Cap. 2, 3 e 4 - Introdução à linguagem C#

Aprenda a Programar com C# - 2ª Edição (2020)

Edições Sílabo

https://bit.ly/36nyKFm

António Trigo, Jorge Henriques
{antonio.trigo,jmvhenriques}@gmail.com

1 de outubro de 2020

Estrutura básica de um programa em C#

```
// Inclusao das livrarias necessarias
using System;

namespace ConsoleApplication17
{
class Program
{
        //Cabecalho ou prototipo da funcao principal
        static void Main(string[] args)
        {
        //Corpo da funcao principal
         //as declaracoes de uma variavel
        double x;
        // instrucao
        x = 2.50;
        // declaracoes e instrucoes terminadas pelo simbolo ';'
        Console.WriteLine("{0, 4:F2}",x);
        }
}
}
```

Notas

- As palavras chave, como if, float, int, etc., e os símbolos da linguagem C#, como =, *, ;, etc., são constituintes indivisíveis que não podem ser utilizados para outros fins do que aqueles para os quais foram definidos.
- A linguagem C# é "case sensitive", ou seja, faz distinção entre minúsculas e maiúsculas.

Variáveis

- Dado cujo valor é alterado durante a execução do programa;
- Uma variável é um nome dado a um espaço da memória principal que armazena um conteúdo;
- O tamanho do espaço ocupado pelas variáveis varia de acordo com o tipo (p.ex.: char ocupa 2 byte, int ocupa 4 bytes, etc.);
- Em C# a declaração das variáveis pode ocorrer em qualquer parte do programa.

Identificadores (para variáveis e não só)

- Na definição dos identificadores (nomes) das variáveis devem ter em conta que:
  - Apenas podem ser usadas as letras minúsculas e maiúsculas do alfabeto anglo-saxónico, algarismos decimais e o sinal "_" e "@";
  - Não podem começar por um algarismo;
  - Não é permitido usar palavras-chave da linguagem (p.ex.: if, int, etc.);
  - Exemplos de variáveis corretas: myBigVar, VAR1 e _test;
  - Exemplos de variáveis incorretas: 15Aula, for, primeiro-nome.

Operadores matemáticos

- +, adição
- -, subtração
- *, multiplicação
- /, divisão (aplica-se à divisão por inteiros e por números reais)
- %, resto da divisão

Atribuição

- A instrução de atribuição oferece o mecanismo por meio do qual é possível modificar dinamicamente as vinculações de valores a variáveis.
- Desta forma, uma instrução de atribuição permite iniciar o valor de uma variável ou substituir o valor antigo de uma variável por um novo;
- Na linguagem C#, a sintaxe de uma instrução de atribuição é: nomevariavel = expressao;
- O símbolo "=" designa a instrução de atribuição, em que o primeiro membro (esquerdo) é uma variável e o segundo membro (direito) é uma expressão (uma constante, uma variável ou uma sequência válida de constantes e variáveis ligadas por operadores).

Operadores de atribuição

- =, var1 = 5
- +=, var1 += 3
- -=, var1 -= 2
- *=, var1 *= 6
- /=, var1 /= 7
- %=, var1 %= 4

Operadores prefixos e posfixos

- ++, var1 = ++var2 - , var1 fica com o valor de var2 + 1
- - -, var1 = –var2, var1 fica com o valor de var2 - 1
- ++, var1 = var2++, var1 fica com o valor de var2 e var2 é incrementado de 1
- - -, var1 = var2–, var1 fica com o valor de var2 e var2 é decrementado de 1

Expressões lógicas

- São aquelas em que o resultado é verdadeiro (1) ou falso (0);
- Operadores lógicos:
  ! (negação); && (conjunção); ||(disjunção);
- Operadores relacionais:
  ==, <>ou !=, <, <=, >e >=.

Precedências dos operadores

- Quando uma expressão contém múltiplos operadores torna-se necessário conhecer a precedência/ordem dos mesmos para conseguir avaliar a expressão corretamente.
- Exemplo, $x + y * z$ é avaliada como $x + (y * z)$.
- No acetato seguinte apresenta-se uma tabela das precedências dos operadores em C#, da mais alta para a mais baixa:

Precedências dos operadores

Operadores	
Primários	x.y f(x) a[x] new
	typeof checked unchecked
Unários	x++ x--
	+ - ! ~ ++x --x (T)x
Multiplicação	* / %
Adição	+ -
Shift	<< >>
Relacionais	< > <= >= is as
Igualdade	== !=
E lógico	&
XOR lógico	^
OU lógico	|
E condicional	&&
OU condicional	||
Condicional	?:
Atribuição	= *= /= %= += -= <<= >>= &= ^= |=

Tipos de dados

- Numéricos
  - Inteiros (short, int e o long)
  - Reais (float e o double)
- Caracteres
  - Armazenados internamente pelo computador como inteiros que variam entre 0 e 255 (ver tabela ASCII).
  - O tipo string permite armazenar conjuntos de carateres, como nomes de pessoas.
- Sem tipo (void)

Tipo	**Intervalo de valores**
sbyte (System.SByte)	Inteiro entre –128 e 127
byte (System.Byte)	Inteiro entre 0 e 255
short (System.Int16)	Inteiro entre –32768 e 32767
ushort (System.UInt16)	Inteiro entre 0 e 65535
int (System.Int32)	Inteiro entre –2147483648 e 2147483647
uint (System.UInt32)	Inteiro entre 0 e 4294967295
long (System.Int64)	Inteiro entre –9223372036854775808 e 9223372036854775807
ulong (System.UInt64)	Inteiro entre 0 e 18446744073709551615
float (System.Single)	Real entre -3.4×10^{38} e $+3.4 \times 10^{38}$, com 7 dígitos significativos
double (System.Double)	Real entre $\pm 5.0 \times 10^{-324}$ e $\pm 1.7 \times 10^{308}$, com 15-16 dígitos significativos
decimal (System.Decimal)	Real entre $-7.9x10^{28}$ e $7.9x10^{28}$, com 28 digitos significativos
char (System.Char)	Carater Unicode (pode ser guardado como um inteiro entre 0 e 65535)
string (System.String)	Cadeia de carateres
bool (System.Boolean)	Valor booleano, true ou false

Entrada e saída de dados

- As funções mais comuns para a entrada de dados são:
  - Console.Read(), que lê o próximo carater
  - Console.ReadLine(), que lê uma linha de careteres
- As funções mais comuns para a saída de dados são:
  - Console.Write(), que escreve um texto no ecrã
  - Console.WriteLine(), que escreve uma linha de texto no ecrã
  - Para mostrar valores temos de os indicar no texto que queremos mostrar e passar os respetivos valores. Ex:
    Console.Writeline("Ola {0}", "Antonio");
    Console.Writeline("{0}+{1}={2}", 3,4,3+4);

Entrada de dados

- Como o C# trata todas as interações com a consola como texto temos de fazer as devidas conversões para as variáveis que temos. Exemplo:

```
using System;

public class Idade
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Como te chamas?");
        string line = Console.ReadLine();
        string nome = line;
        Console.WriteLine("{0}, em que ano nasceste?", nome);
        line = Console.ReadLine();
        int ano = Convert.ToInt32(line);
        int idade = 2015 - ano;
        Console.WriteLine("{0}, tens {1} anos.", nome, idade);
        System.Threading.Thread.Sleep(5000);
    }
}
```

Escrita formatada de números

Formato	Tipo	Exemplo	(float) 1.42	(int) -12400
C	Currency	{0:C}	$1.42	-$12,400
D	Decimal	{0:D}	FormatException	-12400
		{0:10D}	FormatException	-0000012400
E	Scientific	{0:E}	1.420000e+000	-1.240000e+004
F	Fixed point	{0:F}	1.42	-12400.00
		{0:F3}	1.420	-12400.000
		{0, 10:F2}	_____1.42	___-12400.00
G	General	{0:G}	1.42	-12400
N	Number (milhares)	{0:N}	1.42	-12,400
R	Round trippable	{0:R}	1.42	FormatException
P	Percentagem	{0:P}	142.00%	-1 240 000.00%
X	Hexadecimal	{0:x4}	FormatException	cf90

Escrita formatada de números - exemplos

```
using System;

class Program
{
    static void Main()
    {
            Console.WriteLine("{0}", 5);
            Console.WriteLine("{0:D}", 5);
            Console.WriteLine("{0:D10}", 5);
            Console.WriteLine("{0,5:D}", 5);
            Console.WriteLine("{0}-{0}-{0}", 5);
            Console.WriteLine("{0}{1}", 5,5);
            Console.WriteLine("{0,5:D}", 5);
            Console.WriteLine("{0,5:D}", 55);
            Console.WriteLine("{0,5:D}", 555);
            Console.WriteLine("{0,5:D}{1,-5:D}", 5, 5);
            Console.WriteLine("{0,5:D}{1,-5:D}", 55, 55);
            Console.WriteLine("{0,5:D}{1,-5:D}", 555, 555);
    }
}
```

Escrita formatada de texto

Para colocar no ecrã o texto à esquerda ou à direita pode-se utilizar o mesmo formato dos números sem especificar o tipo de dados. Pode-se também utilizar as funções PadRight(Int32)/PadRight(Int32,Char) e PadLeft(Int32)/PadLeft(Int32,Char). Estas funções constroem novas strings com base nas iniciais, preenchendo-as como o carater escolhido. Se não for escolhido nenhum carater utilizam o espaço. Exemplos:

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("{0,-10}{0,10}{0,-10}","Antonio");
        Console.WriteLine("{0,-10}{0,10}{0,-10}","Jorge");

        Console.Write("Letter".PadRight(10));
        Console.WriteLine("Number".PadLeft(8));

        Console.Write("A".PadRight(10));
        Console.WriteLine("14".PadLeft(8));
        Console.Write("B".PadRight(10));
        Console.WriteLine("4".PadLeft(8));
        Console.Write("C".PadRight(10));
        Console.WriteLine("123".PadLeft(8));

    }
}
```

Escrita formatada de texto

Pode-se utilizar a função Format da String para escrever as strings alinhadas à esquerda, direita, etc.

```
using System;

public class Idade
{
   public static void Main(string[] args)
   {
      Console.WriteLine("-----------------------------------------");
      Console.WriteLine("Primeiro nome | Ultimo nome   | Idade");
      Console.WriteLine("-----------------------------------------");
      Console.WriteLine(String.Format("{0,-10} | {1,-10} | {2,5}", "Bill", "Gates", 51));
      Console.WriteLine(String.Format("{0,-10} | {1,-10} | {2,5}", "Edna", "Parker", 114));
      Console.WriteLine(String.Format("{0,-10} | {1,-10} | {2,5}", "Johnny", "Depp", 44));
      Console.WriteLine("-----------------------------------------");
   }
}
```