Curso de C#

Variáveis, Tipos e Expressões



Roteiro

- Variáveis e memória
- Rigidez de tipos
- Tipos simples
- Tipos inteiros em C#
- Tipo booleano
- Declaração de variável
- Comando de atribuição
- Expressões
- Operadores



Variáveis e memória

- Uma variável é uma área de memória, associada a um nome, que pode armazenar valores de um determinado tipo.
- Um tipo de dado define um conjunto de valores e um conjunto de operações.
- Uma variável de um certo tipo T pode conter, num dado instante, um valor pertencente ao tipo T.



Rigidez de tipos

- C# é uma linguagem com rigidez de tipos (em inglês strongly typed).
 - um valor pertencente a um determinado tipo só pode ser usado como argumento em operações que prevêem operações desse tipo.



Tipos simples

- Os tipos simples ou tipos primitivos em C# são:
 - o tipo booleano
 - tipos numéricos
 - inteiros
 - ponto flutuante
 - decimais



Tipos inteiros em C#

- C# oferece diversos tipos de inteiros.
 Cada um deles é definido por:
 - uma gama de valores
 - uma representação interna
- O programador tem a opção de decidir qual o tipo inteiro a ser usado em função da necessidade de sua aplicação.

Tipos inteiros em C#

Tipo	Tamanho (bits)	Intervalo de Valores
sbyte	8	-128 a 127 (-(2 ⁷) a 2 ⁷ -1)
byte	8	0 a 255 (0 a 2 ⁸ -1)
short	16	-32768 a 32767 (-(2 ¹⁵) a 2 ¹⁵ -1)
ushort	16	0 a 65535 (0 a 2 ¹⁶ -1)
int	32	-(2 ³¹) a 2 ³¹ -1
uint	32	0 a 2 ³² -1
long	64	-(2 ⁶³) a 2 ⁶³ -1
ulong	64	0 a 2 ⁶⁴ -1
char	16	0 a 65535 (0 a 2 ¹⁶ -1)



O tipo booleano

- O tipo booleano, em C# chamado de bool, é usado em geral para operações lógicas.
- Os valores possíveis para uma variável desse tipo são true e false.



Declaração de variável

 Em C# a declaração de uma variável tem a seguinte estrutura:

```
tipo nome;
OU
tipo nome = valor inicial;
• Exemplos
string n;
int x = 10; bool b = false;
```



- O comando de atribuição é usado para alterar o valor de uma variável.
- Sua forma:

```
variável = expressão;
```

• Exemplos:

```
x = z + 10;
z = z + 1;
b = true;
```

Uma declaração da forma

int
$$x = 10$$
;

é na verdade a combinação de uma declaração de variável com uma atribuição.



Expressões

- Uma expressão é uma combinação de operandos e operadores.
- Expressões em C# são semelhantes às expressões usadas em outras linguagens como C ou Pascal.



Expressões

- No caso de operações encadeadas, como em a+b*c o cálculo da expressão é feito de acordo com a precedência entre os operadores.
- Parênteses podem ser utilizados para alterar a ordem de cálculo das operações.
- Exemplos:

e



Expressões

- No caso de encadeamento de operações com a mesma precedência, a linguagem define o modo de associatividade (à esquerda ou à direita). Os operadores aritméticos em C# têm associatividade à esquerda.
- Exemplo:

```
6/2/3 é calculado como (6/2)/3 pois neste caso a associatividade é à esquerda. Em C#, apenas os operadores de atribuição (= *= /= %= += -= <<= >>= &= ^= |=) e o operador ternário (? :), têm associatividade à direita.
```



 Principais operadores, em ordem decrescente de precedência

Categoria	Operadores	Associatividade
Unário	+ - !	esquerda
Multiplicativo	* / %	esquerda
Aditivo	+ -	esquerda
Relacional	< = > >= <= == is	esquerda
Igualdade	== !=	esquerda
'and' (bool)	& &	esquerda
'ou' (bool)		esquerda



```
/* Operadores lógicos */
using System;
class Opr
  public static void Main(string[] args) {
      bool t = true;
      bool f = false;
      Console.WriteLine(
             "t||f: {0} t&&f: {1} !t: {2} !f: {3}",
             t||f,t&&f,!t,!f
             );
```

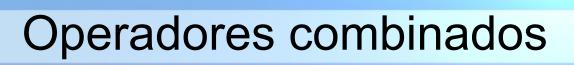


```
/* Operadores relacionais */
using System;
class Opr2 {
 public static void Main(string[] args) {
      bool gt,lt,eq;
      int a = 100; int b = 333;
      gt = a > b;
      lt = a < b;
      eq = a == b;
      Console.WriteLine("a>b:{0}, a<b:{1}, a==b:{2}",
                         gt, lt, eq
               );
```



Outro exemplo

```
/* Operadores aritméticos */
using System;
class Opr3 {
 public static void Main(string[] args) {
      int a = 127, b = 16, c = 4;
      int d = a/b/c; /* (a/b)/c */
      int m = a+b*c; /* a+(b*c) */
      Console . WriteLine (
               "a/b/c:{0}, a+b*c:{1}",
               d,m
              );
```



- O operador de atribuição pode ser combinado com outros operadores aritméticos.
- a op= exp; é equivalente a a = a op (exp);
- Exemplos:

```
a += b; equivalente a a = a + b;
a -= 2; equivalente a a = a - 2;
a *= 1+1; equivalente a a = a * (1+1);
a %= b*c+d; equivalente a a = a % (b*c+d);
```



Auto-incremento e autodecremento

- Numa expressão, os operadores ++ e -- podem ser usados antes ou depois da variável:
 - se usado após a variável, o incremento será feito depois de usar o valor da variável na expressão
 - se usado antes da variável, o incremento será feito antes do uso do valor na expressão
- Exemplos:

```
a = a/i++; equivalente a = a/i; i = i+1;

b = --k*2; equivalente a = k+1; b = k*2;
```