

LP2 – Aula 03

TEÓRICA

Profª Mª Denilce Veloso
denilce.veloso@fatec.sp.gov.br
denilce@gmail.com

C# - Indentação e Blocos de Códigos

- **Indentação do Código**

As teclas de atalho para indentar o código são:

CTRL + k, F -> Formata o que estiver selecionado.

CTRL + k, d -> Formata o documento inteiro.

Utilizar Tab ao invés de espaços.

- **Blocos de Códigos**

Blocos de código são agrupados entre chaves { }.

Ex.:

```
if (textBox1.Text=="")
{
    MessageBox.Show("Nome vazio!");
    //ou
    MessageBox.Show("campo nome vazio!", "Informação",
    MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
```

C# - Ponto e Vírgula, Comentários e Case Sensitive

- **Uso do ponto e vírgula (;)**

Cada linha de código é separada por ponto-e-vírgula.

- **Comentários**

Os comentários de linha simples começam com duas barras (//).

Comentários em bloco são feitos usando os terminadores /* (de início) e */ (de fim).

```
/* Este é um comentário */
```

- **Case Sensitive**

O C# é sensível ao contexto, portanto as variáveis x e X são duas coisas diferentes.

C# - Declaração (Criação) de Variáveis / Objetos

Para criar uma variável de memória basta usar a palavra que representa o tipo de dado seguida pelo nome da variável. Exemplo:

```
int idade;
```

É possível atribuir o valor para a variável durante a sua criação usando a linha da declaração ou criar várias ao mesmo tempo. Assim:

```
int idade = 20;
```

```
string nome, sobrenome;
```

```
string faculdade1, faculdade2="Fatec"; → só coloca dado na faculdade2
```

```
string faculdade1="Fatec", faculdade2="Fatec". → coloca dado nas duas variáveis
```

É possível declarar e criar um objeto do tipo de dado escolhido ou criado, combinando a palavra chave new. Exemplo:

```
<data-type> <objeto> = new <data-type>([parâmetros]);
```

```
ListItem objItem = new ListItem();
```

```
Aluno objAluno = new Aluno();
```

C# - Codificação e Nomenclatura de Objetos e Variáveis

A Especificação da Linguagem C# não define um padrão de codificação. No entanto, a Microsoft sugere algumas diretrizes.

As convenções de codificação atendem às seguintes finalidades:

- Criam uma **aparência consistente para o código**, para que os leitores possam se concentrar no conteúdo e não no layout.
- Permitem que **os leitores entendam o código com mais rapidez**, fazendo suposições com base na experiência anterior.
- Facilitam a cópia, a alteração e a **manutenção do código**.
- Demonstram as **práticas recomendadas do C#**.

C# - Codificação e Nomenclatura de Objetos e Variáveis

Algumas sugestões / recomendações:

- a) Cuidado para não criar variáveis com nomes de palavras reservadas, veja a lista abaixo:

abstract	enum	let	sizeof
add	event	lock	stackalloc
alias	exception	long	static
as	explicit	namespace	string
ascending	extern	new	struct
base	false	null	switch
bool	finally	object	this
break	fixed	operator	throw
byte	float	orderby	true
case	for	out	try
catch	foreach	override	typeof

char	from	params	uint
checked	get	partial	ulong
class	global	private	unchecked
const	goto	protected	unsafe
continue	group	public	ushort
decimal	if	readonly	using
default	implicit	ref	value
delegate	in	remove	var
descending	int	return	virtual
do	interface	sbyte	void
double	internal	sealed	volatile
dynamic	into	select	where
else	is	set	while
	join	short	yield

C# - Codificação e Nomenclatura de Objetos e Variáveis

b) Dê um **nome significativo** para suas variáveis e evite abreviações:

Recomendado: `string nomeCompleto;`
`DateTime dataDeNascimento;`

Não recomendado: `string nomComp;`
`DateTime datNasc;`

c) **Não use caracteres simples** para nomear suas variáveis, como por exemplo `i`, `n`, `s`, etc. Ao invés disto, use uma palavra como `index`, `número`, etc. Uma exceção, seria no uso de loops:

```
for (i == 0; i < 10 ; i++)  
{  
}
```

d) **Não iniciar com número, hífen, caracteres especiais (causará erro).**
Pode usar underline (“_”) mas não é recomendado.

C# - Codificação e Nomenclatura de Objetos e Variáveis

- e) É possível utilizar underline (“_”) mas não é recomendado para variáveis locais;.
- f) **Não utilize acentos** (embora seja possível).
- g) Variáveis booleanas deverão receber nomes que impliquem em verdadeiro ou falso: Ex.: bool existeTitulo;
- h) Quando for criar uma Solution que tem vários projetos o ideal é que o projetos contendo o nome da Solução mais o nome do Projeto, exemplo: Nome da solução: PontoDeVenda Nome dos projetos: PontoDeVenda.Repositorio, PontoDeVenda.Webform, PontoDeVenda.TestesUnitarios.
- i) Portanto é importante basear-se nos padrões de nomenclatura – Naming Guide C# .Net. Os estilos de nomenclaturas que foram documento, baseiam-se nos Capitalization Styles da Microsoft. Serão utilizados o **Pascal Casing e Camel Casing**.

C# - Codificação e Nomenclatura de Objetos e Variáveis

- PASCAL CASE

A primeira letra do identificador e primeira letra de cada palavra concatenada em maiúsculo. Usar Pascal Case para:

- Nome de classes e propriedades/métodos:

```
public class Pessoa
{
    private int id;
    private string nome;
    // propriedade
    public int Id {get; set;}
    public string Nome {get; set;}
}
```

- Nome de métodos:

```
public void RealizarVenda()
{ }
```

C# - Codificação e Nomenclatura de Objetos e Variáveis

- CAMEL CASE

A primeira letra minúscula e cada palavra concatenada em maiúscula. Use Camel Case para:

- Nome de variáveis:

```
int contagemTotal = 0;
```

- Parâmetros dos métodos:

```
private int SomarNumeros(int primeiroNumero, int segundoNumero)  
{  
    int valorDaSoma = primeiroNumero + segundoNumero;  
    return valorDaSoma;  
}
```

C# - Codificação e Nomenclatura de Objetos e Variáveis

Variáveis Locais ou Globais

Em C#, todas as variáveis são declaradas dentro do escopo de uma classe e podem ser dos seguintes tipos:

- **Locais**: são declaradas no escopo de um método, indexador ou evento e não possuem modificadores de acesso. A sua declaração se limita ao tipo seguido do identificador da variável.
- **Globais**: são declaradas no início da classe e podem ser utilizadas pelos outros métodos ou eventos.

Pode-se repetir nomes das variáveis desde que estejam em escopos diferentes. Ex.: uma no escopo global e outra no escopo local. Veja exemplo:

C# - Codificação e Nomenclatura de Objetos e Variáveis

```
namespace Aula1_LeiteVariaveisEEscopo
{
    public partial class Form1 : Form
    {
        string texto = "Global";

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string texto = "Local";
            string texto1, texto2 = "Fatec Sorocaba";

            MessageBox.Show("Global=" + this.texto + " Local=" + texto);
        }
    }
}
```

C# - Codificação e Nomenclatura de Objetos e Variáveis

Atributos

Atributos de uma classe ou campos da classe: a variável é declarada como membro de uma classe. A declaração deve ser efetuada como se segue:
[Modificadordeacesso][tipoatributo]<tipodavariável><identificador>

Ex.:

```
class Conta
{
    private int numero;
    private string titular;
    private double saldo;
}
```

C# - Codificação e Nomenclatura de Objetos e Variáveis

Uso da palavra var

É importante entender que a palavra-chave var não significa "variante" e não indica que a variável é vagamente tipada ou de associação tardia. Isso apenas significa que o compilador determina e atribui o tipo mais apropriado.

A palavra-chave var pode ser usada nos seguintes contextos:

- Em variáveis locais (variáveis declaradas no escopo do método)

// i será compilada como int

```
var i = 25;
```

// s será compilado como string

```
var s = "Oi";
```

- Em uma instrução de inicialização for.

```
for(var x = 1; x < 10; x++)
```

- Em uma instrução de inicialização foreach

```
foreach(var item in list){...}
```

C# - Sugestão de formas de Abreviação para os principais componentes de Windows Forms:

Tipo	Prefixo	Exemplo
Button	Btn	btnOk
CheckBox	Ckbox	ckbxEscolha
CheckListBox	Cklb	cklbTipos
ComboBox	Cbx	cbxTipoAcessoWeb
GridView	Grv	grvDadosCliente
GroupBox	Gbx	gbxAtividade
Label	Lbl	lblNome
ListBox	Lstbx	lstbxCategorias
MaskedTextBox	Mskbx	mskbxValor
Panel	Pnl	pnlForm
RadioButton	Rbtn	rbtnQuestoes
RichText	Rchtxt	rchtxtFrase
Table	Tbl	tblPaíses
TextBox	Txt	txtCargo

C# - Constantes

- Constantes, são identificadores, como as variáveis de memória são, mas não sofrem alterações.

```
const string pais = "Brasil";  
public const int anosFundacao= 100;  
public const string nomeFaculdade = "Fatec";
```



- Constantes dentro do escopo de uma classe pode-se usar o static, e pode ser usada em construtores e por padrão é privada mas pode ser pública.

```
public static string empresa = "Irmãos Silva Ltda.";
```

Exemplo em java: public static final taxa=100;



Units of Computer Memory

Measurement  :

1 bit = Binary digit

8 bits = 1 Byte

1024 byte = 1 KB

1024 KB = 1 MB

1024 MB = 1 GB

1024 GB = 1 TB

1024 TB = 1 Peta Byte

1024 PB = 1 Exa Byte

1024 EB = 1 Zetta Byte

1024 ZB = 1 Yotta Byte

1024 YB = 1 Bronto Byte

1024 BB = 1 Geop Byte

C# - Tipos de Dados (DataTypes) para Variáveis de Memória

O C# contém diversos tipos de dados entre eles:

Tipos Inteiros

Tipo	Intervalo	Tamanho (em bytes)
<u>sbyte</u>	-128 a 127	1 com sinal
<u>byte</u>	0 a 255	1 sem sinal
<u>short</u>	-32.768 a 32.767	2 com sinal
<u>ushort</u>	0 a 65.535	2 sem sinal
<u>int</u>	-2.147.483.648 a 2.147.483.647	4 com sinal
<u>uint</u>	0 a 4.294.967.295	4 sem sinal
<u>long</u>	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807	8 com sinal
<u>ulong</u>	0 a 18.446.744.073.709.551.615	8 sem sinal

Obs.: int16, int32, int64, uint16, uint32, uint64

C# - Tipos de Dados (DataTypes) para Variáveis de Memória

Tipos Ponto Flutuante

O tipo padrão para estes números é double, se não for declarado o **tipo** de uma variável, o **C#** vai inferir como double, tratando a precisão do resultado com **ponto flutuante**. Exemplo: var soma = 155.657;

Supondo o resultado obtido da divisão de 5/9 que dá o número 0,5555556 com 7 dígitos significativos quando for utilizado o tipo **float**. Já se for utilizado o tipo **double** o resultado é expresso com uma precisão de 15 dígitos: 0,555555555555556. Portanto, que deve se usar variáveis do tipo **double** para representar números reais quando a precisão do cálculo for relevante. E para cálculos que envolvam muito dinheiro ou finanças e cálculos científicos, o tipo **Decimal** deve ser utilizado, pois ele tem a **precisão adequada para evitar os erros críticos de arredondamento**.

C# - Tipos de Dados (DataTypes) para Variáveis de Memória

Tipos Ponto Flutuante

Tipo	Intervalo aproximado	Precisão	Tamanho (byte)
<u>float</u>	$\pm 1,5e-45$ para $\pm 3,4e38$	7 dígitos	4 com sinal
<u>double</u>	$\pm 5,0e-324$ para $\pm 1,7e308$	15 a 16 dígitos	8 com sinal
decimal	$10e-28 \times \pm 1.0$ para $10e28 \times \pm 7.9$	Preciso fracionário ou tipo integral que possa representar decimal números com 29 de dígitos significativos	16 com sinal

C# - Tipos de Dados (DataTypes) para Variáveis de Memória

Tipo Char

Tipo	Intervalo	Tamanho	Tipo do Framework .NET
char	U+0000 U+FFFF	a Caractere Unicode de 16 bits	<u>System.Char</u>

Obs.: Cada caractere precisa de 2 bytes para ser representado, pois o sistema usado aqui é o UNICODE. No sistema ASCII cada caractere é representado por um byte (*podemos representar apenas 256 caracteres*). Para acomodar caracteres especiais usados em outras línguas (russo, grego, japonês) e também outros símbolos o sistema UNICODE usa dois bytes para representar cada caractere. O C# usa o UNICODE para todos os caracteres.

C# - Tipos de Dados (DataTypes) para Variáveis de Memória

Outros Tipos

Outros Tipos			
Tipo de Dado	Tamanho em Bytes	Range/Intervalo	Obs.:
<u>Bool</u>	1	<u>True</u> ou False	Usada tratar valores que podem assumir Falso ou verdadeiro.
<u>DateTime</u>	8	01/01/100 até 31/12/9999	Usado no tratamento de datas e horas.

C# - Tipos de Dados (DataTypes) para Variáveis de Memória

Outros Tipos

Tipo String

Um objeto do tipo string pode ter até 2.147.483.647 de caracteres. Cada caractere ocupa 16 bits.

Embora string seja um tipo de referência (vem da classe), os operadores de igualdade (== e !=) são definidos para comparar os valores dos objetos string, não referências. Isso torna o teste de igualdade de cadeia de caracteres mais intuitivo. Por exemplo:

```
string mensagem = "alô";  
  
if (mensagem == "alô")  
{  
    MessageBox.Show("São iguais");  
}  
else  
{  
    MessageBox.Show("São diferentes");  
}
```

string nome = "Rafael";

Pode também ser criado como objeto:

```
string nome = new string(new char[]{'R','a','f','a','e','l'});
```

A diferença é que no segundo caso, a string não vai para a memória no pool de strings (para ser reaproveitado) ocupando mais memória.

Outro exemplo:
string stringona = new
string("a", 20);

C# - Tipos de Dados (DataTypes) para Variáveis de Memória

Veja na tabela abaixo os valores padrões de cada tipo.

Tipo de valor	<u>Valor</u> padrão
<u>bool</u>	False
<u>byte</u>	0
<u>char</u>	'\0'
<u>decimal</u>	0,0M (Quando declara um tipo decimal com dados será necessário colocar a letra m na frente)
<u>double</u>	0,0D
<u>float</u>	0,0F
<u>int</u>	0
<u>long</u>	0L
<u>sbyte</u>	0
<u>short</u>	0
<u>uint</u>	0
<u>ulong</u>	0
<u>ushort</u>	0
String	Nulo

C# - Tipos de Dados (DataTypes) para Variáveis de Memória

Exemplos:

```
decimal numero1 = 5.67M;  
double  numero2 = 5.67;  
float   numero3  = 5.67F;
```

C# - Tipos de Dados (DataTypes) para Variáveis de Memória

OBSERVAÇÃO: Nos tipos String, Float, Double, Char pode ser declarado o tipo iniciando com maiúscula ou minúscula, por exemplo, string é um alias (apelido) para String no .NET Framework, double para Double e assim por diante. O tipo bool é um alias para Boolean. O valor da variável tipo Boolean deve ser true ou false iniciando com minúscula. Não é possível atribuir 0 ou 1 para o tipo Boolean de forma implícita.

Ex.:

```
String nome="João";  
string nome="João";
```

EXERCÍCIOS

1. Declarar variáveis em C# para armazenar:

- Idade Pessoa
- Número de alunos da Fatec Sorocaba
- Populacao de Sorocaba (aproxim. 695 mil)
- Populacao Brasil (aproxim. 215 milhões)
- Populacao Mundial (aproxim. 8 bilhões)
- OpcaoTurno (Manhã ou Tarde)
- É brasileiro
- Salário de um Funcionário
- Nome da Faculdade
- Data de Nascimento

EXERCÍCIOS

2. Considere as sentenças:

```
int n = 100;
```

```
string s = n;
```

Sobre a sentença b pode-se afirmar que:

- a) Foi utilizada uma conversão explícita.
- b) Foi utilizada uma conversão implícita permitida.
- c) Nesse caso não é possível fazer uma conversão implícita, vai apresentar erro.

EXERCÍCIOS

3. Considere as instruções apresentadas abaixo:

```
int x = -32768;  
ushort i = Convert.ToUInt15(x);
```

Podemos afirmar que:

- a) Ocorrerá um erro.
- b) I vale 32768.
- c) I vale -32768;
- d) NDA

EXERCÍCIOS

4. Assinale com **V** para as sentenças verdadeiras e **F** para as falsas:

- ☐ () Uma variável do tipo int ocupa 8 bytes.
- ☐ () Constantes são identificadores assim como as outras variáveis de memória são, mas não sofrem alterações.
- ☐ () Uma variável do tipo sbyte ocupa 1 byte com sinal.
- ☐ () Na nomenclatura das variáveis não deve ser usado o hífen (-).
- ☐ () Toda variável declarada deve ser obrigatoriamente iniciada por um caracter alfabético.
- ☐ () Uma variável do tipo double ocupa 16 bytes.
- ☐ () Na nomenclatura das variáveis locais não é recomendado o underline (_), embora seja possível.
- ☐ () Podem ser declaradas duas variáveis com o mesmo nome desde que em escopos diferentes.
- ☐ () Variáveis x e X são consideradas iguais.
- ☐ () É possível declarar duas variáveis x e y ao mesmo tempo de forma implícita (utilizando var). Ex. var x,y;
- ☐ () A variável _nome é válida.
- ☐ () A notação Camel Case é utilizada para variáveis e parâmetros dos métodos.
- ☐ () A notação Pascal Case pode ser utilizada para nome de classes.

EXERCÍCIOS

5. É importante conhecer os tipos de dados.
Imagine a seguinte situação:

a) `string strDado;`
`strDado = "65536";`

b) `int strDado;`
`StrDado = 65536;`

Em qual dos casos, economiza-se memória?