UNINOVE

# Tipos e declaração de variáveis

CONCEITUAR OS TIPOS DE DADOS PRIMITIVOS, IDENTIFICADORES, VARIÁVEIS E DECLARAÇÃO DE VARIÁVEIS EM C#. APRESENTAR OS MÉTODOS PARA CONVERSÃO DE TIPOS DE DADOS. APRESENTAR AS PROPRIEDADES LOCATION, POSITION, SIZE E READONLY DOS CONTROLES VISUAIS TEXTBOX, LABEL E BUTTON.

AUTOR(A): PROF. APARECIDA DE FATIMA CASTELLO ROSA

## Tipos de Dados Primitivos

Existem tipos de dados já definidos na plataforma .NET, conhecidos como *tipos de dados internos* ou *tipos de dados primitivos*.
O tipo de dados de uma variável informa a quantidade de memória, em bytes, que ela ocupará e o modo como um valor deverá ser armazenado e interpretado. Também define as operações que poderão ser realizadas com essas variáveis.
As informações armazenadas em um tipo podem incluir o seguinte:

- O espaço de armazenamento exigido por uma variável do tipo.
- O máximo e mínimo que podem ser representados.
- Os membros (métodos, campos, eventos e assim por diante) que ele contém.
- O tipo base do qual ele herda.
- A localização da memória para variáveis alocadas em tempo de execução.
- Os tipos de operações que são permitidas.

A linguagem C# tem uma série de tipos de dados primitivos conforme apresentado a seguir.

## Tipos de dados para números inteiros

| Tipo | Valores permitidos (intervalo) | Tamanho (bits) |
|---|---|---|
| sbyte | −128 e 127 | 8 |
| byte | 0 e 255 | 8 |
| short | −32768 e 32767 | 16 |
| ushort | 0 e 65535 | 16 |
| int | −2147483648 e 2147483647 | 32 |
| uint | 0 e 4294967295 | 32 |
| long | −9223372036854775808 e 9223372036854775807 | 64 |
| ulong | 0 e 18446744073709551615 | 64 |

O caractere *u* antes do nome de algumas variáveis acima é uma abreviação para sem sinal, o que significa que você não poderá armazenar valores negativos nas variáveis declaradas daquele tipo, conforme apresentado na tabela na coluna valores permitidos.

# Tipos de dados para números reais

| Tipo | Valores permitidos (intervalo) | Tamanho (bits) |
|---|---|---|
| float | $\pm 1.5 \times 10^{-45}$ até $\pm 3.4 \times 10^{38}$, 7 dígitos de precisão | 32 |
| double | $5.0 \times 10^{-324}$ até $1.7 \times 10^{308}$, 15 dígitos de precisão | 64 |
| decimal | $1.0 \times 10^{-28}$ até $7.9 \times 10^{28}$, 28 dígitos de precisão | 128 |

Para qualquer cálculo que envolva dinheiro ou finanças, o tipo decimal deve ser sempre utilizado. Só este tipo tem a precisão adequada para evitar os erros críticos de arredondamento.

# Outros tipos de Dados

Ainda existem outros três tipos de dados. Veja quais são:

| Tipo | Valores permitidos (intervalo) |
|---|---|
| char | um caractere Unicode no intervalo de 0 and 65535 |
| bool | Valor Booleano (lógico), true ou false |

UNINOVE

# Identificadores

Identificadores são nomes que você utiliza para identificar (nomear) os elementos no seu programa, tais como, *namespaces*, classes, métodos e variáveis.

Identificadores devem seguir estas regras:

- Somente letras (maiúsculas e minúsculas), dígitos (números), e o caractere underline "_" (sublinhado) são permitidos.
- Um identificador pode ter um ou mais caracteres.
- Um identificador deve começar com uma letra. O primeiro caractere não pode ser em hipótese alguma um número.
- Não pode ter espaços em branco.
- identificadores devem ser únicos dentro de um determinado escopo.
- Identificadores são case-sensitive, ou seja, diferenciam letras maiúsculas de minúsculas.
- Palavras reservadas da linguagem não podem ser utilizadas para identificadores, bem como o nome dos controles.
- Não utilizar acentuação e nem "ç".

Utilize as mesmas regras para nomear os controles da Linguagem C#.

Algumas convenções que devem ser seguidas para definir os identificadores:

- Identificadores devem ser legíveis.
- Não utilize abreviações.
- Identificadores devem transmitir o significado, ou seja, dê nomes significativos que expressam o que o identificador representa.

Como vimos, em C#, os identificadores são case-sensitive. As convenções de nomenclatura recomendadas sugerem a utilização dos seguintes padrões:

| Notação | Definição | Para nomear | Exemplos |
|---|---|---|---|
| camel casing | primeira letra de cada palavra em minúsculo | variáveis e parâmetros | nome, nomeCompleto, nomeDoLivro |
| Pascal casing | cada palavra inicia com letra maiúscula | métodos e outros identificadores | CalcularMedia( ) |

# Variáveis

O que é uma variável ?

Podemos dizer que uma variável é um local de armazenamento que contém um valor e que pode sofrer alteração durante a execução de um programa.

Uma variável é uma posição (local) da memória do computador em que um valor pode ser armazenado para ser utilizado por um programa. Você pode pensar em uma variável como sendo um espaço reservado em memória no computador que armazena valores temporariamente.
A linguagem C# é uma linguagem estaticamente tipada e fortemente tipada. Uma linguagem estaticamente tipada exige que os tipos das variáveis sejam definidos antes da compilação e a fortemente tipada exige que os valores armazenados na variável sejam compatíveis com o seu tipo. Na linguagem C# as variáveis devem ser declaradas antes de serem utilizadas.

# Declarando Variáveis

Para declarar variáveis, obrigatoriamente, você deverá definir um tipo de dados e um identificador (nome). A declaração de variáveis pode ser feita em qualquer lugar dentro de um bloco.
Lembre-se das regras descritas acima (no item Identificadores) para declarar variáveis. O nome da variável deve ser único dentro de um escopo no qual ela é utilizada. Você usa o nome da variável para acessa-la e para fazer referência ao valor que ela armazena.
Quando você declarar variáveis no seu programa procure dar nomes significativos para facilitar o entendimento posteriormente e mesmo para facilitar a leitura do seu código por outras pessoas.
Na declaração de variáveis devemos primeiramente informar o seu tipo de dados e a seguir o nome da variável ou das variáveis daquele mesmo tipo, finalizando o comando com ponto-e-vírgula ( ; ). Veja os exemplos a seguir.

```
1. int idade;
2. double nota1, nota2, nota3, media;
```

No primeiro exemplo (linha 1) acima, foi declarada uma variável do tipo inteiro (int) com o nome idade. No segundo (linha 2), declaramos quatro variáveis do tipo double: nota1, nota2, nota3, media.
Um valor pode ser atribuído a uma variável usando o sinal de igual que é o operador de atribuição ( = ). Quando atribuímos um valor à variável estamos inicializando-a.

```
1. idade = 42;
```

Podemos também declarar e inicializar a variável em uma única instrução, ou seja, atribuir um valor à variável na declaração. Veja os exemplos a seguir:

```
1. int idade = 42;
2. double nota1=8.5, nota2=7.0, nota3=7.5, media;
```

Note que para as variáveis do tipo double o separador de casas decimais é o ponto ( . ). Veja também que as variáveis nota1, nota2 e nota3 foram inicializadas e media não.
Depois que uma variável for declarada, ela não poderá ser redeclarada com um novo tipo. Também não pode ser atribuído a uma variável um valor que não é compatível com seu tipo declarado. Tomando como exemplo a variável idade que foi declarada com o tipo int, não será permitido fazer a seguinte atribuição: idade = true;.

# Conversão de tipos

As variáveis possuem diferentes tipos. O tipo determina o tipo de dados que uma variável pode conter, por exemplo, uma variável int pode conter somente dados numéricos inteiros, sem casas decimais; uma variável string pode conter somente texto. O que acontece quando você desejar exibir um int em um controle MessageBox ou em um Label ou até em um TextBox que requerem texto, ou quando você digitar um número em um controle TextBox, que armazena texto, para ser atribuído a uma variável do tipo numérica?
Essa é uma situação comum quando estamos desenvolvendo uma aplicação. Você pode ter os dados de um formulário (Form) que serão do tipo string no qual o usuário digita as informações em controles, como por exemplo, TextBox que armazena texto na sua propriedade Text, e essas informações serem atribuídas a variáveis de outro tipo, como por exemplo, variáveis numéricas. Ou você pode ter um Form que será preenchido com os dados resultantes de uma consulta em um Banco de Dados que pode ter valores que não sejam texto.
Quando isso ocorrer, você precisa realizar a conversão entre tipos de dados.

# Conversão entre tipos numéricos

Essa conversão pode ocorrer de forma explicita (também conhecida como Cast) ou implícita.

- Forma implícita: quando atribuímos um valor de um tipo 'menor' para um tipo 'maior'. Quando um tipo é convertido para outros tipos sem perda de dados. Por exemplo:
    - de short para int
- Conversão explícita: quando atribuímos um valor de tipo 'maior' para outra de tipo 'menor'. Ocorre quando dados podem ser perdidos.
    - de int para short
        - o tipo int tem 32 bits
        - o tipo short tem 16 bits. Nesse caso serão descartados os valores acima de 16 bits.

Para evitar que isso aconteça devemos forçar o C# a realizar uma conversão entre os tipos, uma conversão explícita, também chamada de coersão (cast).
Veja o exemplo a seguir de conversão de tipos de dados numéricos.

```
1. int a = 123;
2. long b = a;           //conversão implícita de um int para long
3. int c = (int) b;      //conversão explicita de long para int
```

# Conversão implícita de tipos numéricos

A tabela a seguir mostra as conversões numéricas implícitas.

| De | Para |
|---|---|
| sbyte (https://msdn.microsoft.com/pt-br/library/d86he86x.aspx) | short , int, long, float, double, ou decimal |
| Byte (https://msdn.microsoft.com/pt-br/library/5bdb6693.aspx) | short , ushort, int, uint, long, ulong, float, double, ou decimal |
| short (https://msdn.microsoft.com/pt-br/library/ybs77ex4.aspx) | int , long, float, double, ou decimal |
| ushort (https://msdn.microsoft.com/pt-br/library/cbf1574z.aspx) | int , uint, long, ulong, float, double, ou decimal |

Observações:

Conversões de int, uint, long, ou ulong para float e de long ou ulong para double podem causar perda de dados (perda de precisão).

# Conversão explícita de tipos numéricos

A tabela a seguir mostra as conversões numéricas explícitas.

| De | Para |
|---|---|
| sbyte (https://msdn.microsoft.com/pt-br/library/d86he86x.aspx) | byte , ushort, uint, ulong, ou char |
| Byte (https://msdn.microsoft.com/pt-br/library/5bdb6693.aspx) | Sbyte ou char |
| short (https://msdn.microsoft.com/pt-br/library/ybs77ex4.aspx) | sbyte , byte, ushort, uint, ulong, ou char |
| ushort (https://msdn.microsoft.com/pt-br/library/cbf1574z.aspx) | sbyte , byte, short, ou char |
| int (https://msdn.microsoft.com/pt-br/library/5kzh1b5w.aspx) | sbyte , byte, short, ushort, uint, ulong, ou char |
| uint (https://msdn.microsoft.com/pt-br/library/x0sksh43.aspx) | sbyte , byte, short, ushort, int, ou char |
| Long (https://msdn.microsoft.com/pt-br/library/ctetwysk.aspx) | sbyte , byte, short, ushort, int, uint, ulong, ou char |
| ulong (https://msdn.microsoft.com/pt-br/library/t98873t4.aspx) | sbyte , byte, short, ushort, int, uint, long, ou char |

```
int num;

num = Convert.ToInt32(txtNumero.Text);

//ou

num = Int32.Parse(txtNumero.Text);
```

Agora vamos fazer ao contrário, você tem uma variável numérica do tipo int e quer que esse valor seja apresentado no controle TextBox.

```
txtNumero.Text = Convert.ToString(num);

//ou

txtNumero.Text = num.ToString();
```

A tabela a seguir lista alguns dos métodos da classe Convert que você pode utilizar.

| Tipo Numérico | Método |
|---|---|
| decimal | ToDecimal(String) |
| float | ToSingle(String) |
| double | ToDouble(String) |
| short | ToInt16(String) |
| int | ToInt32(String) |
| long | ToInt64(String) |
| ushort | ToUInt16(String) |
| uint | ToUInt32(String) |
| ulong | ToUInt64(String) |

Fonte: https://msdn.microsoft.com/pt-br/library/bb397679.aspx (https://msdn.microsoft.com/pt-br/library/bb397679.aspx)

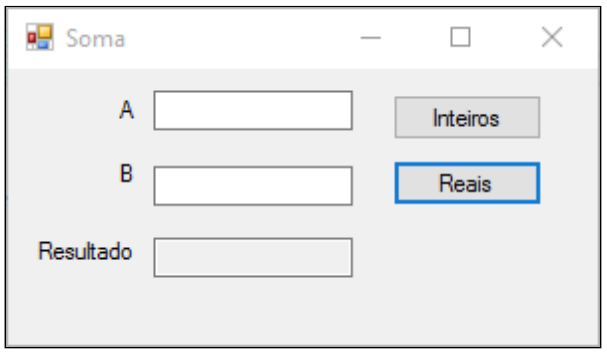
Por exemplo, a chamada ao método Convert.ToInt32(String) converte uma entrada do tipo string para um

UNINOVE

# Colocando em Prática

Vamos criar um projeto para declarar variáveis, fazer a soma das variáveis e apresentar o resultado da operação.

Veja a interface gráfica desse exemplo.




Observação: A aparência pode ser diferente no seu computador dependendo da versão do seu sistema operacional.

O form contém os seguintes controles:

a. 3 Label;
b. 3 TextBox;
c. 2 Button.

Vamos começar.

1. Inicie o Visual Studio (se ainda não o fez).
2. Na Start Page escolha New Project, ou caso a Start Page não esteja visível, na barra de menu escolha File, New Project.

A caixa de diálogo é exibida.

1. Selecione no painel à esquerda o template Visual C#, no painel central escolha Windows Forms Application.
2. Na caixa de texto Name: Variaveis.
3. Location: escolha o local onde deseja salvar o seu projeto.
4. Mantenha selecionado: Create diretory for solution.
5. Clique em OK

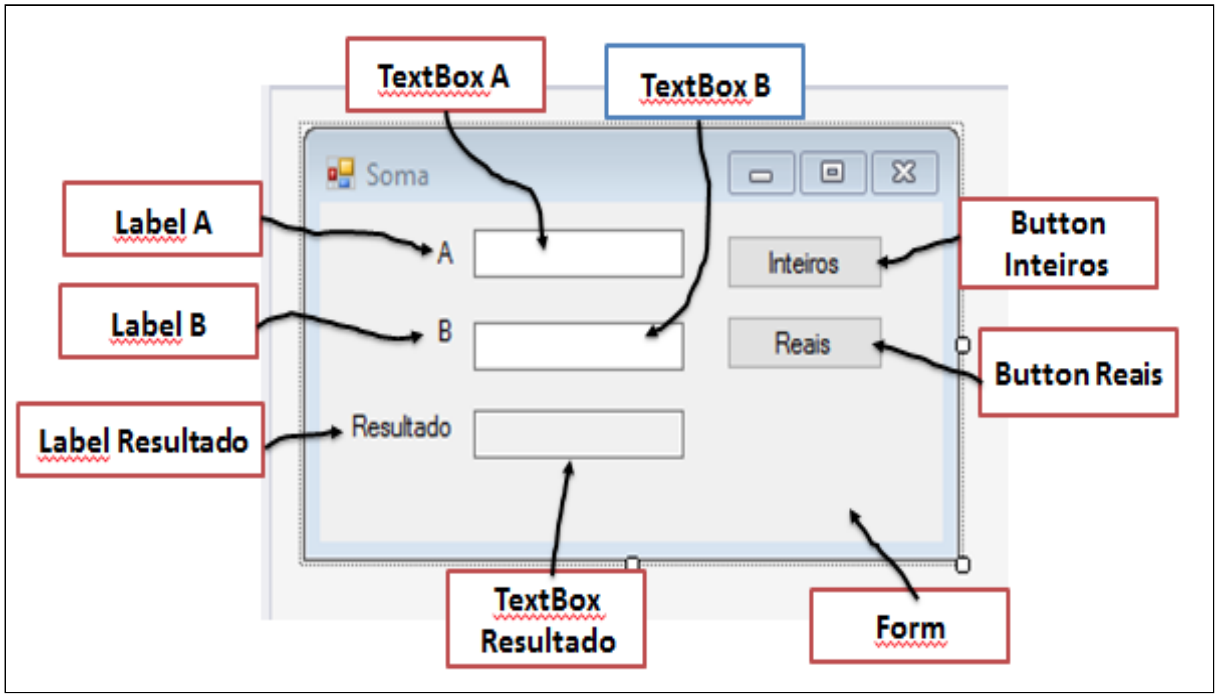
Novas Propriedades

- Location: define o posicionamento do elemento no formulário (form) considerando o eixo x e y
- Size: define o comprimento (Width) e a largura (Height) do controle.
- ReadOnly: Essa propriedade define se o controle TextBox pode ser editado ou é somente leitura, ou seja, quando o valor for igual a true o usuário não consegue inserir dados no TextBox

Criando o Design do Form

1. Após fixar (Auto Hide) a Toolbox, abra o nó Common Controls, e insira os controles no Form. Lembre-se para inserir os controles, basta selecionar o controle na ToolBox, clicar, segurar e arrastar para a posição que desejar dentro do Form.

Você poderá ter um layout parecido com a figura a seguir, mas não se preocupe se o posicionamento e tamanho dos controles não estiverem iguais ao da figura porque hoje você vai trabalhar com novas propriedades que alteram o layout.




Selecione os controles e altere as propriedades conforme definido na tabela abaixo:

| Controle | Propriedade = Valor |
|---|---|
| Form | (name) = frmSoma<br>Text = Soma<br>Size = 312;178 |
| Label A | (name) = lblA<br>Text = A<br>Location = 53;14 |
| Label B | (name) = lblB<br>Text = B<br>Location = 53;46 |
| Label Resultado | (name) = lblResultado<br>Text = Resultado<br>Location = 12;85 |
| TextBox A | (name) = txtA<br>Location = |
| TextBox B | (name) = txtB<br>Location = |
| TextBox Resultado | (name) = txtResultado<br>Location =<br>ReadOnly = true |
| Button Inteiros | (name) = btnInteiros<br>Text = Inteiros<br>Location = |
| Button Reais | (name) = btnReais<br>Text = Reais<br>Location = |

Escrevendo o código do programa Variaveis

Agora que você já criou a interface e alterou as propriedades dos controles, vamos codificar o botão btnInteiros. Para abrir o editor de código dê duplo clique no botão btnInteiros e digite o código a seguir.

```
/*declaração das variáveis*/
            int a, b, soma;

            /* entrada de dados*/

            a = Int32.Parse(txtA.Text);
            b = Int32.Parse(txtB.Text);

            /*processamento*/
            soma = a + b;

            /*saída de dados*/
            txtResultado.Text = soma.ToString();
```

Agora vamos codificar o botão btnReais.

Volte para o modo design e dê duplo clique no botão btnReais e digite o código a seguir.

```
 /*declaração das variáveis*/
            double a, b, soma;

            /*entrada de dados*/
            a = Convert.ToDouble(txtA.Text);
            b = double.Parse(txtB.Text);


            /*processamento*/
            soma = a + b;

            /*saída de dados*/
            txtResultado.Text = soma.ToString();
```

Os códigos devem ter ficado parecidos com a imagem a seguir.

UNINOVE

```
1 reference
private void btnInteiros_Click(object sender, EventArgs e)
{

    /*declaração das variáveis*/
    int a, b, soma;

    /* entrada de dados*/
    a = Int32.Parse(txtA.Text);
    b = Int32.Parse(txtB.Text);

    /*processamento*/
    soma = a + b;

    /*saída de dados*/
    txtResultado.Text = soma.ToString();
}
1 reference
private void btnReais_Click(object sender, EventArgs e)
{
  /*declaração das variáveis*/
    double a, b, soma;

    /*entrada de dados*/
    a = Convert.ToDouble(txtA.Text);
    b = double.Parse(txtB.Text);

    /*processamento*/
    soma = a + b;

    /*saída de dados*/
    txtResultado.Text = soma.ToString();
}
```

Veja que não foi preciso digitar as linhas dos eventos dos botões: private void btnInteiros..... e private void btnReais.....

Quando você deu duplo clique em cada botão, o editor de código do Visual Studio criou esses códigos.
Agora pressione F5 para executar o seu programa e faça alguns testes.
Quando for digitar números reais, com casas decimais, a entrada depende de como está configurado o seu sistema Operacional Windows para trabalhar com números decimais. Digite utilizando vírgula como separador de decimais, como por exemplo, 29,35. Caso o resultado não seja o correto, experimente digitar 22.35. Faça os testes.

## ATIVIDADE

De acordo com as regras para nomeação e declaração de variáveis, selecione a alternativa correta que contém a declaração INVÁLIDA.

A. float  tempo de percurso;
B. int numeros;
C. decimal salario_mensal;
D. bool clienteAtivo;

## ATIVIDADE

Quando declaramos variáveis, devemos informar o seu tipo e o identificador (nome). O tipo de dados  char armazena

A. uma string.
B. números sem casas decimais.
C. valores true ou false.
D. um caractere.

UNINOVE

# ATIVIDADE

As variáveis possuem tipos de dados diferentes e frequentemente precisamos fazer conversões entre tipos de dados. Selecione a alternativa correta referente aos tipos de conversão de dados da linguagem C#.

A. Conversão explícita e textual.

B. Conversão implícita e inteira.

C. Conversão implícita e explícita.

D. Conversão estática e dinâmica.

# REFERÊNCIA

DEITEL, H. M. *et al. C# Como Programar*. São Paulo: Pearson Makron Books, 2003.

DORMAN, Scott. *Sams Teach Yourself Visual C#®2010 in 24 Hours: Complete Starter Kit*. 2010.

OLSSON, Mikael. *C# Quick Syntax Reference*. Berkeley, CA: Apress, 2013.

SHARP, John. *Microsoft Visual C# 2013 Step by Step*. Microsoft Press, 2013.

SITES

Microsoft. Disponível em: <https://msdn.microsoft.com/pt-br/library/ms173104.aspx (https://msdn.microsoft.com/pt-br/library/ms173104.aspx)>. Acesso em 17/09/2015

Microsoft. Disponível em: <https://msdn.microsoft.com/pt-br/library/hh147285(VS.88).aspx#Variables (https://msdn.microsoft.com/pt-br/library/hh147285(VS.88).aspx#Variables) Acesso em 17/09/2015

Macoratti. Disponível em: <http://www.macoratti.net/12/12/c_num1.htm (http://www.macoratti.net/12/12/c_num1.htm)>. Acesso em 17/09/2015

Microsoft. Disponível em: <https://msdn.microsoft.com/pt-br/library/bb397679.aspx (https://msdn.microsoft.com/pt-br/library/bb397679.aspx)>. Acesso em 17/09/2015

Microsoft. Disponível em: https://msdn.microsoft.com/pt-br/library/yht2cx7b.aspx (https://msdn.microsoft.com/pt-br/library/yht2cx7b.aspx).> Acesso em 17/09/2015

Microsoft. Disponível em: <https://msdn.microsoft.com/pt-br/library/y5b434w4.aspx (https://msdn.microsoft.com/pt-br/library/y5b434w4.aspx)>. Acesso em 17/09/2015