# Informações em um computador

Prof. Ricardo Frohlich da Silva

# Informações em um computador

- Todo o trabalho realizado por um computador é baseado na manipulação das informações contidas em sua memória.

- Estas informações podem ser classificadas em dois tipos: instruções e dados.

# Instruções

- Comandam o funcionamento da máquina e determinam a maneira como devem ser tratados os dados;

- Exemplos de instruções:
  - Cálculos;
  - Mostrar informações na tela;
  - Ler informações do teclado;
  - Testar condições;
  - etc.

# Dados

- Dados propriamente ditos, que correspondem à porção das informações a serem processadas pelo computador.

```
Algoritmo soma_numeros

    n1, n2, resultado;

Inicio

  n1 = 5;

  Ler(n2);

  resultado = n1+n2;

  Mostrar(resultado);

Fim
```

**n1**, **n2** e **resultado** são dados, enquanto **Ler**, **Mostrar**, e **resultado = n1+n2** são instruções que manipulam os dados.

# Informações em um computador

- No princípio, os computadores eram dedicados quase exclusivamente ao tratamento de dados numéricos.
- Hoje em dia, o computador armazena os dados de tipos diferentes de formas diferentes e os manipula com instruções diferentes.
- Por isso, é importante identificar e diferenciar os tipos básicos de dados.

# Classificação dos dados

- Observação!
  - A classificação apresentada a seguir não se aplica a nenhuma linguagem de programação específica.
  - Ela sintetiza os padrões utilizados na maioria das linguagens de programação.

# Classificação dos dados

- Dados geralmente são classificados de acordo com algum dos seguintes tipos:

  - Numéricos;
  - Caracteres;
  - Lógicos/Booleanos

# Dados Numéricos

- Os dados numéricos podem ser basicamente de dois tipos:

  - Inteiros:
    - caracterizando números inteiros (não fracionários) tanto positivos quanto negativos.
      - Por exemplo: `0, 35, -56`;

  - Ponto flutuante/Reais:
    - São caracterizados como tipos reais, os dados numéricos positivos e negativos e números fracionários.
      - Por exemplo: `35.0, 0.0, -56.0, 1.2, -45.987`.

# Dados Numéricos

- IMPORTANTE!

  - Por motivos de precisão e limites de representação em um computador, não é correto afirmar que os conjuntos representados pelos dados inteiros e ponto flutuante sejam equivalentes aos conjuntos matemáticos dos inteiros e dos reais, respectivamente.

# Caracteres

- São caracterizados como tipos caracteres as sequências contendo letras, números e símbolos especiais.
- Quando se trata de caracteres isolados, os mesmos são indicados entre aspas simples:
  - `'a', 'B', '-', '~', 'ç', '5', ' ';`

# Caracteres

- Qual a diferença entre `5`,`'5'` e `5.00`?
  - `5` é um dado do tipo inteiro;
  - `5.00` é um dado do tipo real/ponto flutuante;
  - `'5'` é um caractere!
- Com quais deles eu posso fazer operações matemáticas?
  - Resposta:

# Caracteres

- Qual a diferença entre `5`,`'5'` e `5.00`?

  - `5` é um dado do tipo inteiro;
  - `5.00` é um dado do tipo real/ponto flutuante;
  - `'5'` é um caractere!

- Com quais deles eu posso fazer operações matemáticas?

  - Resposta: somente com os dados numéricos: `5` e `5.00`

# Sequência de caracteres/Strings

- Quando tratamos de uma sequência de caracteres, esta deve ser indicada entre aspas duplas (“ ”).

- Este tipo de dado também é conhecido como alfanumérico, String, literal ou cadeia.

- Exemplos:
  - `“Programação”, “Rua Alfa, 52 Apto 1”, “Fone 574-9988”, “04387-030”.`

# Lógicos/Booleanos

- São dados com valor verdadeiro/true ou falso/false, sendo que dados desse tipo podem representar apenas um dos dois valores.

# Variáveis

Prof. Ricardo Frohlich da Silva

# Variáveis

- Para armazenar os dados na memória, imagine que a memória de um computador é um grande armário com várias gavetas;

- Cada gaveta pode armazenar apenas um único valor (seja ele numérico, caractere ou booleano).

- Se é um grande armário com várias gavetas, é necessário identificar com um nome a gaveta com o dado que se pretende utilizar.

- Desta forma o valor armazenado pode ser utilizado a qualquer momento.

# Variáveis

- Têm-se como definição de variável tudo aquilo que é sujeito a variações, que é incerto, instável ou inconstante.

- Num algoritmo, variável é uma entidade que possui um valor, sendo identificada por um nome (identificador).

- O conceito de variável na matemática é bastante familiar, por isso é interessante que se faça uma analogia.

# Variáveis

- Por exemplo, a fórmula para o cálculo da área do quadrado:

```
Área = lado x lado
```

- Esta fórmula possui a variável `lado` que permite que a fórmula seja genérica, isto é, possa ser usada para calcular a área de qualquer quadrado.

- Ou seja:
  - Se `lado = 5, Área = 5 x 5 = 25;`
  - Se `lado = 6, Área = 6 x 6 = 36.`

# Variáveis

- Observando ainda a fórmula da área do quadrado:

```
Área = lado x lado
```

- Esta fórmula também possui a variável `Área`, que irá receber o resultado da multiplicação!

# Variáveis

- No computador, todo dado que será manipulado por um programa deve ser identificado, e armazenado na memória durante a execução. Isto é feito através de variáveis.

- Basicamente, uma variável possui três atributos: um nome, um tipo de dado associado à mesma e a informação por ela guardada.

# Variáveis

- As variáveis em algoritmos devem ser declaradas de forma a definir seu NOME e TIPO.
- A declaração de variáveis acontece no início do programa, antes de serem utilizadas.
- Regras para os nomes de variáveis:
  - Conter um ou mais caracteres;
  - Primeiro caractere do nome da variável não pode ser um número (sempre uma letra);
  - Não pode conter espaços em branco;
  - O nome é constituído por letras (sem acentuação e cedilha) e números, e excluem-se os demais símbolos, exceto o underline (_).

# Variáveis

São nomes de variáveis, por exemplo:

```
n

valor

v

n1

n2

x

minha_variavel

_valor

a_b_c
```

# Variáveis

NÃO são nomes de variáveis:

```
1n

1variavel

variável

área

resultado final

çedilha

2
```

# Exemplos em pseudocódigo

```
Algoritmo calcula_media1

    nota1, nota2, soma, media: Real;



Inicio
  Ler(nota1);
  Ler(nota2);
  soma = nota1 + nota2;
  media = soma/2;


  Mostrar(media);
Fim
```

# Exemplos em pseudocódigo

```
Algoritmo calcula_media2

    nota1, nota2, media: Real;



Inicio

  Ler(nota1);

  Ler(nota2);

  media = (nota1+nota2)/2;

  Mostrar(media);

Fim
```

# Exemplos em pseudocódigo

```
Algoritmo calcula_diferenca_idade1

    idade1, idade2, diferenca: Inteiro;



Inicio

  Ler(idade1);

  Ler(idade2);

  diferenca = idade1 - idade2;

  Mostrar(diferenca);

Fim
```

# Exemplos em pseudocódigo

```
Algoritmo calcula_diferenca_idade2

    idade1, idade2: Inteiro;

Inicio

  Ler(idade1);

  Ler(idade2);

  Mostrar(idade1-idade2);

Fim
```

# Exemplos em pseudocódigo

```
Algoritmo calcula_diferenca_idade_v2
    idade1, idade2, diferenca: Inteiro;
Inicio
  Ler(idade1);
  Ler(idade2);
  diferenca = idade1 - idade2;
  Se diferenca < 0 Então
     diferenca = diferenca * -1;

  Mostrar(diferenca);
Fim
```

# Linguagem de Programação C#

Prof. Ricardo Frohlich da Silva

# Linguagem C#

- C# é uma linguagem de programação, multiparadigma, de tipagem forte, desenvolvida pela Microsoft como parte da plataforma .NET.

- A sua sintaxe orientada a objetos foi baseada no C++ mas inclui muitas influências de outras linguagens de programação, como Object Pascal e, principalmente, Java.

- O código fonte é compilado para Common Intermediate Language (CIL) que é interpretado pela máquina virtual Common Language Runtime (CLR).

- C# é uma das linguagens projetadas para funcionar na Common Language Infrastructure da plataforma .NET Framework.

# Linguagem C#

- C# é uma linguagem de programação, multiparadigma, de tipagem forte, desenvolvida pela Microsoft como parte da plataforma .NET.

- A sua sintaxe orientada a objetos foi baseada no C++ mas inclui muitas influências de outras linguagens de programação, como Object Pascal e, principalmente, Java.

- O código fonte é compilado para Common Intermediate Language (CIL) que é interpretado pela máquina virtual Common Language Runtime (CLR).

- C# é uma das linguagens projetadas para funcionar na Common Language Infrastructure da plataforma .NET Framework.

# Características da Linguagem C#

- É uma linguagem compilada:
  - código fonte → código objeto → código executável (linguagem de máquina).

- Todo comando termina em ‘;‘ (ponto e vírgula).

# Características da Linguagem C#

- Outras características:
  - Início e final de blocos representados por { e }
  - Comentários entre /* */ ou //para comentar 1 linha

Comentários são trechos de texto que o compilador irá ignorar, pois são somente “comentários” que colocamos no código

# Tipos de dados da Linguagem C#

- Tipos básicos de dados
  - char (caractere).
    - Ex: `char letra1 = 'A';`

  - int (inteiro).
    - Ex: `int valor1 = 54;`

- float (ponto flutuante).
  - Ex: `float valor2 = 43.6778;`

- double (ponto flutuante de precisão dupla).
  - Ex: `double valor3 = 32.45345346;`

# Tipos de dados da Linguagem C#

| Tipo | Tamanho aprox. em bits | Faixa |
|---|---|---|
| char | 8 | -127 a 127 |
| int | 16 | -32.767 a 32.767 |
| unsigned int | 16 | 0 a 65.535 |
| short int | O mesmo que int | |
| long int | 32 | -2.147.483.647 a 2.147.483.647 |
| unsigned long int | 32 | 0 a 2.294.967.265 |
| float | 32 | 6 dígitos de precisão |
| double | 64 | 10 dígitos de precisão |
| long double | 80 | 10 dígitos de precisão |

# Variáveis na Linguagem C#

- Variável é uma posição nomeada de memória, que possui um nome para identificação e corresponde a um tipo de dado.

- Declaração de variáveis na linguagem C#:
  - `TIPO_DE_DADO nome_var1, nome_var2;`

- É possível na declaração definir um valor inicial da variável.

# Variáveis na Linguagem C#

- Exemplos de declaração:

  - `int x, y, z=10;`

  - `float a = 3, c = 4.23, d;`

  - `char i, j = 'a';`

# Operadores na Linguagem C#

- Operadores são símbolos utilizados para realizar operações lógicas e aritméticas sobre operandos.
- Operadores aritméticos atuam sobre variáveis, constantes e funções numéricas e produzem um resultado numérico.
- Operadores possuem prioridades em relação aos outros, ou seja, qual operação será executada primeiro em relação as demais.

# Operadores na Linguagem C#

- A tabela abaixo apresenta os operadores da linguagem C# e suas prioridades:

| Prioridade | Operador | Operação |
|---|---|---|
| 1 | - | Inversão de sinal |
| 2 | * | Multiplicação |
| 2 | / | Divisão |
| 2 | % | Resto de Divisão |
| 3 | + | Adição |
| 3 | - | Subtração |

# Operadores na Linguagem C#

- Os operadores que possuem mesma prioridade são executados na ordem em que aparecem quando a expressão é lida da esquerda para a direita.

- O uso dos parênteses define uma ordem de execução prioritária em relação à prioridade dos operadores da linguagem.

- Exemplos:
  - `x + 5 * y - 4`
    - a primeira operação realizada é a multiplicação, em seguida é executada a soma e por fim a subtração.

  - `(x+5) * (y-4)`
    - para estabelecer uma precedência diferente, podem ser utilizados os parênteses.

    - Neste caso, primeiro é realizado a soma, depois a diferença, e por fim a multiplicação.

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | | | |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | =a*3 | | |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | =3*3 | | |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | | |

# Exemplo completo em C

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | =b/2 | |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | =9/2 | |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | 4.5 | |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | 4.5 | =a+c |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | 4.5 | =3+4.5 |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | 4.5 | 7.5 |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | 4.5 | =c-(d/2)+c*d |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | 4.5 | =4.5-(7.5/2)+4.5*7.5 |

Como foi utilizado o próprio valor de `d` na fórmula, é utilizado então o valor atual de `d` antes da execução da operação, ou seja, `7.5`

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | 4.5 | 34.5 |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | =a % 2 | 34.5 |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | =3 % 2<br>?????? | 34.5 |

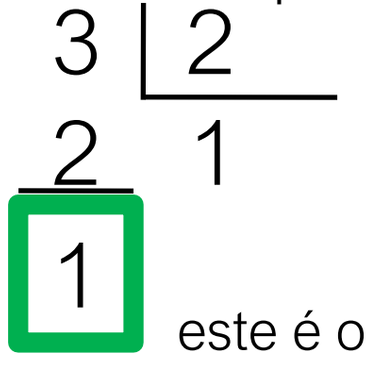
# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | =3 % 2<br>?????? | 34.5 |

O símbolo % representa o resto da divisão.

Ou seja, neste caso, se dividirmos 3 por 2:




este é o resto da divisão!

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | 1 | 34.5 |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | =c+d | 34.5 |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | =1+34.5 | 34.5 |

# Exemplo completo em C#

```
static void Main(string[] args)
{
  int a;
  float b, c, d;
  a = 3;
  b = a * 3;
  c = b / 2;
  d = a + c;
  d = c - (d/2) + c * d;
  c = a % 2;
  c = c + d;
}
```

| int a | float b | float c | float d |
|---|---|---|---|
| 3 | 9 | =35.5 | 34.5 |

# Operadores de incremento e decremento

- **Muitas vezes será necessário realizar o incremento (somar determinado valor) ou decremento (subtrair determinado valor) em algumas variáveis, por exemplo:**

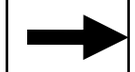
```
static void Main(string[] args)

{
    int a=5, b=10;
    a = a-1;   //a = 4
    b = b+1;   //b = 11
    a = a+5;   //a = 9
    b = b-3;   //b = 8
  }
```

Obs: isto é um comentário!!

# Operadores de incremento e decremento

- A maioria das linguagens possibilita realizar este tipo de operação de uma forma mais simples:

```
static void Main(string[] args)
        {
int a=5, b=10;
    a = a-1;
    b = b+1;
    a = a+5;
    b = b-3;
        }
```



```
static void Main(string[] args)
        {
int a=5, b=10;
a--; //ou a-=1
b++; //ou b+=1
    a+=5;
    b-=3;
        }
```

# Comandos de entrada e saída

- Para permitir a interação do usuário com os programas, existem os comandos que fazem a entrada (leitura) e saída (escrita) de dados.

- O dispositivo padrão de entrada é o teclado e o dispositivo padrão de saída de dados é o monitor de vídeo.

# Comando de saída

- **Console.WriteLine** (`string de controle`)

- Na string de controle iremos colocar os dados que serão impressos e, se necessário, o formato que estes serão exibidos.

# Comando de saída

- Exemplo 1: mostrar um texto na tela:

```
static void Main(string[] args)

{

        Console.WriteLine(“Eita mundão sô!");

}
```

- Quando queremos mostrar somente um texto fixo, sem variáveis, basta colocar o texto entre aspas duplas

# Comando de saída

- Exemplo 2: mostrar um texto e o valor de uma variável na tela:

```
static void Main(string[] args)
{
    int x;
    x = 5;
    Console.WriteLine("Você digitou: "+x);
}
```

# Comando de saída

- Exemplo 3: mostrar várias variáveis junto a um texto na tela:

```
static void Main(string[] args)
{
    int x = 5, y = 10;
    double z;
    z = 3.1415;
    Console.WriteLine("Os valores sao "+x+", "+y+" e "+z);
}
```

```
static void Main(string[] args)
{
    int idade1, idade2, diferenca;
    char letra1, letra2 = 'b';
    double valor1 = 5.788, valor2;
    letra1 = 'z';
    idade1 = 20;
    idade2 = 45;
    diferenca = idade2 - idade1;
    valor2 = valor1 - 2.088;
    Console.WriteLine("este é o meu primeiro algoritmo com saída de dados na tela");
    Console.WriteLine("a idade 1 é: "+idade1);
    Console.WriteLine("a idade 2 é: "+idade2);
    Console.WriteLine("a letra 1 é: "+letra1);
    Console.WriteLine("a letra 2 é: "+letra2);
    Console.WriteLine("o valor 1 é: "+valor1);
    Console.WriteLine("o valor 2 é: "+valor2);
    Console.WriteLine("a diferenca do valor1 para o valor 2 é: "+(valor1-valor2));
    Console.WriteLine("a diferença de idade é:"+diferenca);
    Console.WriteLine("a diferença de idade é: "+(idade2 - idade1));
    Console.WriteLine("mostrando tudo: "+idade1+", "+idade2+", "+letra1+", "+letra2+", "+valor1+", "+valor2);

}
```

# Comando de entrada

- **Console.ReadLine()**

  - Na linguagem C#, a leitura irá diferir para cada tipo de dado.

  - No `scanf` antes dos nomes das variáveis a serem lidas pelo teclado deve-se inserir um `&` (que indica o endereço da variável).

# Comando de entrada

- Exemplo 1: leitura de uma variável:

```
static void Main(string[] args)
{
    int x;
    x = int.Parse(Console.ReadLine());
}
```

# Comando de entrada

- Exemplo 2: leitura de uma variável:

```
static void Main(string[] args)
{
    int x;
    Console.WriteLine("Digite um numero: ");
    x = int.Parse(Console.ReadLine());
}
```

- Neste exemplo, antes de ler o valor para x, mostramos um texto informativo, pedindo para que o usuário digite um valor.

- Torna nossa interface mais amigável ☺

# Comando de entrada

- Exemplo 2: leitura de uma variável:

```
static void Main(string[] args)
{
    int x;
    Console.WriteLine("Digite um numero: ");
    x = int.Parse(Console.ReadLine());
    Console.WriteLine("Você digitou o numero "+x);
}
```

- E também podemos mostrar o que foi digitado anteriormente...

# Comando de entrada

- E para os outros tipos de dados ?
  - Para char -> char.Parse(Console.ReadLine());
  - Para double -> double.Parse(Console.ReadLine());

# Comando de entrada

- Exemplo 3: leitura de variáveis:

```
static void Main(string[] args)
{
    int x;
    char ch;
    double n2;

    Console.WriteLine("Digite um numero inteiro: ");
    x = int.Parse(Console.ReadLine());

    Console.WriteLine("Digite uma letra: ");
    ch = char.Parse(Console.ReadLine());

    Console.WriteLine("Digite um numero, pode ser com decimal: ");
    n2 = double.Parse(Console.ReadLine());

    Console.WriteLine("Você digitou o numero "+x);
    Console.WriteLine("Você digitou a letra " + ch);
    Console.WriteLine("Você o numero " + n2);
}
```

# Comando de entrada

- E para os outros tipos de dados ?
  - Para char -> char.Parse(Console.ReadLine());
  - Para double -> double.Parse(Console.ReadLine());
- Deu tudo certo?

# Comando de entrada

- E para os outros tipos de dados ?
  - Para char -> char.Parse(Console.ReadLine());
  - Para double -> double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

# Comando de entrada

- O que é CultureInfo.InvariantCulture?

# Comando de entrada

```
static void Main(string[] args)
{
    int x;
    char ch;
    double n2;

    Console.WriteLine("Digite um numero inteiro: ");
    x = int.Parse(Console.ReadLine());

    Console.WriteLine("Digite uma letra: ");
    ch = char.Parse(Console.ReadLine());

    Console.WriteLine("Digite um numero, pode ser com decimal: ");
    n2 = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);

    Console.WriteLine("Você digitou o numero "+x);
    Console.WriteLine("Você digitou a letra " + ch);
    Console.WriteLine("Você o numero " + n2);
}
```