Programação Estruturada e Orientada a Objetos

REVISÃO

2013

O que veremos hoje?

- Introdução
- Revisão
- Exercícios

Transparências baseadas no material do Prof. Gilbert Azevedo

Variáveis

- Local de armazenamento que guarda um valor e é acessado por um identificador
- Nomenclatura recomendada
  - Não utilizar sublinhados
  - Não distinguir identificadores apenas por letras maiúsculas e minúsculas
  - Começar com letra minúscula
  - Se possuir mais de uma palavra, iniciar as palavras seguintes em maiúsculo (notação camelCase)
- Declaração de Variáveis
  - Todas as variáveis (locais) devem ser explicitamente declaradas e iniciadas antes de serem usadas no C#.

Tipos Predefinidos

Tipo predefinido	Classe .Net	Descrição	Exemplo
sbyte	System.SByte	Valor inteiro 8 bits com sinal	sbyte a = -1;
byte	System.Byte	Valor inteiro 8 bits sem sinal	byte b = 2;
short	System.Int16	Valor inteiro 16 bits com sinal	short c = -3;
ushort	System.UInt16	Valor inteiro 16 bits sem sinal	ushort d = 4;
int	System.Int32	Valor inteiro 32 bits com sinal	int e = -5;
uint	System.UInt32	Valor inteiro 32 bits sem sinal	unit f = 6;
long	System.Int64	Valor inteiro 64 bits com sinal	long g = -7;
ulong	System.UInt64	Valor inteiro 64 bits sem sinal	ulong h = 8;
char	System.Char	Carácter Unicode 16 bits	char i = 'A';
float	System.Single	Valor real IEEE 32 bits	float j = 0.42F;
double	System.Double	Valor real IEEE 64 bits	double k = 0.42;
bool	System.Boolean	1 byte (true ou false)	bool l = true;
decimal	System.Decimal	Valor monetário 128 bits	decimal m = 0.52M;
object	System.Object	Base de todos os tipos	object m = 10;
string	System.String	Seqüência de caracteres Unicode	string o = "C#";

Caracteres Especiais em Strings

- Alguns caracteres precisam de notação especial para serem incluídos em strings
  - \' – Aspas simples
  - \" – Aspas duplas
  - \\ – Contra-barra
  - \n – Nova linha
  - \r – Retorno ao início da linha
  - \t – Tabulação
- Inicialização de Strings
  - s = "Hello, world"; → Hello, world
  - s = "\"Hello, world\""; → "Hello, world"
  - s = @"'Hello, world'"; → 'Hello, world'

Conversão entre Tipos

- Conversões entre tipos são freqüentemente necessárias no desenvolvimento de aplicações
- Conversão implícita
  - Realizada pelo *runtime* em operações sem perda de informação
  - Ocorre quando um tipo é convertido em outro mais "completo"
- Conversão explícita
  - Deve ser realizada em operações que podem gerar perda de informação
  - Ocorre quanto um tipo é convertido em outro mais simples

Conversões Implícitas

Do tipo	Para o tipo
sbyte	short, int, long, float, double, decimal
byte	short, ushort, int, uint, long, ulong, float, double, decimal
short	int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long, ulong	float, double, decimal
float	double
char	ushort, int, uint, long, ulong, float, double, decimal
todos	object

Conversões Explícitas

- tipo Variavel1 = (cast-type) Variavel2
- Conversão sem perda
  - int x = 500;
  - short z = (short) x;
  - A variável z suporta o valor de x sem perder nenhuma informação, logo z recebe 500.
- Conversão com perda
  - double d = 1234.56;
  - int x = (int) d;
  - A variável x não recebe a parte decimal de d, logo x recebe apenas 1234, ou seja, o valor é truncado

Constantes

- Variáveis cujo valor permanece constante durante a execução do programa
- Utilizadas para deixar o código mais legível, fácil de manter e robusto
- Devem ser declaradas com *const* e receber um valor inicial
  - const double pi = 3.14159;
  - const int raioTerra = 6378;

Operadores

- Símbolos que representam operações que podem ser realizadas com constantes e variáveis
- Operadores Aritméticos
  - Realizam as operações aritméticas básicas
- Operadores Relacionais
  - Realizam comparações entre valores constantes ou variáveis
- Operadores Lógicos
  - Implementam as operações lógicas básicas

Operadores Aritméticos

Operação	C#	Tipos
Soma	+	I,R,C,S
Subtração	–	I,R,C
Multiplicação	*	I,R,C
Divisão	/	I,R,C
Divisão inteira	/	I
Resto	%	I, R, C
Incremento	++	I,R,C
Decremento	––	I,R,C

Exemplos de Operações

- 2+3 = 5
- 2.3+3.4 = 5.7
- '2' + '3' = 101
- 3.0+2 = 5.0
- 5.2-1.2 = 4.0
- 3*4 = 12
- 4/3 = 1
- 7%2 = 1
- 7.0/2 = 3.5
- 8++ = 9
- 8-- = 7

Prioridade nas Operações

- As expressões são avaliadas da esquerda para a direita
- Precedência
  - *, / , %
  - +, –
- Exemplos
  - 1 + 2 * 3 = 7
  - (1 + 2) * 3 = 9
- O uso de parêntesis torna o código mais legível e não diminui a performance da aplicação. O compilador remove os parêntesis extras.

Conversão de Strings em Números

- O método Parse das classes numéricas pode ser utilizado para converter uma string em um valor
  - string s = "123";
  - int x = int.Parse(s);
  - O método Parse gera exceções caso a string não represente um valor válido
- O método TryParse faz uma conversão testada, retornando um booleano caso a conversão seja bem sucedida.

Conversão de Números em Strings

- O método ToString pode ser utilizado para converter valores numéricos em strings
  - int x = 123;
  - string s = x.ToString();
- O método ToString pode ser invocado por constantes
  - string r = 123.ToString();

Classe System.Convert

- A classe System.Convert disponibiliza métodos para conversão entre os diversos tipos de variáveis
  - int x = 123;
  - string s = Convert.ToString(x);
  - int z = Convert.ToInt32(s);

Exercícios

- 1. Calcular a média parcial de uma disciplina, dadas as notas dos 1º e 2º bimestres (pesos 2 e 3).
- 2. Calcular área, perímetro e diagonal de um retângulo, dados base e altura.
- 3. Calcular a quantia gasta por um fumante, dados o Nº de anos que ele fuma, o Nº de cigarros fumados por dia e o preço de uma carteira de cigarros. Admitir que a carteira possui 20 cigarros.
- 4. Calcular o número mínimo de cédulas para obter um valor monetário dado em reais (sem centavos).

Dúvidas