IFRN

PROGRAMAÇÃO BÁSICA EM C#

Introdução ao C#

Prof. Gilbert Azevedo

Objetivos

- Rever os elementos usados para escrever um algoritmo
- Conhecer os tipos básicos de dados no C#
- Utilizar variáveis, operadores e comandos básicos de atribuição, entrada e saída de dados
- Utilizar os comandos de conversão entre textos e números
- Escrever um programa simples em C#

Elementos de um Algoritmo

- Constantes e Variáveis
  - Valores fixos (constantes) e mutáveis (variáveis) em um algoritmo
- Operadores
  - Símbolos que representam operações aritméticas, relacionais e lógicas
- Instruções (Funções e Métodos)
  - Comandos que são executados no programa: ler, escrever, calcular raiz quadrada, converter um texto em número, ...
- Estruturas de Controle
  - Estruturas condicionais: Controlam a execução condicional de comandos
  - Estruturas de repetição: Controlam a repetição de comandos

Tipos Básicos de Dados

- Inteiro (*int*): Representa um valor do conjunto dos números inteiros
  - 1, -10, 0, 50
- Real (*double*): Representa um valor do conjunto dos números reais
  - 1.0, -1.5, -10.0, 3.1415, 1.5e2, 1e-3
- Caractere (*char*): Representa um símbolo do teclado
  - '1' , 'A', 'a', '#', ' '
- Texto (*string*): Representa um conjunto de caracteres
  - "1", "Linguagem C#", "TADS", "", "Informática"
- Lógico (*bool*): Representa um valor verdadeiro ou falso
  - true, false

Variáveis

- Espaços reservados na memória para armazenar informações
  - Todas as variáveis de um programa devem ser declaradas
  - Informar um nome e um tipo de dado
  - Identificadores em minúsculo ou *lowerCamelCase*

```
class MainClass {
  public static void Main (string[] args) {
    int x = 120;
    double minhaVariavel = 1.5;
  }
}
```

Constantes

- Valores imutáveis em um programa
  - Semelhante à declaração de variáveis
  - Utiliza *const* para informar que o valor não pode ser alterado
  - Facilitar a escrita e manutenção do programa

```
class MainClass {
  public static void Main (string[] args) {
    const double pi = 3.14159;
    const int raioTerra = 6378;
  }
}
```

Tipos Básicos Predefinidos

Tipo predefinido	struct/class .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**	**Sequência de caracteres Unicode**	**string o = "C#";**

Exemplo de Tipos Inteiros

- Tipo *byte*: valor inteiro de 8 bits sem sinal
  - Binário: 0000.0000 a 1111.1111
  - Decimal: 0 a 255
  - `byte x = 9;  // 0000.1001`

- Tipo *int*: valor inteiro de 32 bits com sinal
  - Binário: 0000.0000.0000.0000.0000.0000.0000.0000 a 1111.1111.1111.1111.1111.1111.1111.1111
  - Decimal: -2.147.483.648 a 2.147.483.647
  - `int x = 9;  // 0000.0000.0000.0000.0000.0000.0000.1001`

Exemplo de Strings

- Todas as strings devem vir entre aspas duplas "
  - \" – aspas duplas
  - \\ – contra barra
  - \n – nova linha

- Exemplos
  - `string s1 = "Hello World";  // Hello World`
  - `string s2 = "\"Hello World\"";  // "Hello World"`
  - `string s3 = @"\'Hello World'";  // \'Hello World'`

Operadores

- Símbolos usados na realização de algumas operações
- 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,C
Resto	%	I,R,C

Exemplos de Operações

Operação	Resultado	Tipo
`2 + 3`	`5`	`int`
`'2' + '3'`	`101`	`int`
`"2" + "3"`	`"23"`	`string`
`2.3 + 3.4`	`5.7`	`double`
`3.0 + 2`	`5`	`double`
`5.2 - 1.2`	`4`	`double`
`3 * 4`	`12`	`int`
`4 / 3`	`1`	`int`
`7 % 2`	`1`	`int`
`7.0 / 2`	`3.5`	`double`
`7.5 % 2`	`1.5`	`double`
`7.5 % 2.4`	`0.3`	`double`

Prioridade nas Operações

- Expressões em geral são avaliadas da esquerda para a direita
- Precedência dos Operadores
  - Operadores Multiplicativos: *, / , %
  - Operadores Aditivos: +, –
- Exemplos
  - `1 + 2 * 3 = 7`
  - `(1 + 2) * 3 = 9`
- Na dúvida, use parênteses: não diminui a performance da aplicação

Atribuição, Entrada e Saída

- Operador de Atribuição: =
- Entrada: Console.ReadLine() – sempre retorna uma string!
- Saída: Console.WriteLine()
  - `int x, z;`
  - `x = 5;`
  - `int y = 10;`
  - `string s = Console.ReadLine();`
  - `z = int.Parse(Console.ReadLine()); // Parse converte em int`
  - `Console.WriteLine(x);  // mostra 5`
  - `Console.WriteLine("x");  // mostra x`

Conversão de Texto em Número

- O método *Parse* dos tipos numéricos converte um texto em número, se possível. *Parse* é chamado com o tipo.
  - `int x = int.Parse("123");`
  - `double z = double.Parse("123.45");`
  - `int y = int.Parse(Console.ReadLine());`
  - `double w = double.Parse(Console.ReadLine());`

- Se o valor informado não for um número, *Parse* lança uma exceção

Conversão de Número em Texto

- O método *ToString* dos tipos numéricos converte um número em texto. *ToString* é chamado com a variável.
  - `int x = 123;`
  - `double y = 123.4;`
  - `string r = x.ToString();`
  - `string s = y.ToString("0.00");`
  - `Console.WriteLine(r);  // 123`
  - `Console.WriteLine(s);  // 123.40`

Comentários

- São informações acrescentadas a um programa com o objetivo de identificá-lo ou de esclarecer alguns de seus trechos.

- Comentário de linha: //
  - // Este é um comentário de linha

- Comentário de bloco: /* ... */
  - /* Este
  - é um comentário
  - de bloco */

Exemplo: Área do Triângulo

- Algoritmo para calcular a área de um triângulo, dados sua base e sua altura
  - Declaração de Variáveis
    - b, h, a : real;
  - Início
    - Escreva("Digite a base do triângulo");
    - Leia(b);
    - Escreva("Digite a altura do triângulo");
    - Leia(h);
    - a ← b * h / 2;
    - Escreva("Area = ", a);
  - Fim.

Área do Triângulo em C#

```
class MainClass {
  public static void Main (string[] args) {
    double b, h, a;
    Console.WriteLine("Digite a base do triângulo");
    b = double.Parse(Console.ReadLine());
    Console.WriteLine("Digite a altura do triângulo");
    h = double.Parse(Console.ReadLine());
    a = b * h / 2;
    Console.WriteLine($"Área = {a:0.00}");
  }
}
```

Referências

- Microsoft Visual C# 2010 – Passo a passo, John Sharp, Bookman, 2010
- Tipos (Guia de Programação em C#)
  - https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/types/
- Operadores e Expressões
  - https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/statements-expressions-operators/operators
- Classe Console
  - https://docs.microsoft.com/pt-br/dotnet/api/system.console
- Formatação de Números
  - https://docs.microsoft.com/pt-br/dotnet/standard/base-types/custom-numeric-format-strings