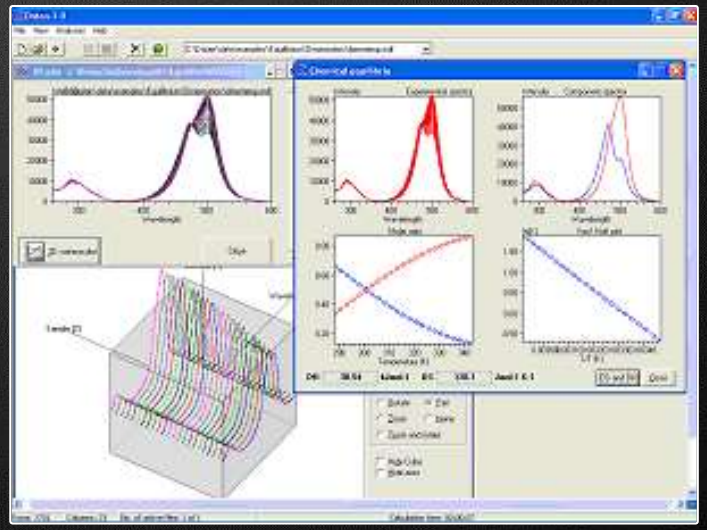
Declaring and Calling Methods

Demo

Telerik Academy

Methods with Parameters

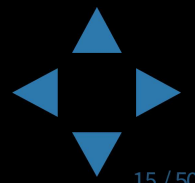
Passing Parameters and Returning values



Telerik Academy

Method Parameters

- To pass information to a method, you can use `parameters`(also known as `arguments`)
  - You can pass zero or several input values
  - You can pass values of different types
  - Each parameter has name and type
  - Parameters are assigned to particular values when the method is called
- Parameters can change the method behavior depending on the passed values

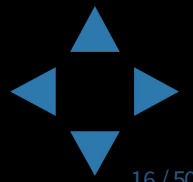


Telerik Academy

Defining and Using Method Parameters

```
static void PrintSign(int number)
{
    if (number > 0)
      { Console.WriteLine("Positive"); }
    else if (number < 0)
      { Console.WriteLine("Negative"); }
    else
      { Console.WriteLine("Zero"); }
}
```

- Method's behavior depends on its parameters
- Parameters can be of any type
  - `int`, `double`, `string`, etc.
  - Arrays (`int[]`, `double[]`, etc.)



Telerik Academy

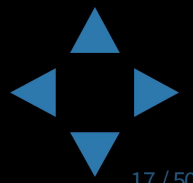
Defining and Using Method Parameters

- Methods can have as many parameters as needed:

```
static void PrintMax(float number1, float number2)
{
    float max = number1;
    if (number2 > number1)
        max = number2;
    Console.WriteLine("Maximal number: {0}", max);
}
```

- The following syntax is not valid:

```
static void PrintMax(float number1, number2)
```

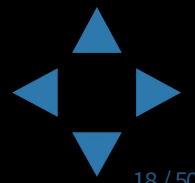


Telerik Academy

Calling Methods with Parameters

- To call a method and pass values to its parameters:
  - Use the method's name, followed by a list of expressions for each parameter
- *Examples*:

```
PrintSign(-5);
PrintSign(balance);
PrintSign(2+3);

PrintMax(100, 200);
PrintMax(oldQuantity * 1.5, quantity * 2);
```

Telerik Academy

Calling Methods with Parameters

- Expressions must be of the same type as method's parameters (or compatible)
  - If the method requires a `float` expression, you can pass `int` instead
- Use the same order like in method declaration
- For methods with no parameters do not forget the parentheses

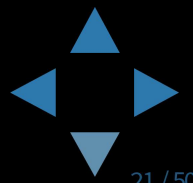
Telerik Academy

Using Methods With Parameters

Examples

Telerik Academy Methods Parameters – *Example*

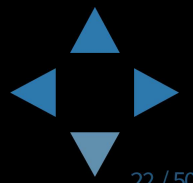
```
static void PrintSign(int number)
{
  if (number > 0)
    {  Console.WriteLine("The number {0} is positive.", number); }
  else if (number < 0)
    { Console.WriteLine("The number {0} is negative.", number); }
  else
    { Console.WriteLine("The number {0} is zero.", number); }
}

static void PrintMax(float number1, float number2)
{
  float max = number1;
  if (number2 > number1)
    { max = number2; }

  Console.WriteLine("Maximal number: {0}", max);
}
```



Telerik Academy

Method Parameters

Demo

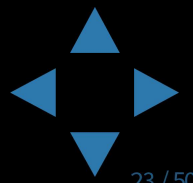


Telerik Academy

Months – *Example*

- Display the period between two months in a user-friendly way

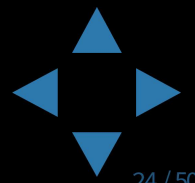
```
static void SayMonth(int month)
{
  string[] monthNames = new string[] {
    "January", "February", "March",
    "April", "May", "June", "July",
    "August", "September", "October",
    "November", "December" };

  Console.Write(monthNames[month-1]);
}
```

Telerik Academy

Months – *Example*

```
static void SayPeriod(int startMonth, int endMonth)
{
  int period = endMonth - startMonth;
  if (period < 0)
  {
    period = period + 12;
    // From December to January the
    // period is 1 month, not -11!
  }

  Console.Write("There are {0} + months from ", period);
  SayMonth(startMonth);
  Console.Write(" to ");
  SayMonth(endMonth);
}
```

Telerik Academy

Months

Demo

Printing Triangle – *Example*

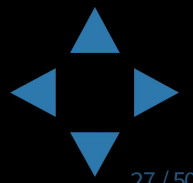
- Creating a program for printing triangles as shown below:

```
n=5 →
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
n=6 →
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Printing Triangle – *Example*

```
static void Main()
{
    int n = int.Parse(Console.ReadLine());
    for (int line = 1; line <= n; line++)
        PrintLine(1, line);
    for (int line = n-1; line >= 1; line--)
        PrintLine(1, line);
}

static void PrintLine(int start, int end)
{
    for (int i = start; i <= end; i++)
    {
        Console.Write(" {0}", i);
    }
    Console.WriteLine();
}
```



Telerik Academy

Printing Triangle

Demo

Telerik Academy

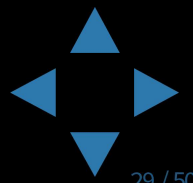
Optional Parameters

- C# supports optional parameters since v4.0 with default values assigned at their declaration:

```
static void PrintNumbers(int start = 0, int end = 100)
{
  for (int i = start; i <= end; i++)
  {
      Console.Write("{0} ", i);
  }
}
```

- The above method can be called in several ways:

```
PrintNumbers(5, 10);
PrintNumbers(15);
PrintNumbers();
PrintNumbers(end: 40, start: 35);
```



Telerik Academy

Optional Parameters

Demo

Telerik Academy

Returning Values From Methods

Telerik Academy

Returning Values From Methods

- A method can `return` a value to its caller
  - Can be assigned to a variable:

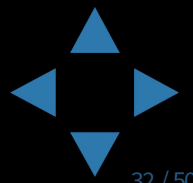
```
string message = Console.ReadLine();
// Console.ReadLine() returns a string
```

  - Can be used in expressions:

```
float price = GetPrice() * quantity * 1.20;
```

  - Can be passed to another method:

```
int age = int.Parse(Console.ReadLine());
```

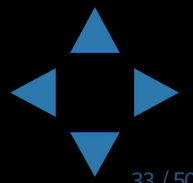
Telerik Academy

Defining Methods That Return a Value

- Instead of `void`, specify the type of data to return

```
static int Multiply(int firstNum, int secondNum)
{
    return firstNum * secondNum;
}
```

- Methods can return any type of data (`int`, `string`, etc.)
- `void` methods do not return anything
- The combination of method's name and parameters is called method signature
- Use `return` keyword to return a result



Telerik Academy

The return Statement

- The return statement:
  - Immediately terminates method's execution
  - Returns specified expression to the caller
  - *Example*:

```
int SomeMethod()
{
    return -1; // optional before and after the return statement
}
```

- To terminate void method, use just:

```
void SomeVoidMethod()
{
    return; // optional before and after the return statement
}
```

- return can be used several times in a method body

Telerik Academy

Returning Values From Methods

Examples

Telerik Academy

Temperature Conversion *Example*

- Convert temperature from Fahrenheit to Celsius:

```
static double FahrenheitToCelsius(double degrees)
{
    double celsius = (degrees - 32) * 5 / 9;
    return celsius;
}

static void Main()
{
    Console.Write("Temperature in Fahrenheit: ");
    double t = Double.Parse(Console.ReadLine());
    t = FahrenheitToCelsius(t);
    Console.Write("Temperature in Celsius: {0}", t);
}
```

Telerik Academy

Temperature Conversion

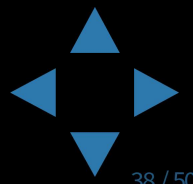
Demo

Telerik Academy

Positive Numbers *Example*

- Check if all numbers in a sequence are positive:

```
static bool ArePositive(int[] sequence)
{
    foreach (int number in sequence)
    {
        if (number <= 0)
        {
            return false;
        }
    }
    return true;
}
```

Telerik Academy

Positive Numbers

Demo

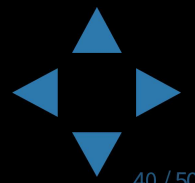
Telerik Academy

Data Validation – *Example*

- Validating input data:

```
using System;

class ValidatingDemo
{
  static void Main()
  {
    Console.WriteLine("What time is it?");

    Console.Write("Hours: ");
    int hours = int.Parse(Console.ReadLine());

    Console.Write("Minutes: ");
    int minutes = int.Parse(Console.ReadLine());
```

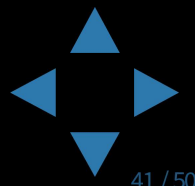
(example continues)



Telerik Academy

Data Validation – *Example*

```
    bool isValidTime = ValidateHours(hours) &&
                       ValidateMinutes(minutes);

    if (isValidTime)
      Console.WriteLine("It is {0}:{1}", hours, minutes);
    else
      Console.WriteLine("Incorrect time!");
  }

  static bool ValidateMinutes(int minutes)
  {
    bool result = (minutes>=0) && (minutes<=59);
    return result;
  }

  static bool ValidateHours(int hours) { ... }
}
```



Telerik Academy

Data Validation

Demo

Telerik Academy

Overloading Methods

Multiple Methods with the Same Name

Telerik Academy

Overloading Methods

- What means "to overload a method name"?
  - Use the same method name for multiple methods with different signature (parameters)

```
static void Print(string text)
{
  Console.WriteLine(text);
}

static void Print(int number)
{
  Console.WriteLine(number);
}

static void Print(string text, int number)
{
  Console.WriteLine(text + ' ' + number);
}
```

Telerik Academy

Variable Number of Parameters

Telerik Academy

Variable Number of Parameters

- A method in C# can take variable number of parameters by specifying the `params` keyword

```
static long CalcSum(params int[] elements)
{
    long sum = 0;
    foreach (int element in elements)
        sum += element;
    return sum;
}

static void Main()
{
    Console.WriteLine(CalcSum(2, 5));
    Console.WriteLine(CalcSum(4, 0, -2, 12));
    Console.WriteLine(CalcSum());
}
```

Telerik Academy

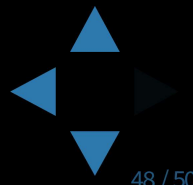
Methods – Best Practices

- Each method should perform a single, well-defined task
- Method's name should describe that task in a clear and non-ambiguous way
  - Good examples: `CalculatePrice`, `ReadName`
  - Bad examples: `f`, `g1`, `Process`
- In C# methods should start with capital letter
  - So called Pascal case
- Avoid methods longer than one screen
  - Split them to several shorter methods

Telerik Academy

Summary

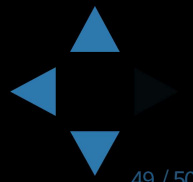
- Break large programs into simple methods that solve small sub-problems
- Methods consist of declaration and body
- Methods are invoked by their name
- Methods can accept parameters
  - Parameters take actual values when calling a method
- Methods can return a value or nothing



Telerik Academy

C# Methods

Questions?



Telerik Academy

Free Training @ Telerik Academy

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