# C#

Professor Leo Larback

Esta apresentação pode ser baixada livremente no site www.larback.com.br

# C# - Definições

C# é uma linguagem orientada a objeto que permite aos desenvolvedores construírem uma variedade de aplicações seguras e robustas, compatíveis com o .NET Framework. Você pode usar o C# para criar aplicações tradicionais do Windows, Web services baseados em XML, componentes distribuídos, aplicativos cliente-servidor, aplicativos com banco de dados e muito, muito mais.

Você pode baixar a versão express do C# diretamente do site www.microsoft.com/express

# Arquitetura da plataforma .NET

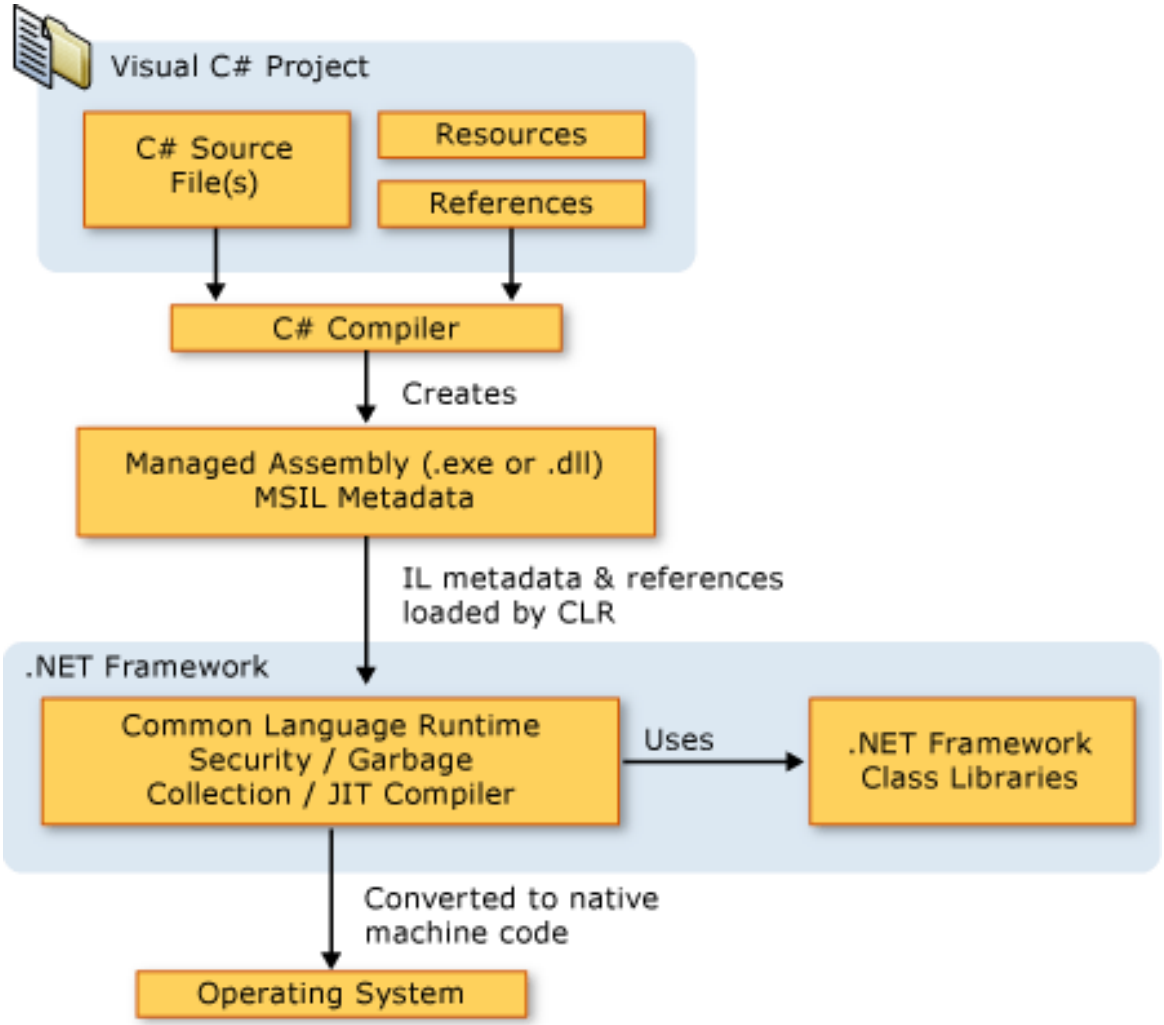
Programas escritos em C# são executados no .NET Framework, um componente do Windows que inclui um sistema de execução virtual chamado Common Language Runtime (CLR) e um conjunto unificado de bibliotecas de classes. O CLR é a implementação comercial da Microsoft da infraestrutura de linguagem comum (CLI), um padrão internacional que é a base para a criação e execução de ambientes de desenvolvimento em que as linguagens e as bibliotecas trabalham juntos sem problemas.

# Arquitetura da plataforma .NET

O código-fonte escrito em C# é compilado em uma linguagem intermediária (IL) que está em conformidade com a especificação CLI. O código IL e recursos, como bitmaps e strings, são armazenados no disco em um arquivo executável chamado de um assembly, normalmente com uma extensão .exe ou .dll. Um assembly contém um manifesto que fornece informações sobre os tipos do assembly, versão, cultura e requisitos de segurança.

Quando o programa C# é executado, o assembly é carregado para o CLR, que pode tomar várias ações com base nas informações do manifesto. Então, se os requisitos de segurança são aprovados, o CLR executa a compilação just in time (JIT) para converter o código IL em instruções de máquina. O CLR também oferece outros serviços relacionados à coleta de lixo automática, tratamento de exceções e gerenciamento de recursos. Código que é executado pelo CLR é muitas vezes referido como "código gerenciado", em contraste com o "código não-gerenciado", que é compilado em linguagem de máquina nativa e que foca um sistema específico. O diagrama a seguir ilustra as relações entre o tempo de compilação e de execução de arquivos de código-fonte C#, bibliotecas do NET Framework, assemblies e o CLR.

# Arquitetura da plataforma .NET




Retirado de www.larback.com.br

# Arquitetura da plataforma .NET

A interoperabilidade entre linguagens é um elemento chave do .NET Framework. Como o código IL produzido pelo compilador C# está de acordo com o Common Type Specification (CTS), o IL código gerado a partir do C# pode interagir com código que foi gerado das versões .NET do Visual Basic, Visual C++, Visual J #, ou qualquer um das mais de 20 linguagens compatíveis com o CTS. Um único assembly pode conter vários módulos escritos em diferentes linguagens .NET, e os tipos podem referenciar uns aos outros como se eles tivessem sido escritos na mesma linguagem.

Como adicional aos serviços de execução, o .NET Framework também inclui uma extensa biblioteca de mais de 4000 classes organizadas em namespaces que oferecem uma grande variedade de funcionalidades úteis para tudo, desde manipulação de arquivos, manipulação de strings, manipulação de arquivos XML até utilização de controles Windows Forms. Uma típica aplicação C# utiliza a biblioteca de classes do .NET Framework extensivamente para lidar com o comum "trabalho braçal".

Para mais, acesse: http://msdn.microsoft.com/pt-br/library/a4t23ktk(v=vs.90)

# C# - Variáveis e tipos de dados

| C# Type | .NET Framework type |
|---|---|
| bool | System.Boolean |
| byte | System.Byte |
| sbyte | System.SByte |
| char | System.Char |
| decimal | System.Decimal |
| double | System.Double |
| float | System.Single |
| int | System.Int32 |
| uint | System.UInt32 |
| long | System.Int64 |
| ulong | System.UInt64 |
| object | System.Object |
| short | System.Int16 |
| ushort | System.UInt16 |
| string | System.String |

**Retirado de www.larback.com.br**

# C# - Variáveis

| C# Type | Valores possíveis de se armazenar |
|---|---|
| bool | Verdadeiro ou Falso (Valores booleandos) |
| byte | 0 a 255 (8 bits) |
| sbyte | -128 a 127 (8 bits) |
| char | Um caractere (16 bits) |
| decimal | ±1.0 × 10−28 a ±7.9 × 1028 (128 bits) |
| double | ±5.0 × 10−324 a ±1.7 × 10308 (64 bits) |
| float | ±1.5 × 10−45 a ±3.4 × 1038 (32 bits) |
| int | -2,147,483,648 a 2,147,483,647 (32 bits) |
| uint | 0 a 4,294,967,295 (32 bits) |
| long | –9,223,372,036,854,775,808 a 9,223,372,036,854,775,807 (64 bits) |
| ulong | 0 a 18,446,744,073,709,551,615 (64 bits) |
| object | Qualquer tipo. |
| short | -32,768 a 32,767 (16 bits) |
| ushort | 0 a 65,535 (16 bits) |
| string | Sequência de caracteres (16 bits por caractere) |

# C# - Variáveis

A declaração de variáveis respeita a seguinte sintaxe:

<tipo> identificador;

Pode-se inicializar a variável no momento da declaração:

int x = 0;

Cada tipo no C# é um atalho para o tipo do Framework, assim declararmos: string nome Ou System.String.nome trará o mesmo resultado.

# C# - Nomeação de variáveis

A documentação do Microsoft .Net Framework da as seguintes recomendações para a nomeação das variáveis:

- Evite usar underline;
- Não crie variáveis que se diferenciem apenas pela sua forma. Exemplo: *minhaVariavel* e outra chamada *MinhaVariavel*;
- Procure iniciar o nome com uma letra minúscula;
- Evite usar todas as letras maiúsculas;
- Quando o nome tiver mais que uma palavra, a primeira letra de cada palavra após a primeira deve ser maiúscula (conhecido como notação camelCase);

**Retirado de www.larback.com.br**

# Estruturas de repetição

**ENQUANTO**

```
int i = 0;
while ( i < 5 ) {
        Console.WriteLine ( i );
         ++i;
}
```

repete 5 vezes e imprime o valor de i.

# Estruturas de repetição

REPITA (do…while)

```
int i = 0;
do {
        Console.WriteLine ( i );
        i++;
} while ( i < 5 );
```

# Estruturas de repetição

**PARA**

```
int i = 0;
for ( int i = 0; i < 5; i++ ) {
        Console.WriteLine ( i );
}
```

# Estruturas de repetição

**PARA APRIMORADO**

```
string [] nomes = new string[] { "Leo", "Larback"};
 foreach ( string nome in nomes ) {
        MessageBox.Show ( nome );
}
```

# Estruturas de repetição – Controle de fluxo

**BREAK**

```
string [] nomes = new string[] { “Leo", “Manoel", "Pedro"};
 foreach ( string nome in nomes ) {
        Console.WriteLine ( nome );
        if ( nome == "Manoel" )
            break;
 }
```

# Estruturas de repetição – Controle de fluxo

**CONTINUE**

```
string [] nomes = new string[] { “Leo", "Miriam", "Pedro"};
foreach ( string nome in nomes ) {
        if ( nome == "Miriam" )
                continue;
        Console.WriteLine ( nome );
}
```

# Estruturas condicionais

**SE**

```
string nome = “João";
if ( nome == "Pedro" ) {
        Console.WriteLine( “Bem vindo João'" );
} else {
        Console.WriteLine( “Você não é o João" );
}
```

# Estruturas condicionais

**SWICH**

```
int i = 6;
 switch ( i ) {
            case 5:
                        Console.WriteLine( "Valor de i é : " + 5 );
                        break;
            case 6:
                         Console.WriteLine( "Valor de i é : " + 6 );
                        break;
             case 4:
                        Console.WriteLine( "Valor de i é : " + 4 );
                        break;
             default:
                        Console.WriteLine( "Valor de i é : " + i );
                        break;
}
```

**Retirado de www.larback.com.br**