Dado

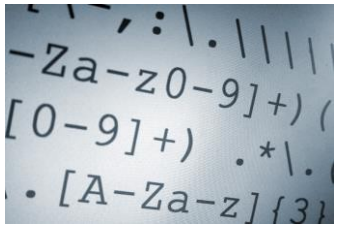
1. É uma unidade básica de informação, como: nome, sexo, data de nascimento, grau de instrução, idade.

   Exemplos: Carlos, masculino, 30/01/2018, superior, 50

2. É a representação numérica, alfanumérica, gráfica ou sonora de uma determinada realidade.

3. Pode ser manipulado (processado) e gerar novo dado.

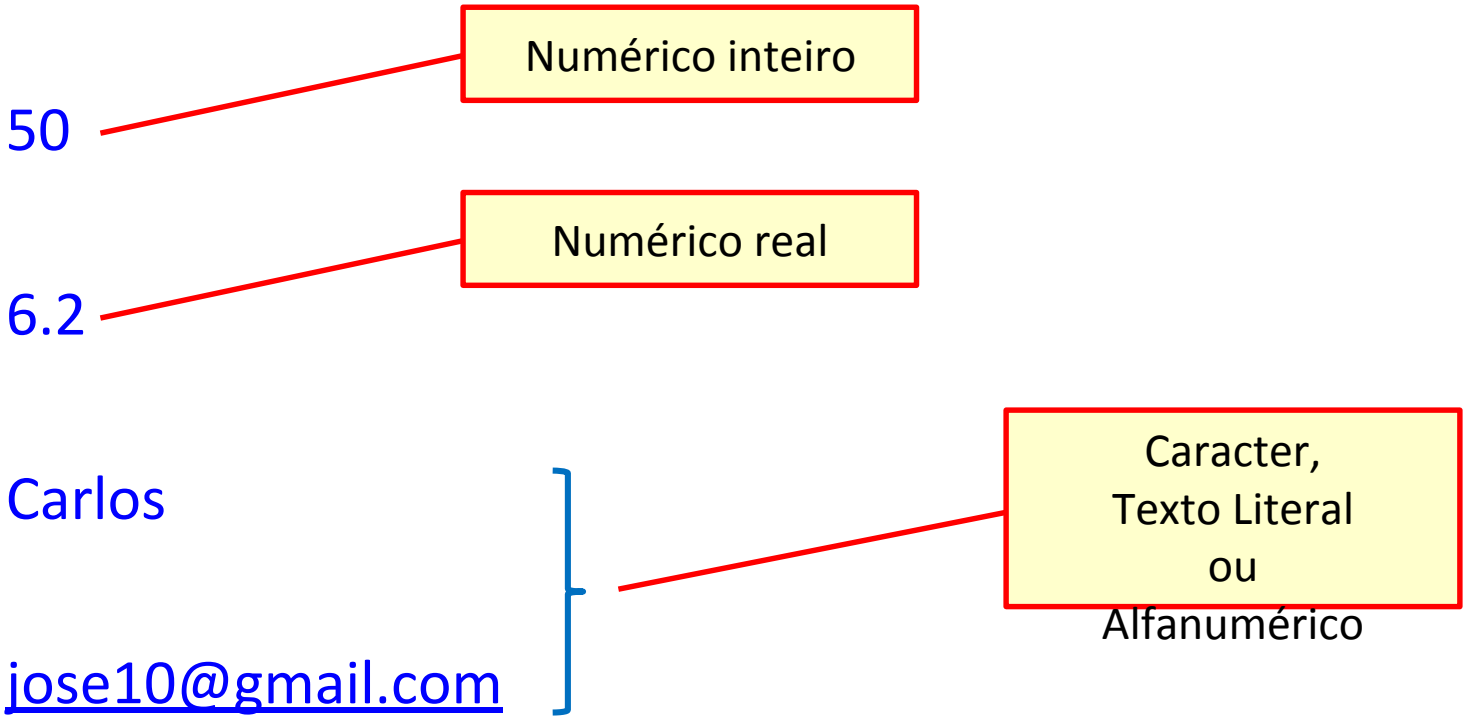
   Exemplo: 400/8 (resultando no dado 50)



Tipos Básicos de Dados

Conforme o conteúdo, um dado pode ser classificado por um determinado tipo:

Exemplos:



Tipos de Dados Primitivos

Leia o artigo do link abaixo:

https://dicasdeprogramacao.com.br/tipos-de-dados-primitivos/

Tipos de Dados Primitivos do C#

- **Inteiro com sinal**
  - sbyte: 8 bits, intervalo de -128 a 127
  - short: 16 bits, intervalo de –32.768 a 32.767
  - int: 32 bits, intervalo de –2.147.483.648 a 2.147.483.647
  - long: 64 bits, intervalo de –9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
- **Inteiro sem sinal**
  - byte: 8 bits, intervalo de 0 a 255
  - ushort: 16 bits, intervalo de 0 a 65.535
  - uint: 32 bits, intervalo de 0 a 4.294.967.295
  - ulong: 64 bits, intervalo de 0 a 18.446.744.073.709.551.615

Tipos de Dados Primitivos do C#

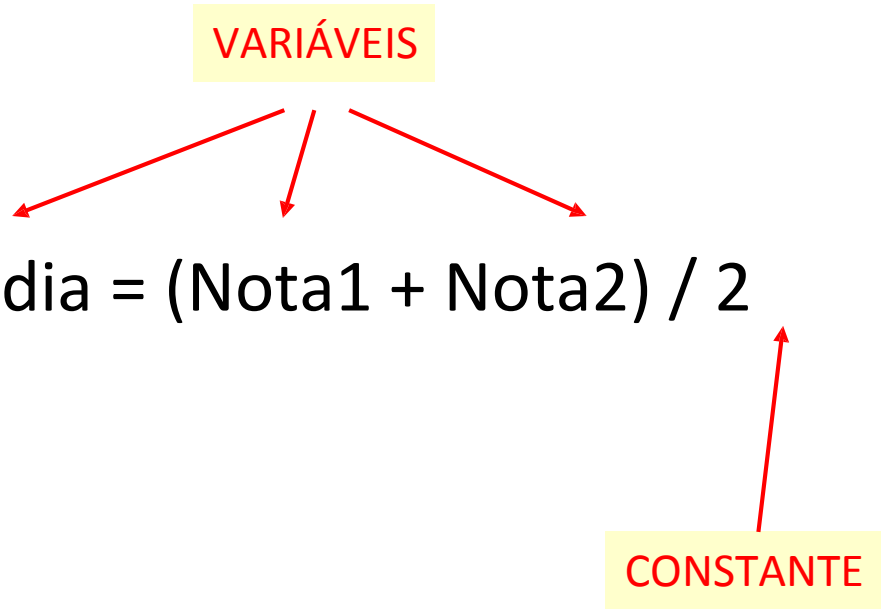
- **Ponto flutuante**
  - float: 32 bits, intervalo de $1,5 \times 10^{-45}$ a $3,4 \times 10^{38}$, precisão de 7 dígitos
  - double: 64 bits, intervalo de $5,0 \times 10^{-324}$ a $1,7 \times 10^{308}$, precisão de 15 dígitos
- **Decimal**
  - decimal: 128 bits, intervalo de pelos menos $-7,9 \times 10^{-28}$ a $7,9 \times 10^{28}$, com precisão de pelo menos 28 dígitos

Tipos de Dados Primitivos do C#

Tipo C#	Tipo .NET	Descrição	Faixa de dados
bool	System.Boolean	Booleano	true ou false
byte	System.Byte	Inteiro de 8-bit com sinal	-127 a 128
char	System.Char	Caracter Unicode de 16-bit	U+0000 a U+ffff
decimal	System.Decimal	Inteiro de 96-bit com sinal com 28-29 dígitos significativos	$1,0 \times 10^{-28}$ a $7,9 \times 10^{28}$
double	System.Double	Flutuante IEEE 64-bit com	$+-5,0 \times 10^{-324}$ a $+-1,7 \times 10^{324}$
float	System.Single	Flutuante IEEE 32-bit com	$+-1,5 \times 10^{-45}$ a $+-3,4 \times 10^{38}$
int	System.Int32	Inteiro de 32-bit com sinal	-2.147.483.648 a 2.147.483.647
long	System.Int64	Inteiro de 64-bit com sinal	-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807
Object	System.Object	Classe base	-
Sbyte	System.Sbyte	Inteiro de 8-bit sem sinal	0 a 255
Short	System.Int16	Inteiro de 16-bit com sinal	-32,768 a 32,767
String	System.String	String de caracteres Unicode	-
Uint	System.UInt32	Inteiro de 32-bit sem sinal	0 a 4,294,967,295
Ulong	System.UInt64	Inteiro de 64-bit sem sinal	0 a 18,446,744,073,709,551,615
Ushort	System.UInt16	Inteiro de 16-bit sem sinal	0 a 65,535

Processando Dados

Durante um processamento de dados o computador manipula dados que podem variar (VARIÁVEIS) e dados que não variam (C0NSTANTES):



Constante

É um dado com valor fixo que não se modifica durante o processamento (execução do programa).

1. Pode ser representado diretamente no programa

```
Media = (Nota1 + Nota2) / 2;
```

2. Em algumas linguagens pode ser declarado explicita e previamente no programa, o que não ocorre no VisualG.

```
const int QtdeNotas = 2;
Media = (Nota1 + Nota2) / QtdeNotas;
```

Variável

É um dado cujo valor pode variar durante o processamento.

1. Corresponde a uma posição de memória

2. Armazena apenas um valor a cada instante

3. Recebe um nome representativo (IDENTIFICADOR)
   Nome, Idade, SalarioBruto, PrecoCusto

Um nome de variável não deve começar por número, ter espaço ou (em algumas linguagens) caracteres especiais.

Declaração de Variável

Devem ser declaradas previamente e, na maioria das linguagens devem ter seu tipo declarado explicitamente.

Sintaxe do C#

```
<tipo> <nome_variavel>
```

Exemplos

```
string nome;

int idade;

double peso;

bool resultado;
```

- C# é case sensitive
- Não existem um local específico para declara uma variável.
- Boa prática: Notação CamelCase (UpperCamelCase e lowerCamelCase)

Fornecendo o Valor da Variável

Durante o processamento o valor de uma variável por ser fornecido das seguintes maneiras:

1. Por atribuição

Sintaxe do C#

<nome_variavel> **=** *<conteúdo>*

O conteúdo atribuído pode ser uma constante, uma variável ou uma expressão.

Exemplos:

```
nome = "José Carlos";
nota1 = 6;
media = (nota1 + nota2) / 2;
```

Fornecendo o Valor da Variável

Durante o processamento o valor de uma variável por ser fornecido das seguintes maneiras:

2. Por entrada via teclado

Sintaxe do C#

```
Console.ReadLine()
```

Todo conteúdo recebido pelo método Console.ReadLine() é recebido como texto pela variável interna do programa, necessitando ser convertido para ser armazenado em uma variável declarada.

Exemplos:

```
idade = Convert.ToInt32(Console.ReadLine());
nota1 = Convert.ToDouble(Console.ReadLine());
```

Exibindo Dados

Gera e exibe na tela uma linha de texto.

Sintaxe do C#

```
Console.Write([<(dado_1)> [+ <(dado_n)> ]])


Console.WriteLine([<(dado_1)> [+ <(dado_n)> ]])
```

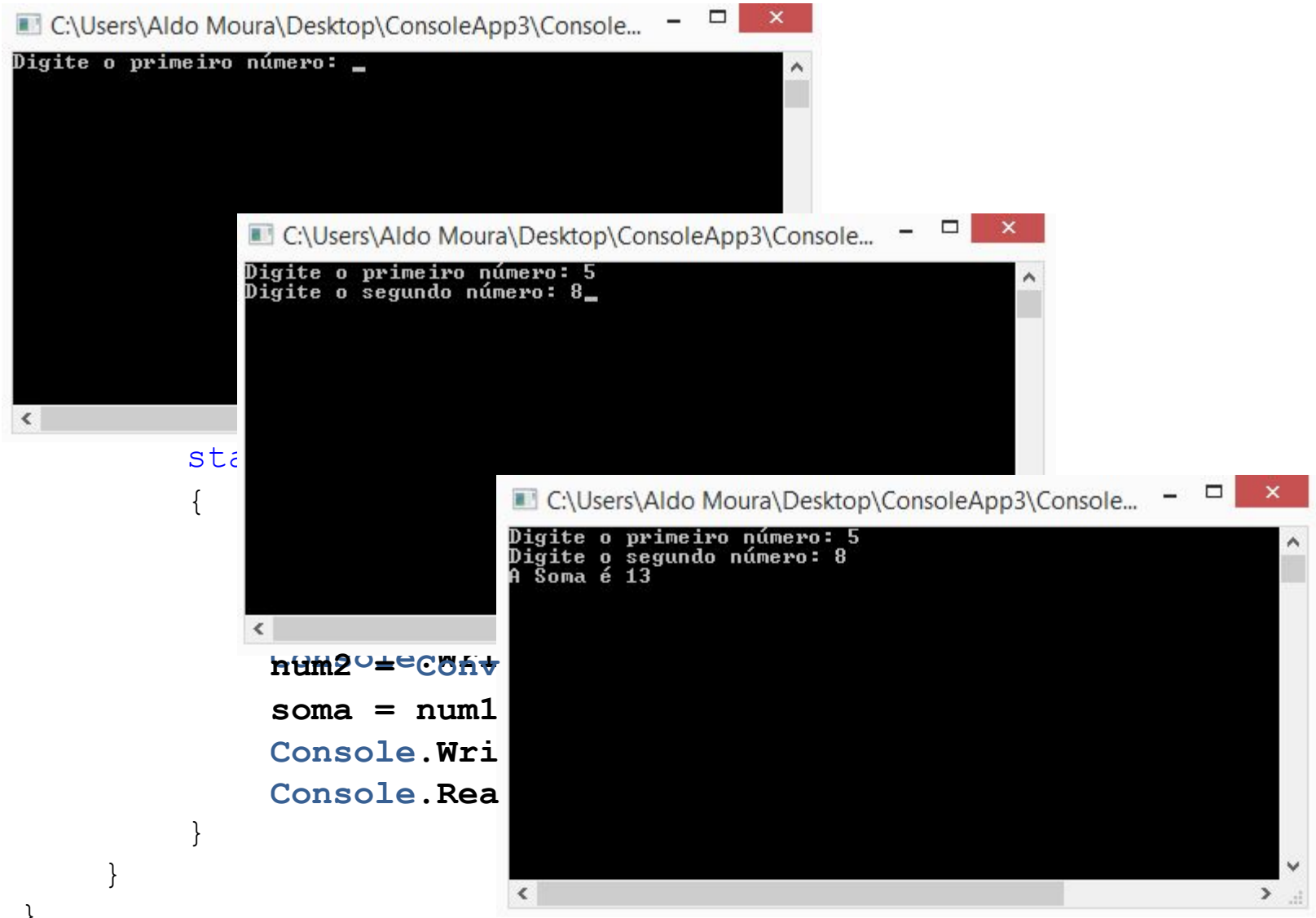
Exemplos:

```
Console.WriteLine("A Soma é " + soma)
```

Primeiro Programa

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SomarDoisNumeros
{
    class Program
    {
        static void Main(string[] args)
        {   int num1, num2, soma;
            Console.Write("Digite o primeiro número: ");
            num1 = Convert.ToInt32(Console.ReadLine());
            Console.Write("Digite o segundo número: ");
            num2 = Convert.ToInt32(Console.ReadLine());
            soma = num1 + num2;
            Console.Write("A Soma é " + soma);
        }   Console.ReadKey();
    }
}
```

Primeiro Programa




```
        sta
        {
                                    Conv
            num2 = Conv
            soma = num1
            Console.Wri
            Console.Rea
        }
    }
}
```

Operadores Aritméticos

Operador	Operação
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão inteira

- Utilizados para nas operações aritméticas com dados do tipo inteiro e real.
- A expressões aritméticas são escritas de forma linear.

Exemplos:

```
100 + 200      // resulta 300
300 – 50       // resulta 250
40 * 3 / 2     // resulta 60
5 % 2          // resta 1
```

Prioridades dos Operadores Aritméticos Básicos

Conforme regra da aritmética, a ordem de precedência dos operadores básicos são:

1º Multiplicação, divisão e resto da divisão inteira

2º Adição e subtração

Exemplos:

10 + 20 * 30 = ?

50 – 8 / 2 = ?

Prioridades dos Operadores Aritméticos Básicos

Em expressões com operadores de mesma prioridade, executa-se da esquerda para direita:

Exemplos:

10 + 20 - 30 = ?

50 * 8 / 2 = ?

20 + 10 * 150 – 60 / 3 = ?

Prioridades dos Operadores Aritméticos Básicos

A ordem de prioridade pode ser alterada através de parênteses.

Exemplos:

(10 + 20) * 30 = ?

(50 – 8) / 2 = ?

20 + 10 * (150 – 60) / 3 = ?

Operador de Caractere - Concatenação

OPERADOR
+

Junta o conteúdo de dados do tipo texto (string).

Exemplos:

- a= "E" + "TE" // Armazena ETE
- b = a + "MAC" // Armazena ETEMAC
- c = a + " " + "MAC" // Armazena ETE MAC

Etapa para elaboração de um Programa

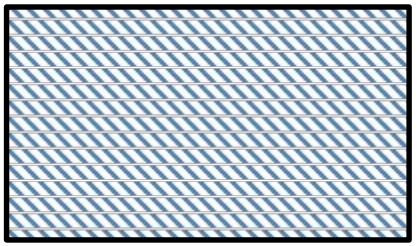
1. Entender o problema através da leitura do enunciado
2. Identificar a(s) entrada(s) de dado(s)

   (quais dados o programa deverá receber)
3. Identificar a(s) saída(s) de dado(s)

   (quais dados o programa deve gerar como resposta)
4. Determinar o processamento que precisa ser feito para que a(s) entrada(s) seja(m) transformada(s) em saída(s)
5. Construir o programa
6. Testar a solução e corrigir, se houver(em) erro(s)

1. Entender o Problema

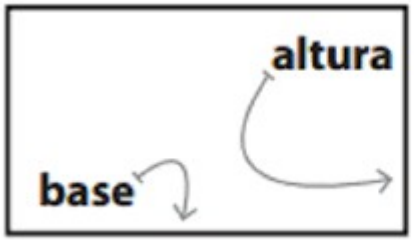
Calcular a área de um retângulo.



2. Identificar a(s) Entrada(s)

Calcular a área de um retângulo.

Quais os dados que são necessários para calcular a área de um retângulo?



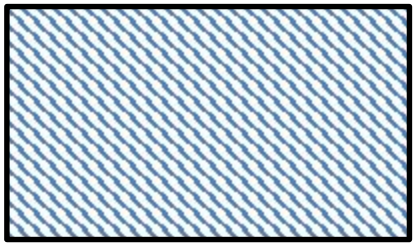
$A = b \cdot h$

Observando-se a fórmula que calcula a área de um retanângulo, verificamos que precisamos da base e da altura

3. Identificar a(s) Saída(s)

Calcular a área de um retângulo.

Que informação(ões) deseja-se encontrar? *A área*



4. Determinar o Processamento

4. Determinar o processamento que precisa ser feito para que a(s) entrada(s) seja(m) transformada(s) em saída(s)

Que processamento (fórmula, por exemplo) precisa ser executado para que a(s) saída(s) possa(m) ser gerada(s), a partir do(s) dado(s) de entrada?

A base do retângulo precisa ser multiplicada pela altura

5. Construir o Programa

Declarando as variáveis de entrada e saída


```
double base, altura, area;

Console.Write("Digite a base do retângulo: ");
base = Convert.ToDouble(Console.ReadLine());

Console.Write("Digite a altura do retângulo: ");
altura = Convert.ToDouble(Console.ReadLine());

area = base * altura;
Console.Write("A área do retângulo é: " + area);
Console.ReadKey();
```

Trabalhando com Percentual

Se o percentual for informado para achar o valor correspondente:

30 → 100%

X → 15%

X * 100 = 30 * 15

X = 30 * 15 / 100

VALOR DESEJADO = VALOR TOTAL * PERCENTUAL / 100

Se o valor for informado para achar o percentual correspondente:

30 → 100%

7 → X%

X% * 30 = 100 * 7

X% = 100 * 7 / 30

PERCENTUAL DESEJADO = 100 * VALOR CORRESPONDENTE / VALOR TOTAL

Comentário no Código Fonte

Usado como anotações relevantes no código fontes.
Seu conteúdo é ignorado pelo interpretador

Comentário de bloco

```
using System.Text;
using System.Threading.Tasks;
/*
 *Programa para somar dois números inteiros
 *Elaborado por: José Carlos
 *Data: 15/08/2019
*/
namespace SomarDoisNumeros
{
    class Program
    {
        static void Main(string[]  args)
        {   int num1, num2, soma;
```

Comentário no Código Fonte

Comentário de linha

```
using System.Text;
using System.Threading.Tasks;
namespace SomarDoisNumeros
{
    class Program
    {
        static void Main(string[] args)
        {
            int num1, num2, soma;  // Declaração de variável
            // Entrada de dados
            Console.Write("Digite o primeiro número: ");
            num1 = Convert.ToInt32(Console.ReadLine());
            Console.Write("Digite o segundo número: ");
            num2 = Convert.ToInt32(Console.ReadLine());
            soma = num1 + num2;
            // Saída de dados
            Console.Write("A Soma é " + soma);
            Console.ReadKey();
        }
    }
}
```

Formatação de Texto

Parâmetros de formatação

Sintaxe: `{I [, M] [:SF[n]]}`

Índice: Inteiro que indica a posição (na lista de argumentos) do argumento a ser formatado.

M (opcional): Largura do dado que será exibido.

Positivo: justifica o texto à direita
Negativo: justifica o texto à esquerda

SF (opcional): String para formatação de números

n (opcional: Quantidade de posições decimais

String	Formato
D	Decimal (inteiro)
F	Ponto fixo
N	Moeda
P	Porcentagem

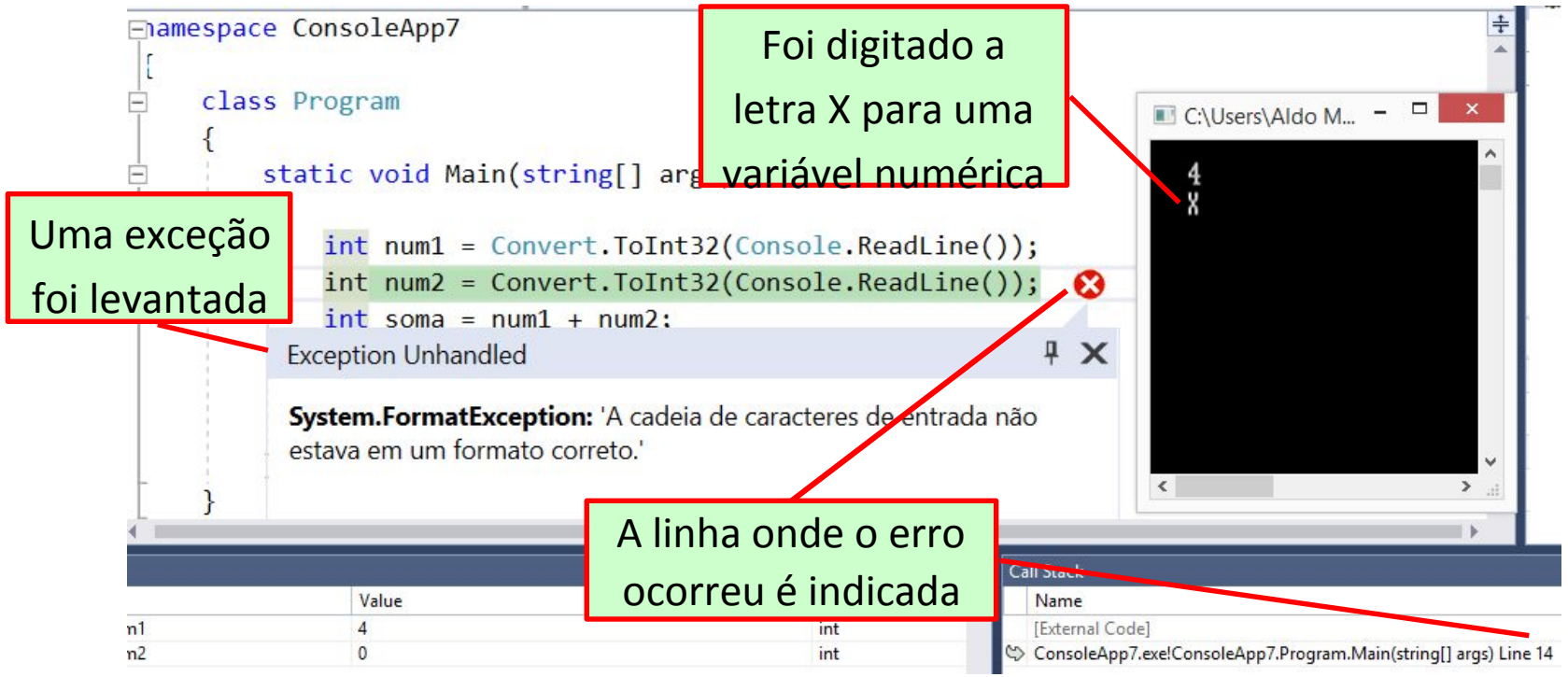
Formatação de Texto

```
int numero = 3;
double valor = 12458.615, desconto = 0.125;

Console.WriteLine("Dados armazenados: {0} | {1} | {2}", numero, valor, desconto);
Console.WriteLine();
Console.WriteLine("[{0,12:D}] Nr. inteiro alinhado a direita", numero);
Console.WriteLine("[{0,-12:D}] Nr. inteiro alinhado a esquerda", numero);
Console.WriteLine("[{0,12:F}] Nr. real com ponto fixo padrão", valor);
Console.WriteLine("[{0,12:F4}] Nr. real com ponto fixo definido", valor);
Console.WriteLine("[{0,12:N}] Moeda com centavos padrão", valor);
Console.WriteLine("[{0,12:N3}] Moeda com centavos definido", valor);
Console.WriteLine("[{0,12:P}] Percentual", desconto);
Console.WriteLine("[{0,12:P1}] Percentual", desconto);
```

```
Dados armazenados: 3 | 12458,615 | 0,125

[           3] Nr. inteiro alinhado a direita
[3          ] Nr. inteiro alinhado a esquerda
[    12458,62] Nr. real com ponto fixo padrão
[  12458,6150] Nr. real com ponto fixo definido
[   12.458,62] Moeda com centavos padrão
[  12.458,615] Moeda com centavos definido
[      12,50%] Percentual
[       12,5%] Percentual
```

Tratamento de Exceção

Quando uma exceção ocorre o programa em execução é cancelado e é apresentada uma mensagem de erro que, normalmente não são claras.



Exceção: São classes definidas pelo C# para o tratamento de erros.

Tratamento de Exceção

Recomenda-se ao programador agrupar as instruções que podem causar exceções, protegendo-as em um bloco.

Sintaxe

```
Try
{
     /* Bloco de instruções a serem executadas que
      * podem causar exceções
     /*
}
Catch
{  /*
     * Bloco de instruções que será executado caso
     * ocorra um erro na execução do bloco anterior
    /*
}
```

Tratamento de Exceção - Exemplo

```
int num1, num2, soma;
try
{
   num1 = Convert.ToInt32(Console.ReadLine());
   num2 = Convert.ToInt32(Console.ReadLine());
}
catch
{
   Console.WriteLine("Digite dados numéricos");
   Console.ReadKey();
   return;
}
soma = num1 + num2;
Console.WriteLine("{0}+{1}={2}",num1, num2, soma);
```

Funções de Caracteres

Substring(): Retorna uma parte de dado tipo texto.

Sintaxe: Substring(Posição inicial, Deslocamento)

O C# indexa uma string a partir da posição 0.

```
String CURSO DE JAVA
                    111
Posição 0123456789012
```

Exemplos:

```
nome = "IBRATEC";
parteNome = nome.Substring(1,3); // Retorna BRA

x = 3;
y = 10;
tema = "ESTRUTURA DE DADOS";
parte = tema.Substring(0,x) + "U" +  tema.Substring(y,2);
// Retorna ESTUDE
```

Funções de Caracteres

Length: Retorna o tamanho de um dado tipo texto.

Sintaxe: Length

Exemplos:

```
string nome = "IBRATEC";
int tamanho = nome.Length; // Retorna 7
```

Operadores Relacionais (dados numéricos)

OPERADOR	OBJETIVO
==	Igual
!=	Diferente
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual

Possibilita a elaboração de expressões lógicas.

Retorna um resultado True ou False.

Exemplos:

- `4 > 2` `// resulta em True`
- `5 == 8` `// resulta em False`

Operadores Lógicos: E (&&) e OU (||)

OPERADOR
&&
||

Permite a elaboração de expressões lógica com mais de um operador relacional.

Exemplos:

- `4 > 2 && 5 == 8`
- `30 < 20 || 5 < 8`

Tabela Verdade – E (&&)

&&		
V	V	**V**
V	F	**F**
F	V	**F**
F	F	**F**

Exemplos:

- `7 > 4  && 4<5  // True && True Resulta True`
- `4 > 2  && 5 == 8  // True && False Resulta False`
- `30 < 20 && 5 < 8  // False && True Resulta False`
- `7 < 2  && 15 > 30  // False && False Resulta False`

Tabela Verdade – OU (||)

V	V	**V**
V	F	**V**
F	V	**V**
F	F	**F**

Exemplos:

- `7 > 4 || 4<5 // True && True Resulta` **`True`**
- `4 > 2 || 5 == 8 // True && False Resulta` **`True`**
- `30 < 20 || 5 < 8 // False && True Resulta` **`True`**
- `7 < 2 || 15 > 30 // False && False Resulta` **`False`**

Operador Lógico: ! (NÃO)

OPERADOR
!

Nega o resultado da expressão lógica.

```
Exemplos:
    !(4 > 2)
// !(True) Resulta False

  30 < 20 || !(5 < 8)
//  True  || !(True)
//  True  || False Resulta False
```

PRIORIDADE DOS OPERADORES LÓGICOS

PRIORIDADE	OPERADOR
1º	!
2º	&&
3º	||

Exemplo:

```
   (6==3) || (5<8) && (3==3) || !((5<=4) && (3==2))
// False ||  True &&  True  || !( False && False)
// False ||  True &&  True  ||      !(False)
// False ||  True &&  True  ||       True
// False ||      True       ||       True
// True
```

Comparação entre Strings

OPERADOR
==

Compara se duas strings são iguais.

Valor do retorno	Condição
True	Se a String1 for igual a String2
False	Se a String1 for diferente da String2

Exemplos:

- `"A" == "A"` `// Retorna True`
- `"A" == "a"` `// Retorna False`
- `"IVO" == "IVONE"` `// Retorna False`

Comparação entre Strings

String.Equals: Compara duas strings.

Sintaxe: String.Equals(String1, String2)

Valor do retorno	Condição
True	Se a String1 for igual a String2
False	Se a String1 for diferente da String2

Exemplos:

- `String.Equals("A","A")` `// Retorna True`
- `String.Equals("A","a")` `// Retorna False`
- `String.Equals("IVO","IVONE")` `// Retorna False`

Comparação entre Strings

String.Compare: Compara duas strings.

Sintaxe: String.Compare(String1, string2)

Valor do retorno	Condição
< 0	A String1 precede a String2
0	A String1 é igual a String2
> 0	A String1 segue a String2

Exemplos:

```
String.Compare("A","B")                    // Retorna -1
String.Compare("A","a")                    // Retorna 1
String.Compare("CASA","CASO")              // Retorna 1
String.Compare("MARIO","MARTA")            // Retorna -1
String.Compare("IVO","IVONE")              // Retorna -1
String.Compare("LEGENDADO","LEGENDA")      // Retorna 1
String.Compare("100","70")                 // Retorna -1
String.Compare("315","38")                    Retorna
```

CLASSE MATH

Fornece constantes e métodos (funções) trigonométricas, logarítmicas entre outras funções matemáticas.

Math.Round: Arredonda um valor a quantidade de casas decimais especificado.

Math.Floor: Arredonda um valor para o inteiro mais baixo.

Math.Truncate: Arredonda para o número inteiro mais próximo em direção a zero.

```
double n1Double, n2Double, n3Double, n4Double;
int nInteiro;
n1Double = 15.92345;
n2Double = Math.Round(n1Double, 2);                    // 15,92 double
n3Double = Math.Round(n1Double, 0);                    // 16 double
n4Double = Convert.ToInt32(Math.Floor(n1Double));      // 15 double
nInteiro = Convert.ToInt32(Math.Truncate(n1Double));   // 15 inteiro
```