**Resumo**

**introducao**

**1 legibilidade**

**2 redigibilidade**

**3 confiabilidade -----femis**

**4 eficiencia**

**5 facilidade de aprendizado**

**6 ortogonalidade**

**7 reusabilidade**

**8 modificabilidade**

**9 portabilidade -------- Jean**

**10- references**

Resumo:

Faz-se necessario que um programador conheca as principais caracteristicas e paradigmas que regem uma Linguagem de programacao. Logo, estudando a fundo a linguagem de programação C#, do paradigma orientado a objetos, observamos varias implementacoes de propriedades desejáveis a uma linguagem de programacao estao presentes nessa Linguagem. O C# tem raízes em C, C++ e Java, adaptando os melhores recursos de cada linguagem e acrescentando novas capacidades próprias. Uma linguagem fortemente tipada que permite que os desenvolvedores criem uma variedade de aplicativos robustos e seguros executados no .NET Framework.

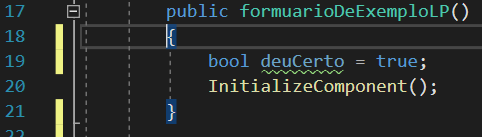
Introducao:

O C# é uma linguagem fortemente tipada e permite compilar programas orientados a objetos de forma produtiva, permitindo também que os desenvolvedores usem o Windows, a Web e os dispositivos móveis como destino. Como C# é uma linguagem orientada a objetos ela suporta conceitos como: Encapsulamento, Heranca e polimorfismo. Ela foi desenvolvida pela Miscrosoft juntamente com a arquitetura da plataforma .NET Framework (fica a cargo do controle de memória), o principal envolvido no projeto foi o engenheiro Anders Hejlsberg, que além do C# foi criador do Turbo Pascal e do [Delphi](https://www.devmedia.com.br/guia/delphi/38186).

Sua sintaxe é simples e de fácil aprendizagem, muito familiarizada com C e Java. Fornecendo recursos como: tipos de valor nulo, enumeracoes, delegacoes expressoes labidas e acesso direto a memória, possui suporte a DLL's. É case-sensitive e suas classes podem implementar várias interfaces.

**1 legibilidade:**

A Legibilidade corresponde à facilidade para se ler e entender um programa. C# por ser fortemente tipada, operações entre variáveis de tipos de diferentes ocasionam erros. C# não permite a conhecida “programação não estruturada” (uso de goto), isso colabora para uma melhor legibilidade do programa. A linguagem de programação utiliza como demarcadores de Blocos as chaves {}, usando a IDE do Visual Studio, facilita ainda mais a compreensão de onde começa e onde acaba aquele bloco, quando se é selecionado uma chaves({), a IDE marca qual é o par. Essa funcionalidade é um auxílio para melhorar a legibilidade. Segue exemplo abaixo:



**Exemplo 1.0 Auxílio a Legibilidade usando a IDE Visual Studio.**

Note que: ao selecionar a chave({) que abre o bloco, ele automaticamente sinaliza o seu par, ou seja, mostra ao programador qual é a chave de fechamento (}).

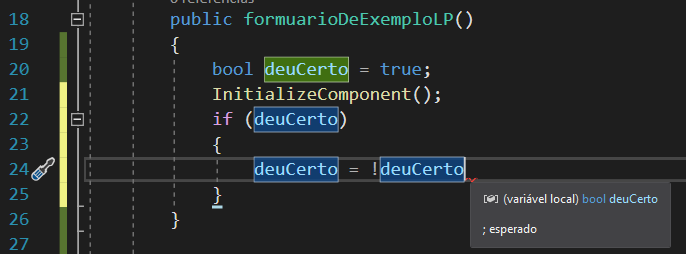
**2 redigibilidade:**

Essa propriedade possibilita ao programador se concentrar nos algoritmos centrais do programa. A redigibilidade de programas pode conflitar com a legibilidade já que está ligada a escrita do programa. O problema se encontra na redação de comandos complexos, mas que podem não identificar de maneira muito clara a sua funcionalidade, dificultando a legibilidade do programa. C# tem tratamento de erros, uma funcionalidade que tende a ajudar bastante quanto a redigibilidade. A Redigibilidade é baseada em C++ e Java. Por exemplo, já existem várias funções previamente desenvolvidas e tem uma vasta gama de tipos de variáveis, diferente de uma linguagem como Fortran por exemplo que não possui registros, armazenar dados requer a criação de vetores específicos para cada tipo de dado. Segue abaixo uma lista dos tipos de dados que temos em C#, (Lista retirada da documentação oficial da Microsoft sobre a Linguagem):

* + - Integral com sinal: sbyte, short, int,long
    - Integral sem sinal: byte, ushort, uint,ulong
    - Caracteres Unicode: char
    - Ponto flutuante binário de IEEE: float, double
    - Ponto flutuante decimal de alta precisão: decimal
    - Booliano: bool
    - Tipos definidos pelo usuário do formulário enum E {...}
    - Tipos definidos pelo usuário do formulário struct S {...}
    - Extensões de todos os outros tipos de valor com um valor null
    - Classe base definitiva de todos os outros tipos: object
    - Cadeia de caracteres Unicode: string
    - Tipos definidos pelo usuário do formulário class C {...}
    - Tipos definidos pelo usuário do formulário interface I {...}
    - Unidimensional e multidimensional, por exemplo, int[] e int[,]
    - Tipos definidos pelo usuário do formulário delegate int D(...)

**3 Confiabilidade:**

São confiáveis as Linguagens de programação que requerem a declaração de dados permitem verificar automaticamente erros de tipos durante compilação ou execução, para aumentar mais a confiabilidade algumas linguagens possuem tratamento de erro. O caso do C# é uma linguagem fortemente tipada e que possue tratamento de execessão, e para erros de sintaxe, o programador é avisado ainda enquanto escreve o código, podendo debuga-lo ao mesmo tempo que escreve. Segue abaixo um Exemplo de erro de sintaxe em C# usando a IDE Visual Studio:



**Exemplo 2.0 Auxílio à Detecção de erros de Sintaxe usando a IDE Visual Studio.**

Note que na linha 24 a IDE gera uma mensagem diz que é esperado que ao final da linha exista o separador de comandos “;”. Logo, auxilia na detecção de erros de sintaxe no código, trazendo mais confiabilidade à Linguagem C#.

**4 eficiencia**

A linguagem de Programação C# é extremamente eficiente, por ser muito versátil e realizar uma gama extensa de funcionalidades com segurança, como: desenvolver jogos, viculação com banco de dados, programação orientada a objetos, permitir aplicações completas e criação de APIs, desenvolvimento web usando framework ASP .NET e desenvolvimento mobile. Por essas razões C# é muito utilizada em aplicações comerciais reais, para programas completos e, se necessário for, complexos. A própria Microsoft(desenvolvedora da linguagem) mantém um guia detalhado em sua documentação sobre como otimizar a criação de códigos seguros e eficientes. Acessado na data de: 29/10/2019 e desponibilizado no site: <https://docs.microsoft.com/pt-br/dotnet/csharp/write-safe-efficient-code>

**5 Facilidade de aprendizado**

No quesito facilidade de aprendizado, para novos programadores que não possuem conhecimento em Programação orientada a objetos, se torna uma tarefa razoavelmente complexa aprender C#, pois por a linguagem possui muitas funcionalidades, logo também possui várias maneiras de escrever a mesma instrução, o que faz com que cada programador

C# é uma linguagem elegante, orientada a objeto e fortemente tipada, que permite que os desenvolvedores criem uma variedade de aplicativos robustos e seguros executados no .NET Framework. Você pode usar C# para criar aplicativos de cliente do Windows, serviços Web XML, componentes distribuídos, aplicativos cliente-servidor, aplicativos de banco de dados e muito, muito mais. O Visual C# fornece um editor de código avançado, designers de interface do usuário convenientes, depurador integrado e muitas outras ferramentas para facilitar o desenvolvimento de aplicativos com base na linguagem C# e no .NET Framework.

A sintaxe de C# é altamente expressiva, mas também é simples e fácil de aprender. A sintaxe de chaves de C# será instantaneamente reconhecível para qualquer pessoa familiarizada com C, C++ ou Java. Normalmente, os desenvolvedores que conhecem qualquer uma dessas linguagens são capazes de começar a trabalhar de forma produtiva em C# dentro de um período muito curto. A sintaxe de C# simplifica muitas complexidades de C++ e fornece recursos poderosos como tipos de valor anulável, enumerações, delegados, expressões lambda e acesso direto à memória, que não existem em Java. C# oferece suporte a tipos e métodos genéricos, o que proporciona mais segurança e desempenho para os tipos, e iteradores, que permitem aos implementadores das classes de coleção definir os comportamentos personalizados da iteração simples de usar pelo código do cliente. As expressões LINQ (Consulta Integrada à Linguagem) tornam a consulta fortemente tipada uma construção de linguagem de primeira classe.

Por ser uma linguagem orientada a objeto, o C# oferece suporte aos conceitos de encapsulamento, herança e polimorfismo. Todas as variáveis e métodos, incluindo o método Main, o ponto de entrada do aplicativo, são encapsulados em definições de classe. Uma classe pode herdar diretamente de uma classe pai, mas pode implementar qualquer quantidade de interfaces. Métodos que substituem métodos virtuais em uma classe pai exigem a palavra-chave override como uma forma de evitar uma redefinição acidental. Em C#, um struct é como uma classe simplificada; é um tipo alocado na pilha que pode implementar interfaces, mas não oferece suporte a herança.

Além desses princípios básicos orientados a objeto, C# facilita o desenvolvimento de componentes de software por meio de várias construções de linguagem inovadoras, incluindo o seguinte:

* Assinaturas de método encapsulado chamadas de *delegados*, que permitem as notificações de eventos fortemente tipados.
* Propriedades, que servem como acessadores para variáveis de membro privado.
* Atributos, que fornecem metadados declarativos sobre os tipos no tempo de execução.
* Comentários embutidos da documentação XML.
* LINQ (Consulta Integrada à Linguagem) que fornece recursos de consulta internos em várias fontes de dados.

Se você precisar interagir com outros softwares do Windows, como objetos COM ou DLLs Win32 nativas, faça isso em C# através de um processo denominado "Interoperabilidade". A interoperabilidade permite que programas em C# façam quase tudo que um aplicativo C++ nativo pode fazer. C# oferece suporte até mesmo para ponteiros, e o conceito de código "não seguro" para os casos nos quais o acesso direto à memória é absolutamente essencial.

O processo de compilação de C# é simples comparado ao C e C++, e mais flexível do que em Java. Não há arquivos de cabeçalho separado, e nenhum requisito de que os métodos e os tipos sejam declarados em uma ordem específica. Um arquivo de código-fonte de C# pode definir qualquer quantidade de classes, estruturas, interfaces e eventos.

Resumo em Ingles:

--------------------------------------------------------------------------------------------------------------------------

* Simplicidade: os projetistas de C# costumam dizer que essa linguagem é tão poderosa quanto o C++ e tão simples quanto o *Visual Basic*;
* Completamente orientada a objetos: em C#, qualquer variável tem de fazer parte de uma classe;
* Fortemente tipada: isso ajudará a evitar erros por manipulação imprópria de tipos e atribuições incorretas;
* Gera código gerenciado: assim como o ambiente .NET é gerenciado, assim também é o C#;
* Tudo é um objeto: *System.Object* é a classe base de todo o sistema de tipos de C#;
* Controle de versões: cada *assembly* gerado, seja como *EXE* ou *DLL*, tem informação sobre a versão do código, permitindo a coexistência de dois assemblies homônimos, mas de versões diferentes no mesmo ambiente;
* Suporte a código legado: o C# pode interagir com código legado de objetos COM e DLLs escritas em uma linguagem não-gerenciada;
* Flexibilidade: se o desenvolvedor precisar usar ponteiros, o C# permite, mas ao custo de desenvolver código não-gerenciado, chamado “*unsafe*”;
* Linguagem gerenciada: os programas desenvolvidos em C# executam num ambiente gerenciado, o que significa que todo o gerenciamento de memória é feito pelo *runtime* via o *GC (Garbage Collector*).[[19]](https://pt.wikipedia.org/wiki/C_Sharp#cite_note-19)

Portabilidade:

Por design, C# é a linguagem de programação que reflete mais diretamente a infraestrutura de linguagem comum (CLI).[[32]](https://pt.wikipedia.org/wiki/C_Sharp#cite_note-32) A maioria dos seus tipos intrínsecos corresponde a tipos de valor implementados pelo framework CLI. No entanto, a especificação de linguagem não indica os requisitos de geração de código do compilador: ou seja, não indica que um compilador C# deve segmentar um Common Language Runtime, ou gerar Common Intermediate Language (CIL) ou gerar qualquer outro formato específico. Teoricamente, um compilador C# poderia gerar código de máquina como compiladores tradicionais de C++, Pascal ou Fortran. De fato há iniciativas para que isso se torne realidade.[[33]](https://pt.wikipedia.org/wiki/C_Sharp#cite_note-33)

<https://pt.slideshare.net/cloudson/c-net-um-overview-da-linguagem>

<https://imasters.com.br/back-end/7-erros-que-todo-programador-csharp-deve-evitar>

<https://docs.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/types-and-variables>

livro varejola

<https://docs.microsoft.com/pt-br/dotnet/csharp/write-safe-efficient-code>