Fundamentos de C#

Capítulo 1. O que é C# e .NET.

Prof. Samuel Santos

Fundamentos de C#

Aula 1.1. O que é C# e .NET.

Prof. Samuel Santos

Nesta aula

- O que é C# e .NET.

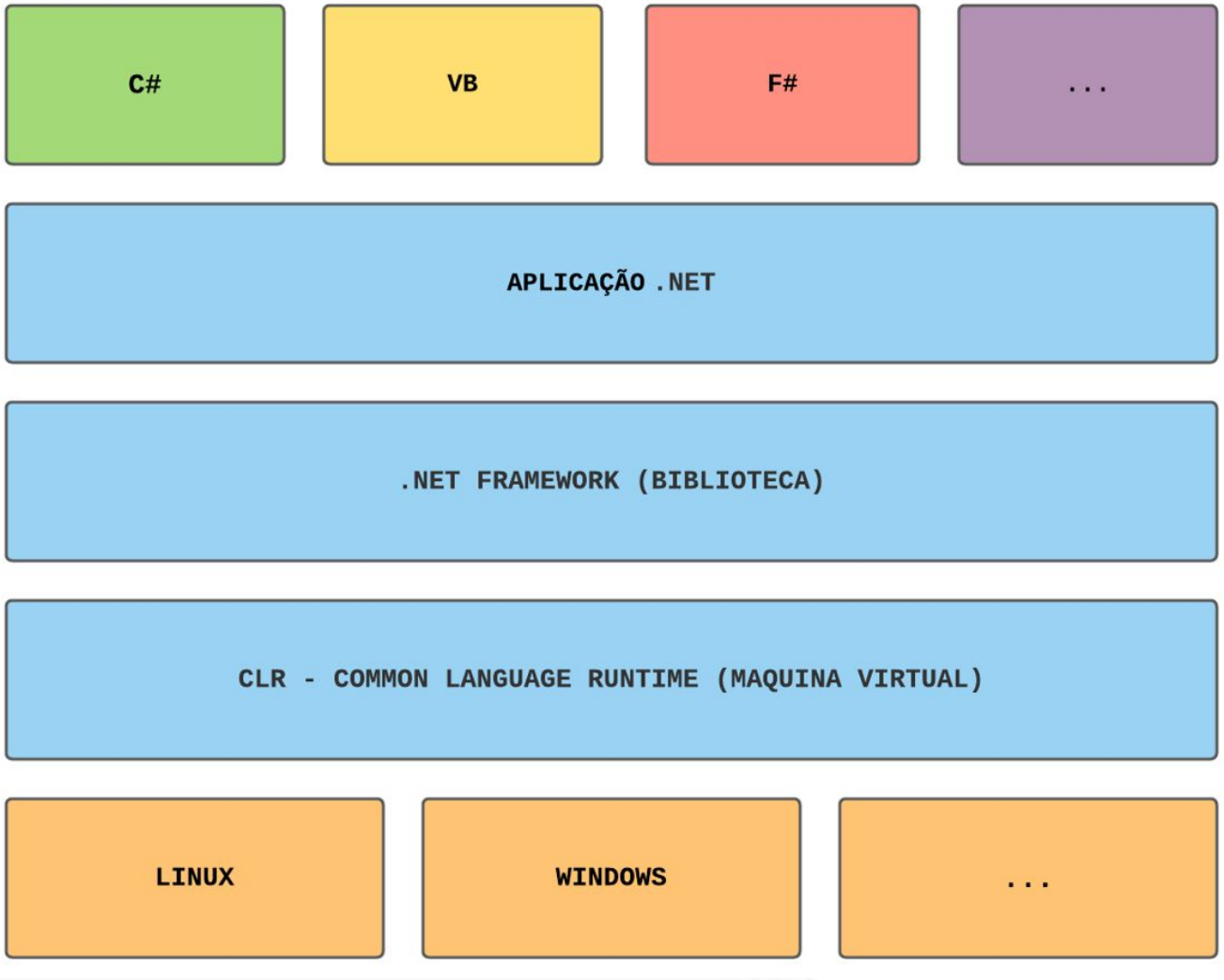
Contextualizando C# e .NET

- C # é parte do conjunto de ferramentas desenvolvida dentro da iniciativa .NET da Microsoft.
- C # é uma linguagem de programação orientada a objetos, simples, robusta, fortemente tipada e escalável.
- O .NET framework é uma plataforma que permite escrever aplicativos em diferentes contextos (web, desktop, mobile e IoT).
- A estrutura .NET consiste em uma enorme biblioteca de código usada pela linguagem cliente, como C #. Em outras palavras, já existem muitos recursos integrados ao C # antes mesmo de você começar!

Máquina Virtual

- Common Language Runtime (CLR) - camada intermediária.
- Common Intermediate Language (CLI) - código binário (código de máquina).
- Manipulação de exceção, segurança de tipo, gerenciamento de memória (usando o Garbage Collector), segurança, performance melhorada, independência de idioma, independência de plataforma e independência da arquitetura.

Máquina Virtual



Fundamentos de C#

Capítulo 2. O Ambiente de Desenvolvimento do C#

Prof. Samuel Santos

Fundamentos de C#

Aula 2.1. O Ambiente de Desenvolvimento do C#

Prof. Samuel Santos

Nesta aula

- O Ambiente de Desenvolvimento do C#.

Ambientes de Desenvolvimento Integrado (IDE)

- Visual Studio (versão completa) é um ambiente de desenvolvimento "completo" e "conveniente".
- Community é uma versão simplificada da versão completa e substitui como edições expressas usadas antes de 2015.
- É uma solução completa usada principalmente por desenvolvedores .NET e relacionados. Inclui tudo, desde controle de origem a ferramentas de rastreamento de bugs, etc. Tem tudo o que é necessário para desenvolver.

Ambientes de Desenvolvimento Integrado (IDE)

- É um editor de código-fonte leve que pode ser usado para visualizar, editar, executar e depurar códigos-fonte para aplicativos.
- Simplesmente é o Visual Studio sem a interface visual.
- É principalmente orientado em torno de arquivos, não projetos.
- Não tem suporte para o sistema de controle de versão da Microsoft; Team Foundation Server.
- É usado principalmente por desenvolvedores em um Mac que lida com as tecnologias do lado do cliente (HTML, JavaScript e CSS).

Ambientes de Desenvolvimento Integrado (IDE)

- IntelliSense avançado. Com o poder do Roslyn
- Visual Studio for Mac traz o IntelliSense
- Depurador efetivo
- Refatoração Inteligente
- Controle de fonte integrado
- Testes

O Primeiro Programa C#

```
using System;

namespace FundamentosCSharp
{
    0 references
    class Program
    {

        0 references
        static void Main(string[] args)
        {
            int contador = 10;

            while (contador < 10)
            {
                Console.WriteLine($" Contador atual .: {contador}");
                contador++;
            }
        }
    }
```

Fundamentos de C#

Capítulo 3. Variáveis e Tipos Primitivos

Prof. Samuel Santos

Fundamentos de C#

Aula 3.1. Variáveis e Tipos Primitivos

Prof. Samuel Santos

Nesta aula

- Variáveis e Tipos Primitivos.

Variáveis

Toda variável é uma alocação de uma quantidade de memória, é nesse espaço de memória que está armazenado o conteúdo da variável, internamente uma variável possui um ponteiro, o ponteiro para o sistema operacional é um endereçamento físico de memória, serve para localizar onde está armazenado tal dado.

Nomeando uma variável

- A regra para nomear uma variável é que o nome dela sempre comece por uma letra ou _.
- Não crie variáveis que apenas 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;
- Para usar esse tipo de convenção, capitalize a primeira letra de cada palavra menos da primeira.

Declarando uma variável

```
1 | int numero;
```

Tipos Valor

- Uma variável deste tipo contém o valor, e não um endereço de referência para o valor;
- Derivam de System.ValueTypes;
- Variáveis de escopo local precisam ser inicializadas antes de serem utilizadas;
- Atribuir o valor de variável a outra, implicitamente, é feita uma cópia do conteúdo da variável. Sendo assim, qualquer alteração no conteúdo de uma delas, não afetará a outra. Quanto maior for um objeto deste tipo mais custosa será sua cópia.

Tipos Primitivos

Tipos primitivos do C#	Tipo Framework Class Library	Descrição
sbyte	System.SByte	Valor a 8 bits com sinal
byte	System.Byte	Valor a 8 bits sem sinal
short	System.Int16	Valor a 16 bits com sinal
ushort	System.UInt16	Valor a 16 bits sem sinal
int	System.Int32	Valor a 32 bits com sinal
uint	System.UInt32	Valor a 32 bits sem sinal
long	System.Int64	Valor a 64 bits com sinal
ulong	System.UInt64	Valor a 64 bits sem sinal
char	System.Char	Caracter Unicode a 16 bits
float	System.Single	IEEE 32 bits
double	System.Double	IEEE 64 bits
bool	System.Boolean	1 bit (True ou False)
decimal	System.Decimal	Float 128 bits (nao e primitivo no CTS)

Tipos de Dados

- Tipos Valor
- Tipos Referência

Modificadores de Acesso

Modificador	Funcionamento
public	O acesso não é restrito.
protected	O acesso é limitado às classes ou tipos derivados daclasse que a variável está.
Internal	O acesso é limitado ao conjunto de módulos(assembly) corrente.
protected internal	O acesso é limitado ao conjunto corrente ou tiposderivados da classe recipiente.
private	O acesso é limitado à classe que a variável está.

Palavras reservadas (keywords)

abstract	as	base	Bool
break	byte	case	Catch
char	checked	class	Const
continue	decimal	default	Delegate
do	double	else	Enum
event	explicit	extern	false
finally	fixed	float	for
foreach	goto	if	implicit
in	int	interface	internal
is	lock	long	namespace
new	null	object	operator
out	override	params	private
protected	public	readonly	ref
return	sbyte	sealed	short
sizeof	stackalloc	static	string
struct	switch	this	throw

Enum (Enumeradores)

Um enum é uma "classe" especial que representa um grupo de constantes (variáveis inalteráveis / somente leitura).

Struct (Estrutura)

A estrutura C # é um tipo simples definido pelo usuário, uma alternativa leve para uma classe. Um struct em C # é simplesmente um tipo de dados composto que consiste em vários elementos de outros tipos.

Inferência de Tipos

Operações com Variáveis

Variáveis do Tipo String

Em C #, uma string é uma série de caracteres usados para representar texto. Pode ser um caractere, uma palavra ou uma longa passagem entre aspas duplas ". Os seguintes são literais de string.

Interpolação de Strings

Comentários, como aplicá-los?

Fundamentos de C#

Capítulo 4. Estruturas de Controle

Prof. Samuel Santos

Fundamentos de C#

Aula 4.1. Estruturas de Controle

Prof. Samuel Santos

Nesta aula

- Estruturas de Controle.

If/Else

- Use a instrução if para especificar um bloco de código C # a ser executado se uma condição for verdadeira.

- Use a instrução else para especificar um bloco de código a ser executado se a condição for falsa.

Switch

Em C #, a instrução Switch é uma instrução de ramificação de várias vias. Ele fornece uma maneira eficiente de transferir a execução para diferentes partes de um código com base no valor da expressão. A expressão switch é do tipo inteiro, como int, char, byte ou short, ou de um tipo de enumeração, ou do tipo string. A expressão é verificada para casos diferentes e uma correspondência é executada.

Fundamentos de C#

Capítulo 5. Estruturas de Repetição

Prof. Samuel Santos

Fundamentos de C#

Aula 5.1. Estruturas de Repetição

Prof. Samuel Santos

Nesta aula

- Estruturas de Repetição.

Estruturas de Repetição

Quando precisamos executar um bloco de código repetidas vezes devemos recorrer às estruturas de repetição. Assim, conseguimos programar o código desejado sem que para isso criamos cópias desse mesmo conjunto de instruções. Com a linguagem C# temos quatro opções para implementar estruturas de repetição: For, While, Do...While e Foreach.

For

O For é uma estrutura de repetição que utiliza quando sabemos a quantidade de repetições que um bloco de código deve ser desenvolvido.

While

O enquanto trata-se da estrutura de repetição mais aproveitada quando programamos com C #. Com ela, enquanto a condição para a verdadeira, o bloco de código será evoluído. Primeiramente o sistema testa essa condição. Caso verdadeira, execute as linhas declaradas dentro do while; do contrário, sai do loop.

Do...While

Esta estrutura de execução funciona de forma semelhante ao, porém, ela garante que o código dentro do loop seja executado pelo menos uma vez. Para isso, a condição é declarada após o bloco de código.

Foreach

O foreach é um recurso do C # que possibilita executar um conjunto de comandos para cada elemento presente em uma coleção (matriz, lista, pilha, fila e outros). Portanto, diferentemente faça enquanto e faça por, não precisamos definir uma condição de parada. Isso é definido de forma implícita, pelo tamanho da coleção.

Fundamentos de C#

Capítulo 6. Arrays

Prof. Samuel Santos

Fundamentos de C#

Aula 6.1. Arrays

Prof. Samuel Santos

Nesta aula

- Arrays.

O que são arrays ?

Os arrays são usados para armazenar vários valores em uma única variável, em vez de declarar variáveis separadas para cada valor.

Manipulando Arrays

Fundamentos de C#

Capítulo 7. POO - (Programação Orientada a Objetos)

Classes e Objetos

Prof. Samuel Santos

Fundamentos de C#

Aula 7.1. POO - (Programação Orientada a Objetos)

Classes e Objetos

Prof. Samuel Santos

Nesta aula

- POO - (Programação Orientada a Objetos).
- Classes e Objetos.

POO - Programação Orientada a Objetos

A programação orientada a objetos (OOP) é um paradigma de programação que usa objetos e suas interações para projetar aplicativos e programas de computador.

Existem alguns conceitos básicos de programação em OOP:

- Abstração
- Polimorfismo
- Encapsulamento
- Herança

POO - Programação Orientada a Objetos

- A abstração está simplificando a realidade complexa modelando classes apropriadas ao problema.
- O polimorfismo é o processo de usar um operador ou função de diferentes maneiras para diferentes entradas de dados.
- O encapsulamento oculta os detalhes de implementação de uma classe de outros objetos.
- A herança é uma forma de formar novas classes usando classes que já foram definidas.

Campos (Fields)

Um **campo** é uma variável declarada diretamente dentro de uma classe ou estrutura. Em geral, campos são declarados com visibilidade privada ou protegida.

Propriedades

Através de uma propriedade, que é um membro de uma classe ou estrutura com métodos específicos para acessar o campo que está escondido. Propriedades permite à classe expor uma maneira de definir seus valores enquanto esconde sua implementação.

Métodos

Um método é um bloco de código que contém uma série de instruções. Um programa faz com que as instruções sejam executadas chamando o método e especificando quaisquer argumentos de método necessários. Em C #, cada instrução executada é realizada no contexto de um método.

Construtores

Um construtor em C # é membro de uma classe. É um método na classe que é executado quando um objeto de classe é criado. Normalmente, colocamos o código de inicialização no construtor. O nome do construtor é sempre o mesmo nome da classe. Um construtor C # pode ser público, protegido ou privado. Uma classe pode ter vários construtores sobrecarregados.

Fundamentos de C#

Capítulo 8. Herança

Prof. Samuel Santos

Fundamentos de C#

Aula 8.1. Herança

Prof. Samuel Santos

Nesta aula

- Herança.

Classes e Objetos

Uma definição de classe ou estrutura é como um projeto que especifica o que o tipo pode fazer. Um objeto é basicamente um bloco de memória que foi alocado e configurado. Um programa pode criar muitos objetos da mesma classe. Os objetos também são chamados de instâncias e podem ser armazenados em uma variável nomeada ou em uma matriz ou coleção. Em uma linguagem orientada a objetos, como C #, um programa típico consiste em vários objetos interagindo dinamicamente.

Instâncias de estrutura x instâncias de classe

- Como as classes são tipos de referência, uma variável de um objeto de classe contém uma referência ao endereço do objeto no heap gerenciado. Se uma segunda variável do mesmo tipo for atribuída à primeira variável, ambas as variáveis se referem ao objeto naquele endereço.
- Instâncias de classes são criadas usando o operador *new*.

Classes Abstratas

Classes abstratas, marcadas pela palavra-chave abstract na definição de classe, são normalmente usadas para definir uma classe base na hierarquia. O que é especial sobre eles, é que você não pode criar uma instância deles - se você tentar, você receberá um erro de compilação. Em vez disso, você precisa definir subclasses e criar uma instância desta.

Classes Parciais

É possível dividir a definição de uma classe, estrutura, interface ou método em dois ou mais arquivos fonte. Cada arquivo de origem contém uma seção do tipo ou definição de método e todas as partes são combinadas quando o aplicativo é compilado.

Classes Parciais

Aqui estão várias situações em que a divisão de uma definição de classe é desejável:

- Ao trabalhar em grandes projetos, distribuir uma classe em arquivos separados permite que vários programadores trabalhem ao mesmo tempo.

- Ao trabalhar com o código-fonte gerado automaticamente, o código pode ser adicionado à classe sem a necessidade de recriar o arquivo-fonte. O Visual Studio usa essa abordagem ao criar Formulários do Windows, código de wrapper de serviço da Web e assim por diante. Você pode criar o código que usa essas classes sem precisar modificar o arquivo criado pelo Visual Studio.

- Ao usar geradores de origem para gerar funcionalidade adicional em uma classe.

Classes Seladas

A classe selada, por sua vez, impede que outras classes herdem dela. É o oposto da abstrata, onde precisa de herdeiros para fazer sentido.

Fundamentos de C#

Capítulo 9. Polimorfismo

Prof. Samuel Santos

Fundamentos de C#

Aula 9.1. Polimorfismo

Prof. Samuel Santos

Nesta aula

- Polimorfismo.

Polimorfismo

Polimorfismo significa que existem múltiplas formas. No paradigma de programação orientada a objetos, polimorfismo é muitas vezes expresso como "uma interface, múltiplas funções."

Polimorfismo com Herança

Através da herança, é possível ao polimorfismo permitir que as classes sobrescrevam os membros da classe base, isso possibilita a vinculação dinâmica.

Polimorfismo com Classe Abstrata

Como uma classe abstrata não pode ser instanciada, os recursos das interfaces e da herança entram em jogo com a programação polimórfica.

Polimorfismo com Interfaces

Sabemos que várias classes podem implementar a mesma interface ou implementar mais de uma interface. A interface descreve os métodos e os tipos de parâmetros que cada membro do método precisa receber e retornar, mas deixa os detalhes do corpo do método para as classes que implementam a interface definida.

Fundamentos de C#

Capítulo 10. Interfaces

Prof. Samuel Santos

Fundamentos de C#

Aula 10.1. Interfaces

Prof. Samuel Santos

Nesta aula

- Interfaces.

O que são interfaces?

Outra forma de obter abstração em C # é com interfaces.

Uma interface se parece com uma classe, mas não tem implementação. A única coisa que contém são declarações de eventos, indexadores, métodos e / ou propriedades. A razão pela qual as interfaces fornecem apenas declarações é porque elas são herdadas por estruturas e classes, que devem fornecer uma implementação para cada membro da interface declarado.

Trabalhando com Interfaces

Fundamentos de C#

Capítulo 11. Listas

Prof. Samuel Santos

Fundamentos de C#

Aula 11.1. Listas

Prof. Samuel Santos

Nesta aula

- Listas.

O que são Listas?

A classe List representa a lista de objetos que podem ser acessados por índice. Ele vem com o namespace System.Collection.Generic. A classe List pode ser usada para criar uma coleção de diferentes tipos, como inteiros, strings etc. A classe List <T> também fornece os métodos para pesquisar, classificar e manipular listas.

Inicializando Listas

Manipulando Listas

Fundamentos de C#

Capítulo 12. Membros Estáticos (Static)

Prof. Samuel Santos

Fundamentos de C#

Aula 12.1. Membros Estáticos (Static)

Prof. Samuel Santos

Nesta aula

- Membros Estáticos (Static).

Classes Estáticas (Static)

As classes estáticas são aquelas que selecionam apenas membros estáticos, além de que elas não podem ser instanciadas. Quando o programa ou o namespace que possui classe estática carregado, o CLR (Common Language Runtime) do .NET Framework carrega automaticamente como classes estáticas.

Campos e Métodos Estáticos (Static)

O modificador static pode ser usando em um campo ou em um método de determinada classe para informar que aquele campo ou método existe independentemente do objeto.

Construtores Estáticos (Static)

Um construtor estático é usado para inicializar qualquer dado estático ou para executar uma ação específica que precisa ser executada apenas uma vez. Ele é chamado automaticamente antes que a primeira instância seja criada ou qualquer membro estático seja referenciado.

Fundamentos de C#

Capítulo 13. Exceções (Exceptions)

Prof. Samuel Santos

Fundamentos de C#

Aula 13.1. Exceções (Exceptions)

Prof. Samuel Santos

Nesta aula

- Exceções (Exceptions).

O que são exceções (Exceptions)?

Uma exceção é um erro em tempo de execução em um programa, que viola uma condição que não foi especificada para acontecer durante a operação normal. Um exemplo na prática é quando um programa tenta fazer a divisão por zero ou tenta escrever em um arquivo somente leitura. Quando isso ocorre, o sistema pega o erro e lança uma exceção.

Trabalhando com Exceções (Exceptions)