

## **Algoritmia e Estruturas de Dados**

### **Módulo II – Introdução ao C#**



1. Plataforma .NET
2. Visual Studio
3. Sintaxe da linguagem de programação C#
  - ▶ Variáveis e Constantes
  - ▶ Tipos de dados
  - ▶ Operadores
  - ▶ Entrada e Saída de dados

## ▶ Definição

- ▶ Iniciativa da Microsoft, que visa uma plataforma única para desenvolvimento e execução de sistemas e aplicações
- ▶ O código gerado para .NET pode ser executado em qualquer dispositivo que possua essa plataforma
- ▶ Lançado em 13 de Fevereiro de 2002
- ▶ Versão estável: 4.5.1 (de 2013)
- ▶ Página oficial: <http://www.microsoft.com/net>

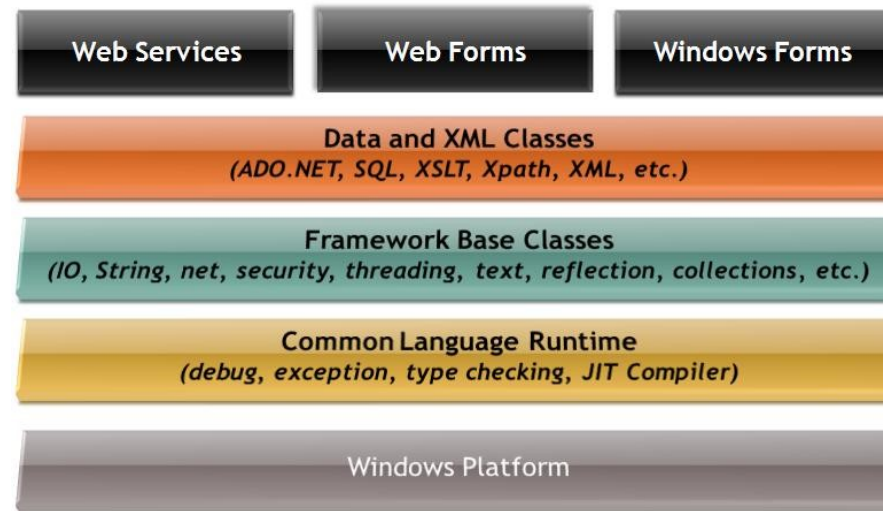


## ▶ .NET Framework

- ▶ Ambiente de desenvolvimento e de execução, que inclui um vasto conjunto de bibliotecas, e que permite que diferentes linguagens de programação corram na mesma plataforma para Windows
- ▶ Principais ferramentas:
  - ▶ Framework Class Libraries (FCL)
  - ▶ Common Language Runtime (CLR)

## ▶ Principais componentes

- ▶ FCL (Framework Class Libraries): classes pré-definidas que pode-se usar no código dos programas permitindo invocar código sem ter que escrever instruções de mais baixo nível



- ▶ CLR (Common Language Runtime): ambiente de execução e gestão do código dos programas (carregar e executar aplicações, tratar erros, gerir memória, etc...)

## ► Compilação

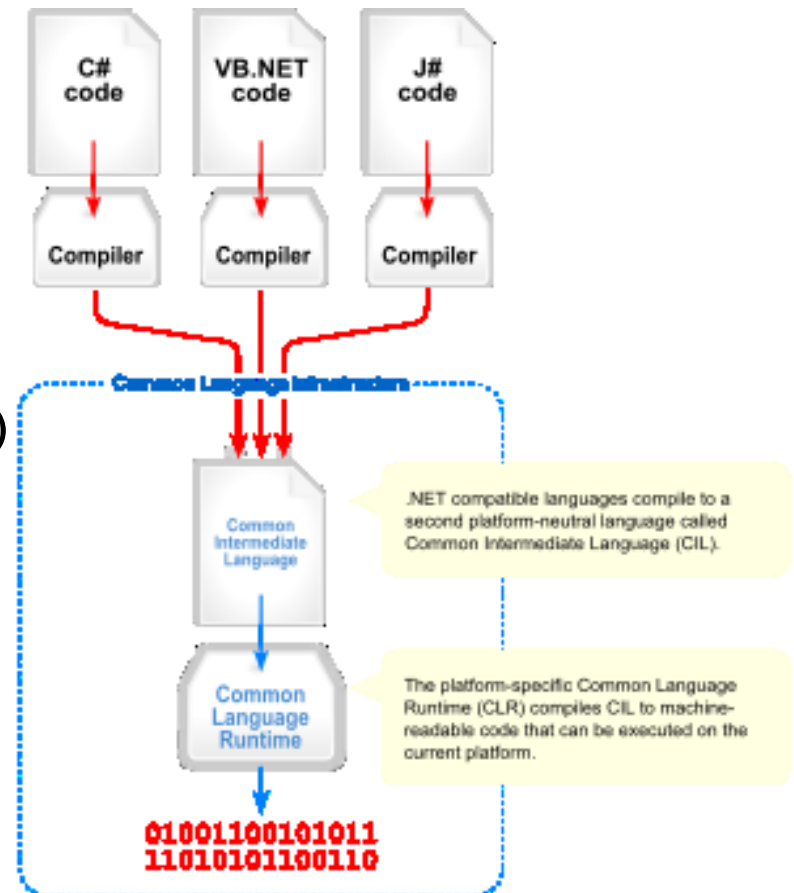
### ► Os programas são duplamente compilados:

#### ► Distribuição:

- ❑ Gerando o CIL (Common Intermediate Language)
- ❑ O CIL é executado no CLR interagindo com a framework

#### ► Execução:

- ❑ Usa o compilador JIT (Just-in-Time) de acordo com o uso no programa



## ► Compilação

- A plataforma .NET baseia-se num dos princípios utilizados na tecnologia Java (Just In Time Compiler - JIT)
- Os programas são compilados duas vezes, uma na distribuição (gerando um código que é conhecido como "bytecodes") e outra na execução.
- Um programa pode ser escrito em qualquer das mais de trinta e três linguagens de programação disponíveis para a plataforma.

- APL
- Boo
- Clarion
- COBOL
- Component Pascal
- C#
- C++
- F#
- Eiffel
- Forth
- Fortran

- Haskell
- Java
- JScript
- J#
- Lua
- Mercury
- Piet
- Oberon
- Delphi
- Oz
- Pascal

- Perl
- PowerBuilder
- PowerShell
- Python
- RPG
- Ruby
- Scheme
- SmallTalk
- Standard ML
- Visual Basic
- xBase

## ► Compilação

- O código fonte gerado pelo programador é compilado pela linguagem escolhida, gerando um código intermedio numa linguagem chamada MSIL (Microsoft Intermediate Language).
- Este novo código fonte gera um arquivo na linguagem de baixo nível Assembly, de acordo com o tipo de projeto:
  - EXE - Arquivos Executáveis, Programas
  - DLL - Biblioteca de Funções
  - ASPX - Página Web
  - ASMX - Web Service

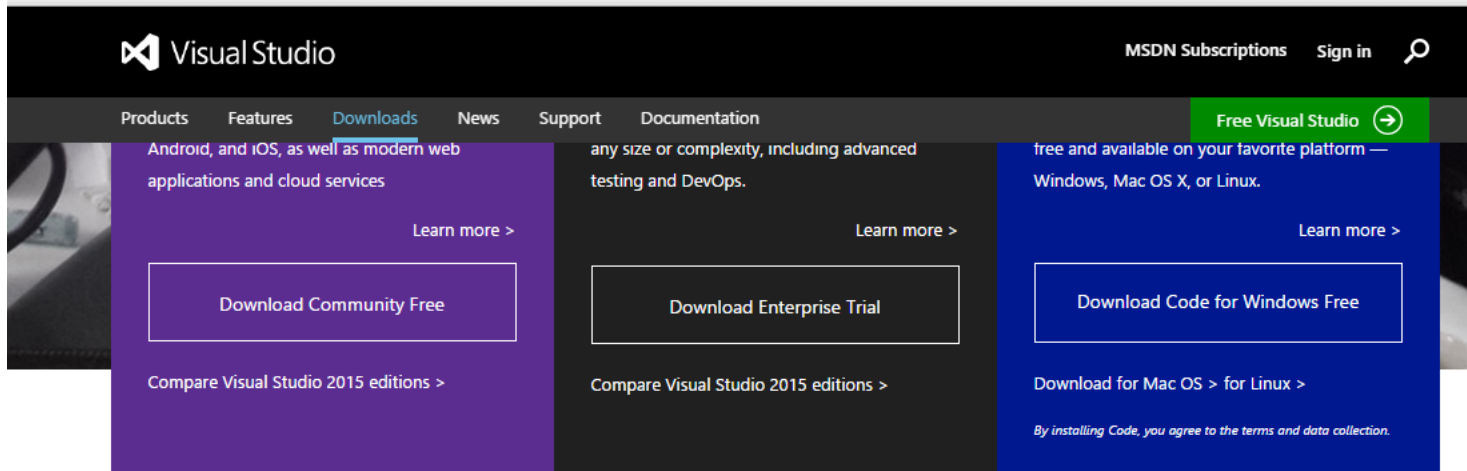


## ▶ Definição

- ▶ IDE (Integrated Development Environment) da Microsoft
- ▶ Usado para desenvolver aplicações executadas no:
  - ▶ Computador
    - ☐ Consola (sem interface gráfica)
    - ☐ Windows Forms (com interface gráfica)
  - ▶ Web (aplicações, sites e serviços)
- ▶ Versão estável: Visual Studio 2013
- ▶ Edição Express – gratuita!
- ▶ Página oficial: <http://www.microsoft.com/visualstudio>



- Download: <http://www.visualstudio.com/downloads/>



## Visual Studio downloads

[Visual Studio 2015](#)

[Team Foundation Server 2015](#)

[Visual Studio Code](#)

[Tools for Visual Studio 2015](#)

[Visual Studio 2013](#)

[Team Foundation Server 2013](#)

[Tools for Visual Studio 2013](#)

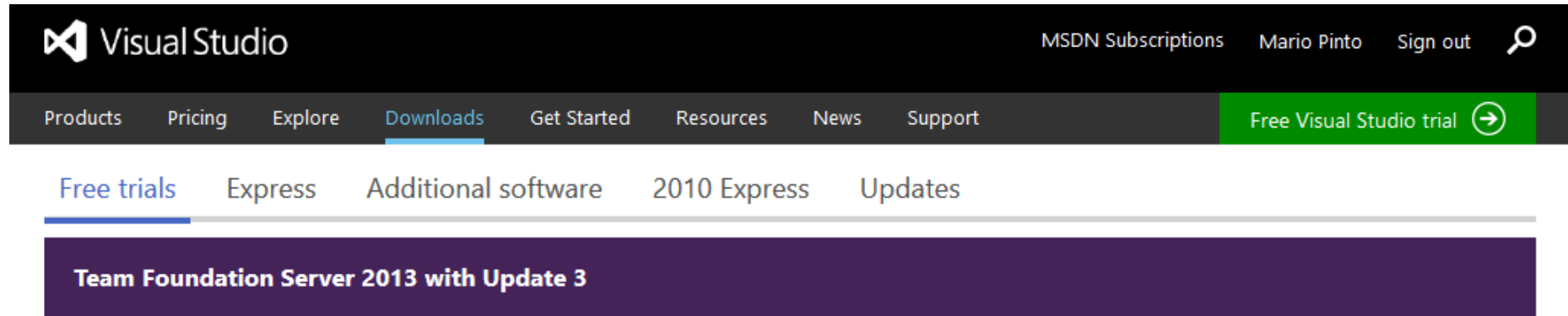
### Visual Studio Community 2013 with Update 5

Visual Studio Community 2013 is a free, full-featured IDE with powerful coding productivity features, cross-platform mobile development tools for Windows, iOS and Android, and access to thousands of extensions. This edition of Visual Studio is available at no cost for non-enterprise application development. After installation, check out the [Visual Studio Gallery](#). It provides quick access to tools, controls, and templates to help you get the most out of Visual Studio.

**Note:** If you already have Visual Studio Community 2013 (original release version) and run this download, only Update 5 is installed. If you don't have Visual Studio Community 2013 and run this download, both Visual Studio Community 2013 and Update 5 are installed. In either case, Visual Studio 2013 Language Packs (original release versions) can also be installed.

[Release notes](#)

► Download: <http://www.visualstudio.com/downloads/>



The screenshot shows the Visual Studio website's Downloads page. The top navigation bar includes the Visual Studio logo, MSDN Subscriptions, Mario Pinto, and Sign out. The main navigation bar has links for Products, Pricing, Explore, Downloads (highlighted), Get Started, Resources, News, and Support. A green button for 'Free Visual Studio trial' is on the right. Below the navigation bar, there are tabs for 'Free trials', 'Express', 'Additional software', '2010 Express', and 'Updates'. The 'Free trials' tab is selected, showing a list of download options. The first option is 'Team Foundation Server 2013 with Update 3'.

Visual Studio

MSDN Subscriptions Mario Pinto Sign out

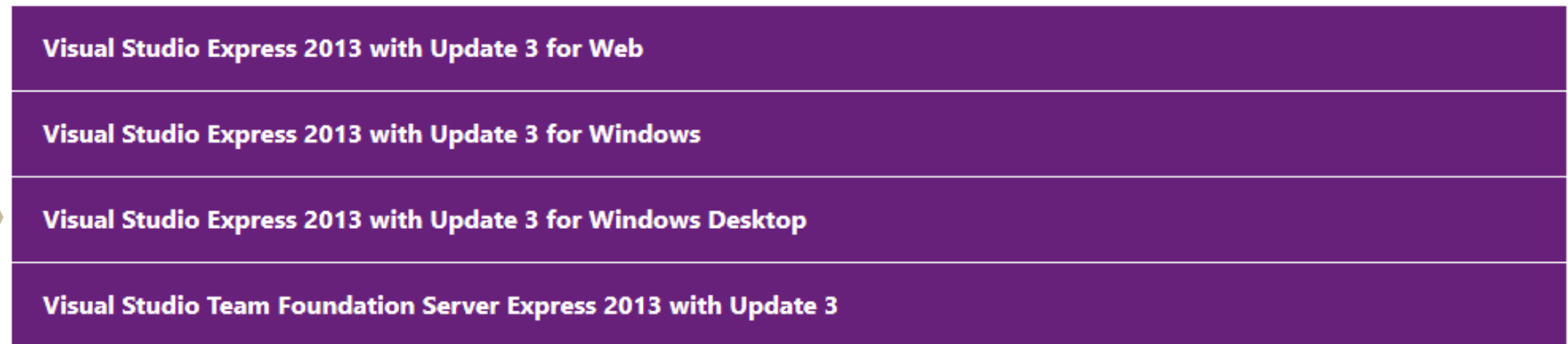
Products Pricing Explore Downloads Get Started Resources News Support

Free Visual Studio trial →

Free trials Express Additional software 2010 Express Updates

**Team Foundation Server 2013 with Update 3**

## Visual Studio Express



A list of four download options for Visual Studio Express, each in a purple box. A large yellow arrow points to the third option, 'Visual Studio Express 2013 with Update 3 for Windows Desktop'.

- Visual Studio Express 2013 with Update 3 for Web
- Visual Studio Express 2013 with Update 3 for Windows
- Visual Studio Express 2013 with Update 3 for Windows Desktop
- Visual Studio Team Foundation Server Express 2013 with Update 3

No cost to students: Visual Studio Professional 2013!

Download it today at DreamSpark.com →

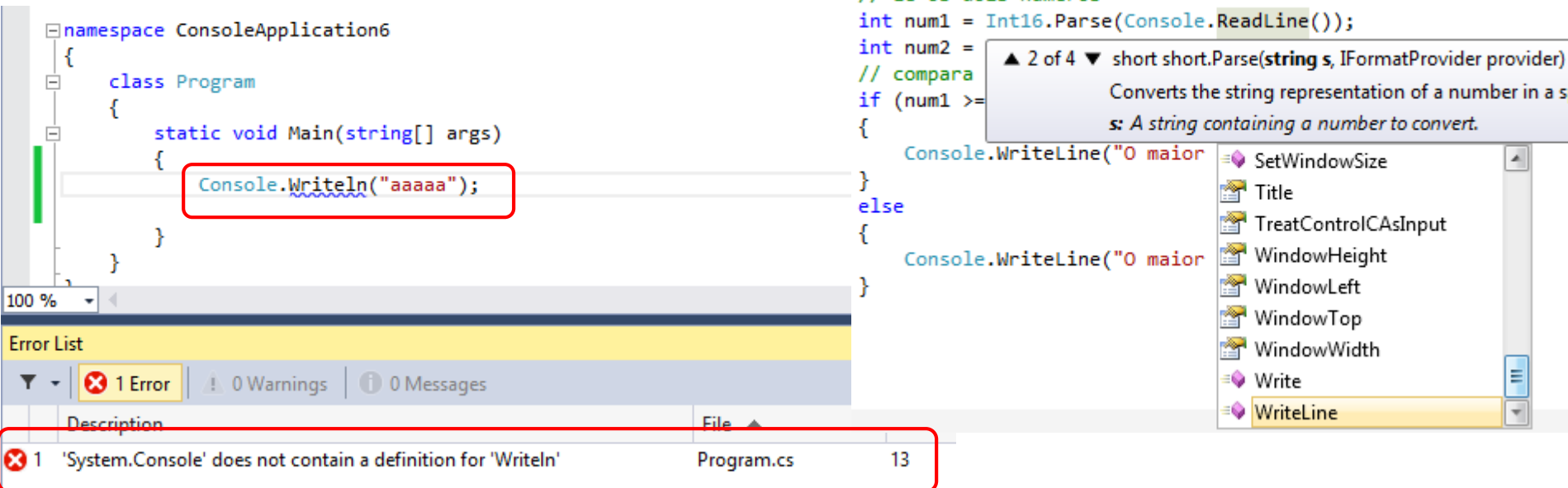
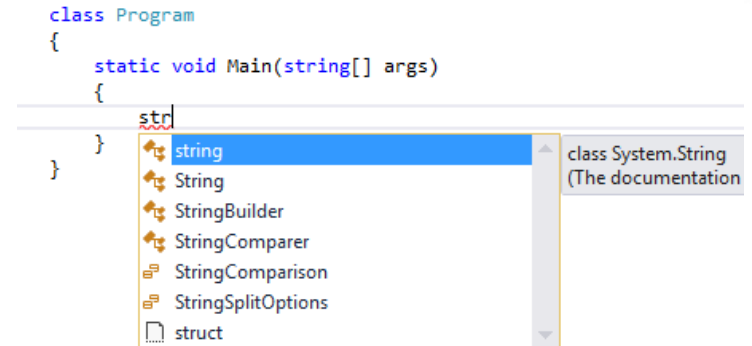
Microsoft DreamSpark

# Visual Studio

## ▶ Componentes

### ▶ Code Editor

- ▶ Realce da sintaxe
- ▶ IntelliSense : disponibiliza sugestões de sintaxe, permite completar palavras de forma automática
- ▶ Deteção de erros de sintaxe



# Visual Studio

## ▶ Componentes

### ▶ Debugger

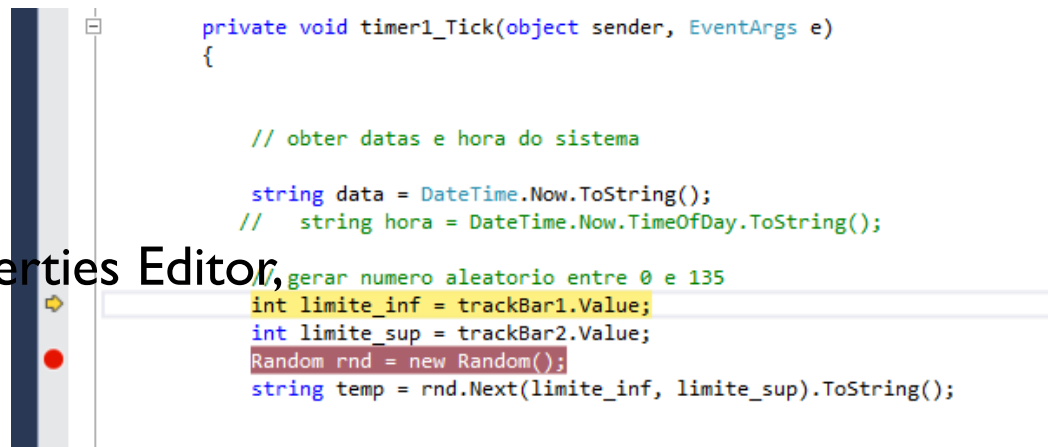
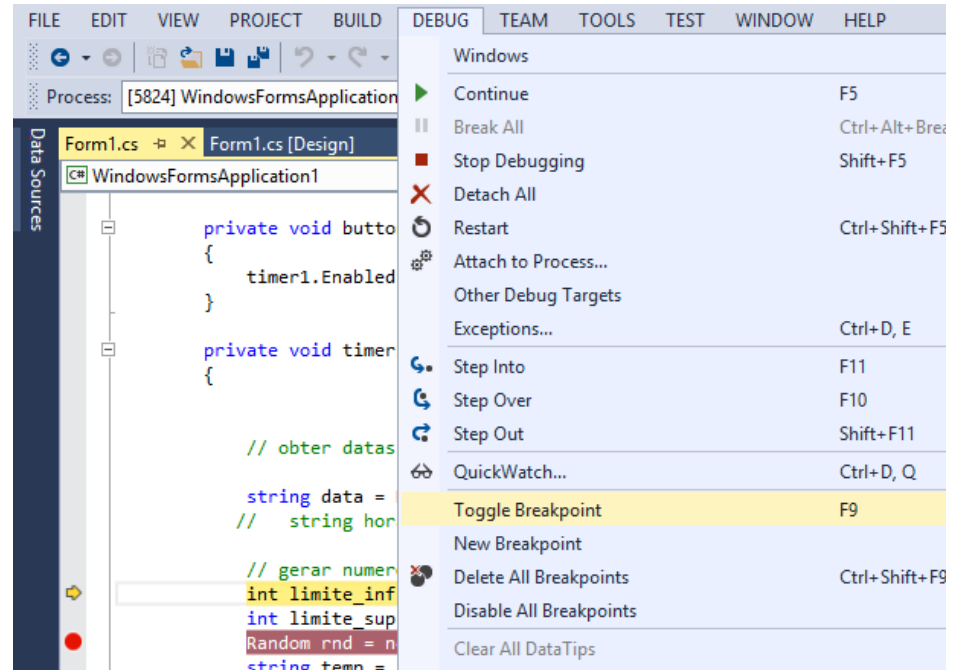
- ▶ Pontos de paragem
- ▶ Inspeção de variáveis

### ▶ Designer

- ▶ Desenho gráfico das janelas da aplicação
- ▶ Classes e Dados

### ▶ Outras ferramentas

- ▶ Solution Explorer, Properties Editor, Object Browser, etc.



## ► Passos para criar um projeto usando o VS Express

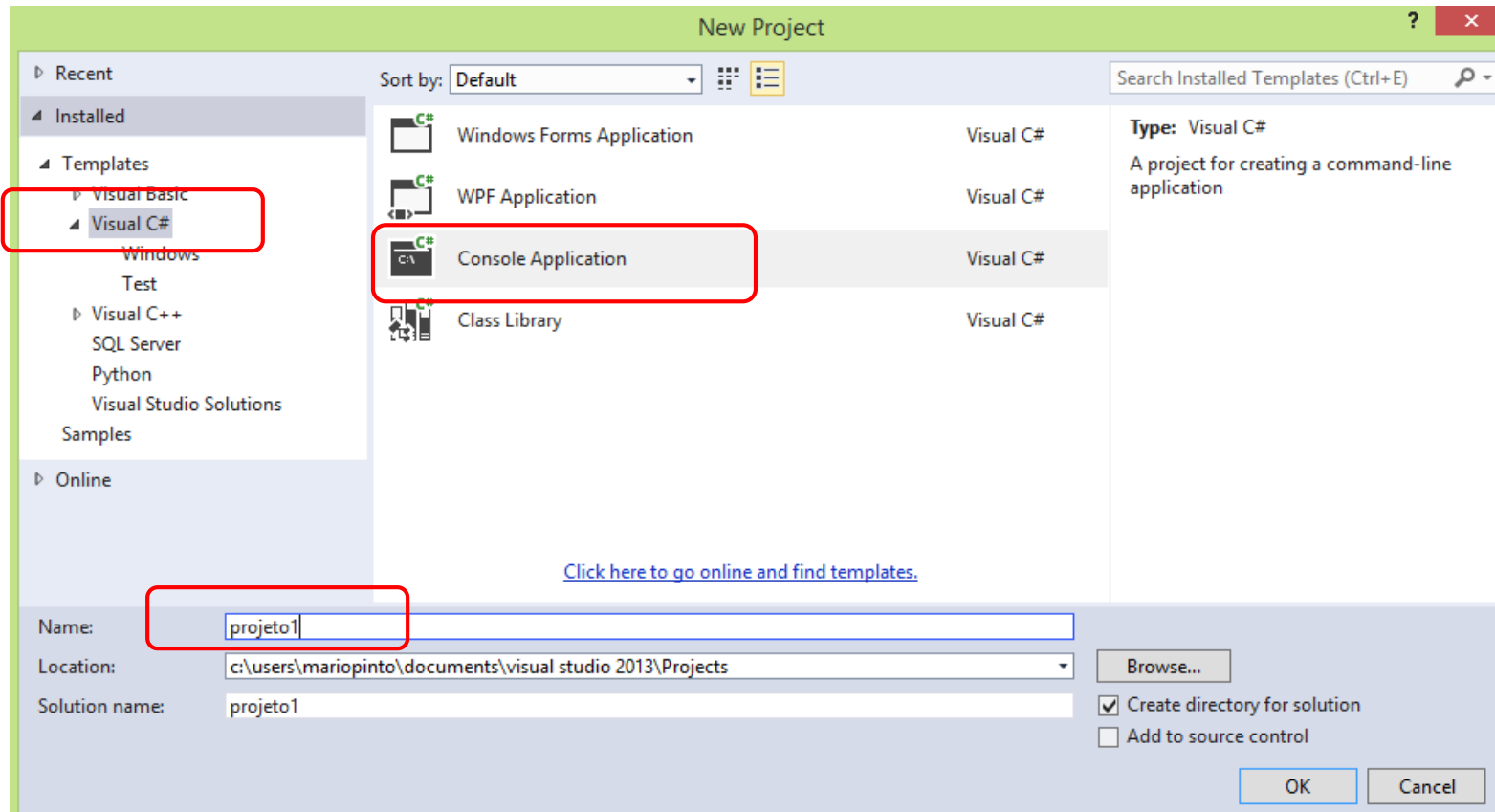
### I. Executar o Visual Studio



### 2. Clicar em **New Project** ou Menu **File > New Project**

## ► Passos para criar um projeto usando o VS Express

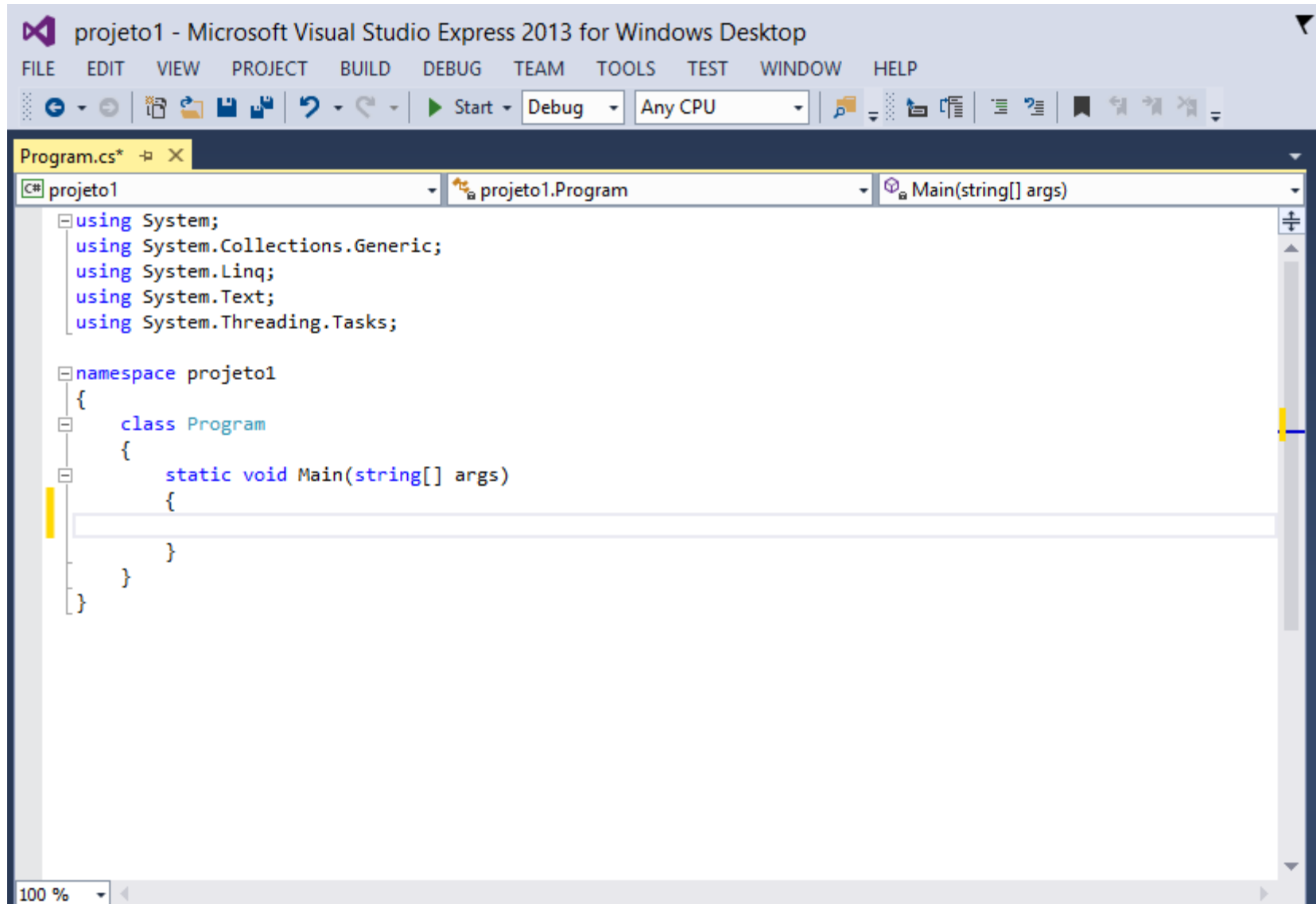
### 3. Selecionar **C#** nos templates instalados



### 4. Selecionar **Console Application** nos tipos de projeto

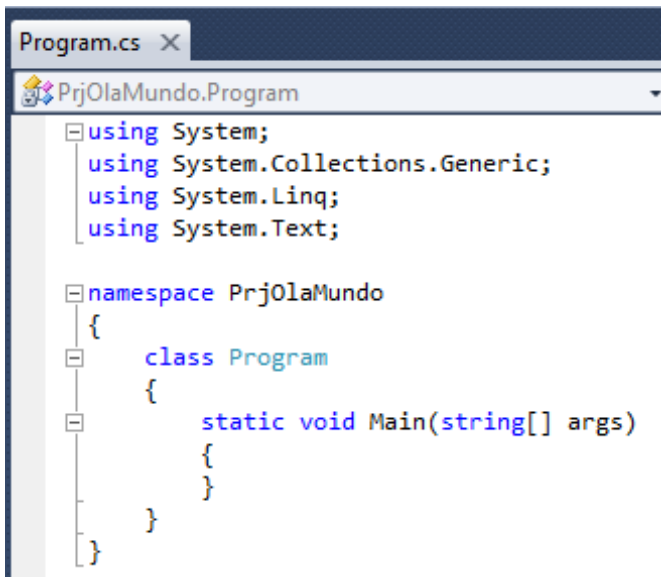
### 5. Nome do projeto

## ▶ Passos para criar um projeto usando o VS Express





- ▶ Passos para criar um projeto usando o VS Express
  - ▶ Conceitos gerais
    - ▶ Classes
    - ▶ Métodos e propriedades



```
Program.cs
PrjOlaMundo.Program

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace PrjOlaMundo
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

## O que é uma classe?

Uma classe representa um conjunto de objetos com características comuns. Consiste num modelo que define a natureza e a forma de um objeto (abstrato).

Todo o programa tem pelo menos

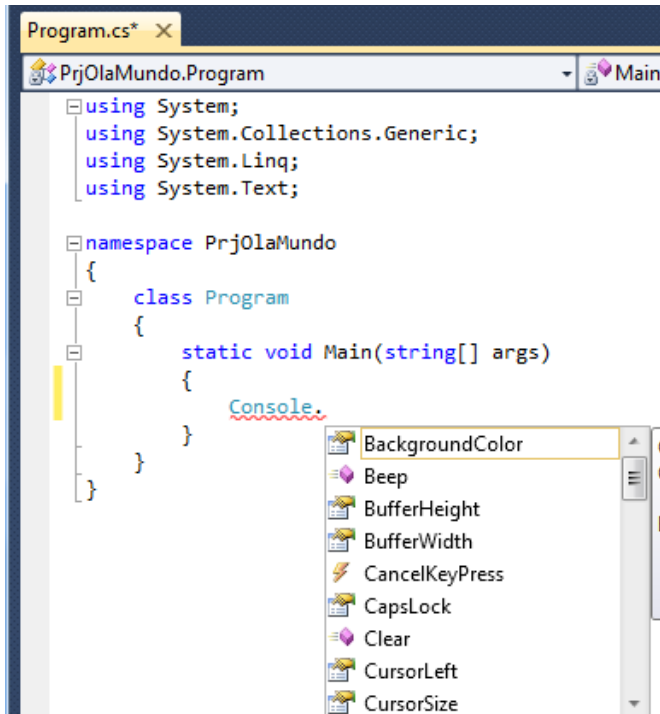
- uma classe (*Program*)
- um método (*Main*)

Apenas o conceito! Este tema será abordado com mais profundidade adiante!

## ▶ Passos para criar um projeto usando o VS Express

### ▶ Conceitos gerais

- ▶ Classes
- ▶ Métodos e propriedades



### O que é um método?

Os métodos representam as operações que os objetos pertencentes a essa classe podem executar.

Os métodos são declarados dentro das classes, são funções.

### O que é uma propriedade?

As propriedades definem o estado de um objeto.

- ▶ Variáveis e Constantes
- ▶ Tipos de Dados
- ▶ Operadores
- ▶ Entrada e Saída de dados (classe Console)

## ▶ Declaração

```
int numero;           // Declaração da variável chamada 'numero', do tipo 'int'
```

## ▶ Inicialização

```
int numero;           // Declaração da variável  
numero = 35;          // Inicialização da variável
```

## ▶ Declaração e Inicialização

```
int numero = 35;       // Declaração e inicialização da variável ao mesmo tempo
```

```
int a, b;               // Declaração de múltiplas variáveis do mesmo tipo  
int a = 2, b = 3;       // Declaração e inicialização de múltiplas variáveis do mesmo tipo
```

## ▶ Constantes

```
const double PI = 3.14; // Declaração e inicialização obrigatória para constantes
```

Tipos de dados primitivos

Tipo	Intervalo	Tamanho	Valor por omissão
Inteiros			
sbyte	-128 até 127	8 bits	0
short	-32,768 até 32,767	16 bits	0
int	-/+2,147,483,648	32 bits	0
long	-/+9,223,372,036,854,775,808	64 bits	0
byte	0 até 255	8 bits	0
ushort	0 até 65,535	16 bits	0
uint	0 até 4,294,967,295	32 bits	0
ulong	0 até 18,446,744,073,709,551,615	64 bits	0
Decimais			
decimal	$1.0 \times 10^{-28}$ to $7.9 \times 10^{28}$	128 bits	0.0
float	$1.5 \times 10^{-45}$ até $3.4 \times 10^{38}$	32 bits	0.0
double	$5.0 \times 10^{-324}$ to $1.7 \times 10^{308}$	64 bits	0.0
Não numéricos			
bool	true ou false	8 bits	false
char	'\u0000' até '\uFFFF'	16 bits	'\u0000' (null)

- ▶ Strings
  - ▶ Representa uma sequência imutável de caracteres
  - ▶ Não é um tipo de dados primitivo, mas sim uma classe!

```
string nome= “ESEIG”;    // Declaração e inicialização da variável nome com o valor “ESEIG”
```

Falaremos das Strings com  
mais profundidade adiante!

- ▶ Categorias
  - ▶ Matemáticos
  - ▶ Relacionais
  - ▶ Lógicos

## ▶ Operadores matemáticos

Categoria	Operadores
Básicos	+ (soma) - (subtração) * (multiplicação) / (divisão) % (resto da divisão inteira)
Atribuição simples	=
Atribuição composta	+=, -=, *=, /=, %=, &=
Incremento Decremento	++, --

*// Operações matemáticas básicas*

```
int a = 5, b = 2, c;
```

```
c = a + b; // c vai ser igual a 7
```

```
c = a - b; // c vai ser igual a 3
```

```
c = a * b; // c vai ser igual a 10
```

```
c = a / b; // c vai ser igual a 2
```

```
c = a % b; // c vai ser igual a 1
```

*// Atribuição simples e composta*

```
int x = 5;
```

```
int y = x; // atribui o valor de x a y. Neste caso y = x = 5.
```

```
y += 2;
```

*// Incrementa a variável y de 2 valores.*

*// Neste caso y = 7 e x mantém o valor 5. Similar a y = y + 2*



## ► Incrementação/Decrementação

Categoria	Operadores
Básicos	+ (soma) - (subtração) * (multiplicação) / (divisão) % (resto da divisão inteira)
Atribuição simples	=
Atribuição composta	+=, - =, *=, /=, %=, &=
Incremento Decremento	++, --

*// Operações de incrementação e decrementação*

```
int x = 5;
```

```
int y = ++x;
```

*// incrementa a variável x e atribui a y. Neste caso: x=6 e y=6*

```
int x = 5;
```

```
int y = x++;
```

*// atribui o valor de x a y. Neste caso y = 5. incrementa x (x = 6)*

```
int x = 5;
```

```
int y = --x;
```

*// decrementa a variável x e atribui a y. Neste caso: x=4 e y=4*

```
int x = 5;
```

```
int y = x--;
```

*// atribui o valor de x a y. Neste caso y = 5. Decrementa x (x = 4)*

## ► Operadores relacionais

Categoria	Operadores
Relacionais	<code>==</code> <code>!=</code> <code>&lt;</code> <code>&gt;</code> <code>&lt;=</code> <code>&gt;=</code>

*// Operações relacionais*

```
int a = 5, b = 2;
```

```
bool c;
```

```
c = a == b; // c vai ser igual a false
```

```
c = a != b; // c vai ser igual a true
```

```
c = a < b; // c vai ser igual a false
```

```
c = a > b; // c vai ser igual a true
```

```
c = a <= b; // c vai ser igual a false
```

```
c = a >= b; // c vai ser igual a true
```

## ▶ Operadores lógicos

Categoria	Operadores
Lógicos	&& (E)    (OU) ! (NÃO)

*// Operações lógicas*

**int** a = 5, b = 2;

**bool** c;

c = (a == b && a!=b);      // c vai ser igual a false

c = (a == b || a!=b);      // c vai ser igual a true

c = !(a == b);              // c vai ser igual a true

## ▶ Console

- ▶ A classe Console oferece suporte às funções de
  - ▶ Entrada de dados
  - ▶ Saída de dados
  - ▶ Outras interações com aplicações no modo de consola

Nome	Tipo	Descrição
WriteLine	Método	Escreve dados de saída
ReadLine	Método	Lê dados de entrada
Clear	Método	Limpa o ecrã
Beep	Método	Emite um som
BackgroundColor	Propriedade	Define uma cor de fundo da consola
ForegroundColor	Propriedade	Define um cor para o texto da consola
Title	Propriedade	Define um título na janela da consola

## ▶ Entrada de dados // LER em algoritmia

### ▶ Console.ReadLine

```
// lê dados do teclado até que o utilizador pressione ENTER. O texto digitado é colocado na variável nome  
string nome = Console.ReadLine();
```

```
// lê dados do teclado até que o utilizador pressione ENTER. O texto digitado é convertido para inteiro e guardado na variável numero  
int numero = Convert.ToInt16(Console.ReadLine());
```

### ▶ Console.Read

```
// lê um carácter do teclado e guarda-o na variável letra  
char letra = Convert.ToChar(Console.Read());
```

- ▶ Saída de dados // **ESCREVER** em algoritmia

- ▶ **Console.WriteLine**

```
// escrita de literais
```

```
Console.WriteLine("olá mundo!");
```

```
// escrita de texto concatenado com variáveis
```

```
int a = 1, b=2;
```

```
Console.WriteLine("A soma de {0} com {1} é igual a {2}", a, b, a+b);
```

```
// escrita de variáveis
```

```
Console.WriteLine(a+b);
```

- ▶ **Console.Write**

```
// Escrita de texto sem mudança de linha
```

```
Console.Write("A");
```

```
Console.Write("B");           //resultado é "AB"
```

➤ Saída de dados // **ESCREVER** em algoritmia

- Console.WriteLine();

```
Console.WriteLine("Função Factorial");  
Console.WriteLine("Factorial de {0} = {1}", numero, factorial);  
Console.WriteLine(factorial);
```

Console.WriteLine("Factorial de {0} = {1}", numero, factorial);

Escrita de Texto concatenado  
com 2 argumentos

- ▶ Exercício: ler dois números e apresentar o resultado da sua soma

```
// Programa que lê dois números e apresenta o resultado da sua soma
```

```
int numero1, numero2;
```

```
Console.Write("Escreva o primeiro número: ");
```

```
numero1 = Convert.ToInt16(Console.ReadLine());
```

```
Console.Write("Escreva o segundo número: ");
```

```
numero2 = Convert.ToInt16(Console.ReadLine());
```

```
Console.Write("O resultado da soma é {0}", numero1+numero2);
```

```
Console.ReadLine();
```



## Classe Console: Algumas propriedades

Propriedade	Descrição
BackgroundColor	Define a cor de fundo da consola
ForegroundColor	Define a cor dos caracteres da consola
Title	Define o título da janela da consola
CapsLock	Verifica o estado do Caps Lock

```
static void Main(string[] args)
{
    // cor de fundo
    Console.BackgroundColor = ConsoleColor.Blue;
    // cor do texto
    Console.ForegroundColor = ConsoleColor.White;
    // verifica de CpasLock está ligado
    if Console.CapsLock then
        Console.WriteLine("Caps Lock está ligado");
    // título da janela de execução
    Console.Title = "Cálculo de Factorial";
}
```

## Classe Console: Alguns métodos

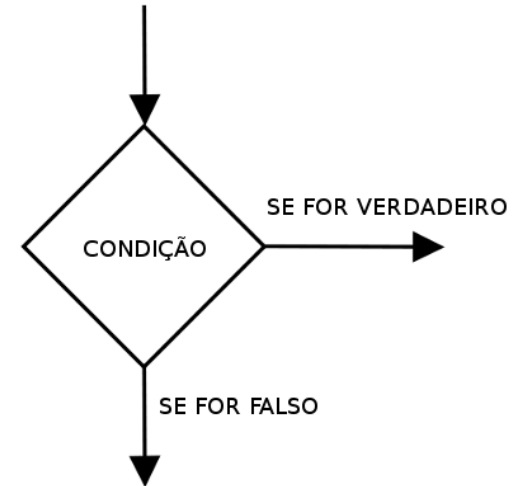
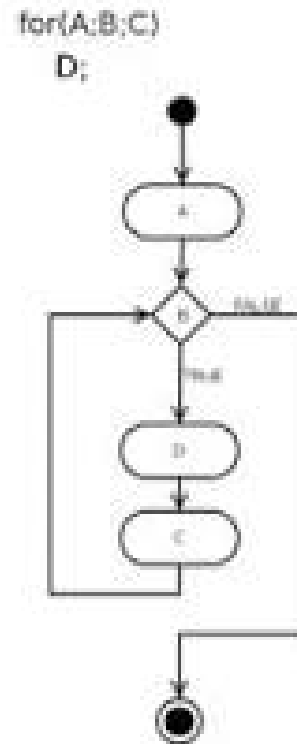
Método	Descrição
Clear	Limpa a janela da consola
Beep	Som, por defeito 800 hertz com duração de 200 milissegundos
ReadKey	Lê um carácter
ReadLine	Entrada de dados
WriteLine	Saída de dados
SetCursorPosition	Posição do cursor na consola (coluna, linha)

```
static void Main(string[] args)
{
    // limpa a janela da consola
    Console.Clear();
    // posiciona o cursor
    Console.SetCursorPosition(8, 12);
    // emite um beep
    Console.Beep();
}
```

## □ Estruturas de Controlo

□ Condicionais

□ Repetitivas



## ► Estruturas Condicionais

```
SE condição ENTAO
    bloco_instruções I
SENAO
    bloco_instruções I
FIM SE
```

```
If (condição)
{
    bloco_instruções I
}
else
{
    bloco_instruções2
}
```

```
Console.Clear();
Console.WriteLine("Número:");
int numero = int.Parse(Console.ReadLine());
int resto = numero & 2;
if (resto == 0)
{
    Console.WriteLine("O Número {0} é par", numero);
}
else
{
    Console.WriteLine("O Número {0} é ímpar", numero);
}
Console.ReadLine();
```

- ▶ Estruturas Condicionais
  - ▶ Estrutura de Condição Simples

## SINTAXE

```
if (<expressão-lógica>)  
    instrução
```

```
int r = 5;  
if (r > 0 && r%2==1)  
    Console.WriteLine("r é maior que 0");  
Console.WriteLine("r é ímpar"); //linha sempre executada
```

## SINTAXE

```
if (<expressão-lógica>) {  
    bloco de instruções  
}
```

```
int r = 5;  
if (r > 0 && r%2==1) {  
    Console.WriteLine("r é maior que 0");  
    Console.WriteLine("r é ímpar");  
}
```

- ▶ Estruturas Condicionais
  - ▶ Estrutura de Condição Compostas

## SINTAXE

```
if (<expressão-lógica>
{
    <bloco de comandos verdade>
}
else
{
    <bloco de comandos falsidade>
}
```

```
Console.WriteLine("Numero: ");
int numero = Convert.ToInt16(Console.ReadLine());
int resto = numero % 2;

if (resto==0)
{
    Console.WriteLine("O numero {0} é par", numero);
}
Else
{
    Console.WriteLine("O numero {0} é ímpar", numero);
}

Console.ReadLine();
```

## ► Estruturas Condicionais

### ► Estruturas encadeadas

#### **SINTAXE**

```
if (<expressão-lógica>) {  
    <bloco de instruções>  
} else if (<expressão-lógica>) {  
    <bloco de instruções>  
} else {  
    <bloco de instruções>  
}
```

```
if (nota >= 10 && nota <= 20)  
    Console.WriteLine("Aprovado");  
else if (nota >= 7 && nota < 10)  
    Console.WriteLine("Oral");  
else if (nota >= 0 && nota < 7)  
    Console.WriteLine("Reprovado");  
else  
    Console.WriteLine("Nota inválida!");
```

## ► Estruturas Condicionais

### ► Estrutura de ESCOLHE

LER mes

ESCOLHE mes seja

caso 1: ESCREVER “Janeiro”

caso 2: ESCREVER “Fevereiro”

caso 3: ESCREVER “Março”

.....

caso 12: ESCREVER “Dezembro”

DEFEITO

ESCREVER “Mês incorrecto”

FIM ESCOLHE

```
Console.Clear();
Console.WriteLine("Mês:");
int mes = int.Parse(Console.ReadLine());
switch (mes)
{
    case 1: Console.WriteLine("Janeiro");
            break;
    case 2: Console.WriteLine("Fevereiro");
            break;
    case 3: Console.WriteLine("Março");
            break;
    case 4: Console.WriteLine("Abril");
            break;
    case 5: Console.WriteLine("Maio");
            break;
    case 6: Console.WriteLine("Junho");
            break;
    case 7: Console.WriteLine("Julho");
            break;
    case 8: Console.WriteLine("Agosto");
            break;
    case 9: Console.WriteLine("Setembro");
            break;
    case 10: Console.WriteLine("Outubro");
            break;
    case 11: Console.WriteLine("Novembro");
            break;
    case 12: Console.WriteLine("Dezembro");
            break;
    default: Console.WriteLine("Mês incorrecto");
            break;
}
```



- ▶ Estruturas Condicionais
  - ▶ Estruturas ternárias
  - ▶ Mais compactas
  - ▶ Uso dos operadores ? e :

## SINTAXE

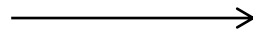
**<expressão> ? <verdadeira> : <falsa>**

```
int x = 1;  
int y = 2;  
Console.WriteLine(x > y ? "X é maior que Y" : "X é menor que Y");  
// resultado é: "X é menor que Y"
```

## ► Estruturas Repetitivas

### ► Ciclo FOR

PARA i de 1 até 10  
LER (nota)  
FIM PARA



```
int nota;  
for (int i = 1; i <=10; i++)  
{  
    nota = int.Parse(Console.ReadLine());  
}
```

**Exemplo do programa que dado um número, calcula o seu factorial**

```
static void Main(string[] args)  
{  
    // Calculo de factorial de um numero  
    Console.Clear();  
    Console.WriteLine("Número:");  
    int numero = int.Parse(Console.ReadLine());  
    int factorial = numero;  
    for (int i = numero - 1; i > 1; i--)  
    {  
        factorial = factorial * i;  
    }  
    Console.WriteLine("Factorial de {0} = {1}", numero, factorial);  
  
    Console.ReadLine();  
}
```

## ► Estruturas Repetitivas

### ► Ciclo FOR

#### SINTAXE

```
for(inicialização; condiçãoDeExecução; actualização)
{
    bloco de instruções
}
```

```
int i;
for (i = 0; i < 10; i++)
{
    Console.WriteLine("{0}", i);
}
```

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

- A **inicialização** é feita na primeira entrada no ciclo
- A **condição** a manter é especificada e avaliada em cada iteração
- A **atualização** é sempre executada no fim do ciclo

## ► Estruturas Repetitivas

### ► Ciclo WHILE

#### SINTAXE

```
while(condição)
{
    bloco de instruções
}
```

```
int x = 1;
while (x!=0)
{
    Console.Write("Escreva um número: ");
    x = Convert.ToInt16(Console.ReadLine());
}
Console.WriteLine("Escreveu um {0}", x);
```

```
Inteiro X ← 1
ENQUANTO x != 0 FAZ
    ESCREVER "Escreva um número"
    LER x
FIM ENQUANTO
```

```
Escreva um número: 2
Escreva um número: 3
Escreva um número: 0
Escreveu um 0
Prima qualquer tecla para continuar . . .
```

- A **condição** é especificada e avaliada em cada iteração no topo (à cabeça)

## ► Estruturas Repetitivas

### ► Ciclo DO...WHILE

#### SINTAXE

```
do
{
    bloco de instruções
} while(condição)
```

```
int num, i = 1;
Console.Write("Escreva um número: ");
num = Convert.ToInt16(Console.ReadLine());
do
{
    Console.WriteLine("{0}x{1}={2}", num, i, num *
i);
    i += 1;
} while (i < 11);
```

```
Escreva um número: 7
7x1=7
7x2=14
7x3=21
7x4=28
7x5=35
7x6=42
7x7=49
7x8=56
7x9=63
7x10=70
```


- A **condição** é especificada e avaliada no final de cada iteração (na cauda)

## ▶ Estruturas Repetitivas


### ▶ Quebra de Ciclo

- ▶ **break** – termina imediatamente qualquer ciclo
- ▶ **continue** – permite continuar diretamente para a próxima iteração de um ciclo sem executar as instruções seguintes da iteração corrente

```
int i = 0;
while(i <= 10)
{
    i++;
    if (i%4 == 0)
        break;
    Console.WriteLine("{0}",
        i);
}
```



```
int i = 0;
while(i <= 10)
{
    i++;
    if (i%4 == 0)
        continue;
    Console.WriteLine("{0}",
        i);
}
```



## ▶ Estruturas Repetitivas

### ▶ Escopo das variáveis

- ▶ As variáveis ao serem declaradas são apenas conhecidas dentro do bloco de código onde se encontram

```
For(int i=0; i<10; i++) {  
    int quadrado;  
    quadrado = i * i;  
    Console.WriteLine("O quadrado de {0} é {1}", i, quadrado);  
}
```

// Erro: i e quadrado não são conhecidos fora do ciclo for!!

```
Console.WriteLine("i={0}, quadrado={1}", i, quadrado);
```