alura

Variáveis e tipos

O que são variáveis?

Variáveis são formas de guardarmos valores na memória do computador.

Como o computador só entende “números”, as variáveis são “apelidos” que nos ajudam a identificar em qual endereço de memória está determinado dado.

Variáveis

```
int quantidadeProdutos = 42;
```

Na imagem da memória, conseguimos ver a correspondência. O computador entende que o conteúdo ‘42’ está armazenado no endereço 120, enquanto nós conseguimos identificar esse conteúdo pelo nome ‘quantidadeProdutos’

Memória		
Endereço	**Variável**	**Conteúdo**
119		
120	int quantidadeProdutos	42
121		

Atribuição de variáveis

Para atribuir um valor a uma variável no C#, usamos o símbolo = e utilizamos o padrão de nomenclatura *camelCase*.

```
int quantidadeProdutos = 42;

tipo nomeVariavel = dado;
```

Declaração de variáveis

Ao utilizar a instrução anterior, estamos fazendo duas coisas: declarando e inicializando uma variável. Para declarar uma variável, basta apenas utilizar o tipo e o nome dela.

```
int quantidadeProdutos;
```

Declaração de variável

Declaração de variáveis

Já para inicializar uma variável, usamos o nome dela e a atribuição do valor

```
int quantidadeProdutos;

quantidadeProdutos = 42;
```

Declaração de variável

Inicialização de variável

Tipos de dados

Os tipos de dados podem ser agrupados de duas formas:

- **Primitivos:** são aqueles disponibilizados pela linguagem, que já têm uma palavra reservada. Em C#, os tipos primitivos representam valores booleanos (true ou false), caracteres individuais, valores numéricos inteiros e valores numéricos decimais.

- **Referenciados:** são tipos geralmente compostos por outros tipos. Serão melhor explorados nos cursos de Orientação a Objetos.

Tipos primitivos

Os principais tipos primitivos de C# estão relacionados nas tabelas a seguir:

Tipo	Valores possíveis	Exemplo
bool	true ou false	bool ativo = true;
char	caracteres unicode	char inicial = 'A';
byte	0 a 255	byte idade = 25;
sbyte	-128 a 127	sbyte temperatura = -5;
short	-32.768 a 32.767	short numeroCurto = 1000;
ushort	0 a 65.535	ushort codigoProduto = 500;

Tipos primitivos

Tipo	Valores possíveis	Exemplo
int	-2.147.483.648 a 2.147.483.647	int contador = 100;
uint	0 a 4.294.967.295	uint populacao = 50000;
long	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807	long distancia = 1000000000;
ulong	0 a 18.446.744.073.709.551.615	ulong estrelas = 9999999999;
float	$\pm 1{,}5 \times 10^{-45}$ a $\pm 3{,}4 \times 10^{38}$	float altura = 1.75f;
double	$\pm 5{,}0 \times 10^{-324}$ a $\pm 1{,}7 \times 10^{308}$	double peso = 72.5;
decimal	$\pm 1{,}0 \times 10^{-28}$ a $\pm 7{,}9 \times 10^{28}$	decimal preco = 199.99m;

Operadores aritméticos

Operador	Significado	Exemplo	Resultado
+	Adição	`5 + 3`	`8`
-	Subtração	`10 - 4`	`6`
*	Multiplicação	`6 * 2`	`12`
/	Divisão	`10 / 2`	`5`
%	Módulo (resto da divisão)	`10 % 3`	`1`
++	Incremento (+1)	`int x = 5; x++;`	`x = 6`
--	Decremento (-1)	`int y = 8; y--;`	`y = 7`

Ao trabalhar com tipos numéricos, podemos utilizar os operadores aritméticos, listados ao lado.

Conversão de tipos

Podemos converter tipos de duas formas:

- **Implícita:** quando um tipo menor é convertido para um tipo maior;

```
int quantidadeProdutos = 42;

long novaQuantidade = quantidadeProdutos;
```

Uma variável do tipo inteiro é menor que um do tipo long, por isso pôde ser atribuída diretamente.

Conversão de tipos

- **Explícita:** quando um tipo maior é convertido para um tipo menor;

```
double quantidadeProdutos = 42.0;

int novaQuantidade = (int) quantidadeProdutos;
```

Foi preciso indicar explicitamente que era para pegar o valor inteiro da variável quantidadeProdutos e atribuir à variável novaQuantidade.

Conversão de tipos

Devemos sempre nos atentar à conversão automática de tipos do compilador, principalmente em operações matemáticas:

```
double valorUnitario = 30/4;

// resultado da operação = 7.0
```

Como temos a divisão entre números “automaticamente” inteiros, o compilador entende que o resultado será um inteiro também. Ele realiza a divisão e obtém o inteiro, truncando o resultado. Só depois a variável é convertida de int para double.

Conversão de tipos

Para que não tenhamos problemas com o compilador, devemos declarar um dos valores da operação explicitamente como double:

```
double valorUnitario = 30.0/4;

// resultado da operação = 7.5
```

Agora, temos a divisão de um número “automaticamente” double e um inteiro. Assim, o compilador entende que o resultado será um **double**, não tendo problemas com truncamento. Também daria certo se fizéssemos **30/4.0**.

Tabela de conversão de tipos

Tipo	Tamanho(bits)	Conversão implícita	Conversão explícita
byte	8	short, ushort, int, uint, long, ulong, float, double, decimal	sbyte, char
sbyte	8	short, int, long, float, double, decimal	byte, ushort, uint, ulong, char
short	16	int, long, float, double, decimal	byte, sbyte, char
ushort	16	int, uint, long, ulong, float, double, decimal	byte, sbyte, short, char
char	16	ushort, int, uint, long, ulong, float, double, decimal	byte, sbyte, short
int	32	long, float, double, decimal	byte, sbyte, short, ushort, char

Tabela de conversão de tipos

Tipo	Tamanho(bits)	Conversão implícita	Conversão explícita
uint	32	long, ulong, float, double, decimal	byte, sbyte, short, ushort, char, int
long	64	float, double, decimal	byte, sbyte, short, ushort, char, int, uint
ulong	64	float, double, decimal	byte, sbyte, short, ushort, char, int, uint, long
float	32	double	byte, sbyte, short, ushort, char, int, uint, long, ulong, decimal
double	64	Nenhuma	byte, sbyte, short, ushort, char, int, uint, long, ulong, float, decimal
decimal	128	Nenhuma (é o maior)	byte, sbyte, short, ushort, char, int, uint, long, ulong, float, double

Inferência de tipos

Quando declaramos uma variável em C#, além de usar os tipos de dados, podemos também usar a palavra reservada **var**. Ao utilizá-la, estamos indicando ao compilador que ele deve **inferir** o tipo da variável.

```
int quantidadeProdutos = 42;
var quantidadeProdutos = 42;
```

Ao atribuir o valor 42 à variável, o compilador irá entender que ela representa um valor inteiro e atribuir o tipo **int** a ela. Isso porque, para valores inteiros, o padrão usado é esse tipo. Já ao identificar valores decimais, a variável será automaticamente do tipo **double**.

Inferência de tipos

Variáveis do tipo var devem ser sempre inicializadas, para que a inferência de tipos aconteça!

```
var quantidadeProdutos;
```

Esse código não compila! O compilador precisa de um tipo para armazenar a variável na memória. Se não atribuirmos um valor ao var, esse tipo não será identificado.

Compartilhe um resumo de seus novos conhecimentos em suas redes sociais.
#aprendizadoalura

alura | Escola Programação