FHO | UNIARARAS

João Paulo - 102654

Lorran Kühl - 102145

Lucas Pedrini - 102136

Valmir dos Reis - 102890

Victor Menezes - 103704

História, Características e desenvolvimento C#.

Araras

2019

História da linguagem

A linguagem C# foi criada junto com a arquitetura .NET. Embora existam várias outras linguagens que suportam essa tecnologia (como VB.NET, C++, J#), C# é considerada a linguagem símbolo do .NET pelas seguintes razões:

* Foi criada praticamente do zero para funcionar na nova plataforma, sem preocupações de compatibilidade com código de legado.
* O compilador C# foi o primeiro a ser desenvolvido.
* A maior parte das classes do .NET Framework foram desenvolvidas em C#.

A criação da linguagem, embora tenha sido feita por vários desenvolvedores, é atribuída principalmente a Anders Hejlsberg, hoje um Distinguished Engineer na Microsoft. Anders Hejlsberg era desenvolvedor de compiladores na Borland, e entre suas criações mais conhecidas estão o Turbo Pascal e o Delphi.

Criação do nome

Muitos pensam que o nome C# viria de uma sobreposição de 4 símbolos "+" dando a impressão de "++++". Na verdade o "#" de C# refere-se ao sinal musical (sustenido), que aumenta em 1/2 tom uma nota musical.
O símbolo real seria o ♯ e não o #, porém, devido a limitação de telas, fontes e alguns browsers, no momento da normalização junto a ECMA, fora especificado apenas que o nome da linguagem seria uma letra C maiúscula (U+0043) e o sinal "#" (U+0023), facilitando assim, publicações e artigos com um caracter encontrado facilmente dos layouts de teclado padrões.
Desta forma, caso o nome fosse usado em português, seria "C-Sustenido" (ou "Dó-Sustenido"), e não "C-cerquilha".

Características

C# (pronuncia-se "cê chárp" em português ou "sí sharp" para o inglês) é, de certa forma, a linguagem de programação que mais diretamente reflete a plataforma .NET sobre a qual todos os programas .NET executam. C# está de tal forma ligado a esta plataforma que não existe o conceito de código não-gerenciado (unmanaged code) em C#. Suas estruturas de dados primitivas são objetos que correspondem a tipos em .NET. A desalocação automática de memória por garbage colletor além de várias de suas abstrações tais como classes, interfaces, delegados e exceções são nada mais que a exposição explicita recursos do ambiente .NET.

Quando comparada com C e C++, a linguagem é restrita e melhorada de várias formas incluindo:

* Ponteiros e aritmética sem checagem só podem ser utilizados em uma modalidade especial chamada modo inseguro (unsafe mode). Normalmente os acessos a objetos são realizados através de referências seguras, as quais não podem ser invalidadas e normalmente as operações aritméticas são checadas contra sobrecarga (overflow).
* Objetos não são liberados explicitamente, mas através de um processo de coleta de lixo (garbage collector) quando não há referências aos mesmos, prevenindo assim referências inválidas.
* Destrutores não existem. O equivalente mais próximo é a interface Disposable, que juntamente com a construção using block permitem que recursos alocados por um objeto sejam liberados prontamente. Também existem finalizadores, mas como em Java sua execução não é imediata.
* Como no Java, não é permitida herança múltipla, mas uma classe pode implementar várias interfaces abstratas. O objetivo principal é simplificar a implementação do ambiente de execução.

* C# é mais seguro com tipos que C++. As únicas conversões implícitas por default são conversões seguras, tais como ampliação de inteiros e conversões de um tipo derivado para um tipo base. Não existem conversões implícitas entre inteiros e variáveis lógicas ou enumerações. Não existem ponteiros nulos (void pointers) (apesar de referências para Object serem parecidas). E qualquer conversão implícita definida pelo usuário deve ser marcada explicitamente, diferentemente dos construtores de cópia de C++.
* A sintaxe para a declaração de vetores é diferente ("int[] a = new int[5]" ao invés de "int a[5]").
* Membros de enumeração são colocados em seu próprio espaço de nomes (namespace)
* C# não possui modelos (templates), mas C# 2.0 possui genéricos (generics).
* Propriedades estão disponíveis, as quais permitem que métodos sejam chamados com a mesma sintaxe de acesso a membros de dados.
* Recursos de reflexão completos estão disponíveis

Apesar de C# ser frequentemente tido como similar a Java, existem uma série de diferenças importantes, mas a maioria é implementada de forma diferenciada em ambas as linguagens.

Por exemplo, o Java não implemente propriedades, mas permite a utilização de métodos Get e Set que realizam o mesmo processo.

Bibliotecas de código

Ao contrário das outras linguagens de programação, nenhuma implementação de C# atualmente inclui qualquer conjunto de bibliotecas de classes ou funções. Ao invés disso, C# está muito vinculada ao framework .Net/[Mono (projeto)|Mono]], do qual C# obtém suas classes ou funções de execução. O código é organizado em um conjunto de namespaces que agrupam as classes com funções similares. Por exemplo: System.Drawing para gráficos, System.Collections para estrutura de dados e System.Windows.Forms para o sistema Windows Form.

Um nível de organização superior é fornecido pelo conceito de montador (assembly). Um montador pode ser um simples arquivo ou multiplos arquivos ligados juntos (como em al.exe) que podem conter muitos namespaces ou objetos. Programas que precisam de classes para realizar uma função em particular podem se referenciar a montadores como System.Drawing.dll e System.Windows.Forms.dll assim como a biblioteca core (conhecida como mscorlib.dll na implementação da Microsoft).

Exemplo Olá Mundo

Segue abaixo um pequeno exemplo de programa C#:

```
1: public class ClasseExemplo
2: {
3:     public static void Main()
4:     {
5:         System.Console.WriteLine("Olá mundo!");
6:     }
7: }
```

O código acima escreve o texto Olá mundo! na console. Agora vamos examiná-lo linha por linha:

1: public class ClasseExemplo

Esta linha define a classe ClasseExemplo como pública (public), ou seja, objetos em outros projetos podem utilizar esta classe livremente.

3: public static void Main()

Este é o ponto de entrada do programa quando executado a partir da console. Este método também pode ser chamado de outro código utilizando-se a sintaxe ClasseExemplo.Main(). A definição public static void indica que o método Main é público (public), que pode ser acessado diretamente através da classe (static) e que não retorna nenhum valor (void).

5: System.Console.WriteLine("Olá mundo!");

Esta linha escreve a mensagem na console. Console é um objeto do sistema que representa a linha de comando, e através da qual o programa pode obter e mostrar texto. O método WriteLine (EscreverLinha) de Console é executado, acarretando que o literal passado como parâmetro seja mostrado na console.