

# Tipos de Dados

# O que são Dados?

- Definição "formal": Dados são  *fatos individuais ou mensurações, obtidos e coletados por meio de técnicas de medição ou por observação direta ou indireta.*
- Tudo o que podemos observar ou medir de alguma forma, e coletar e guardar em algum meio de armazenamento, pode ser chamado de "dado".

Algumas definições de "dado" tiradas de dicionários e da Wikipedia incluem:

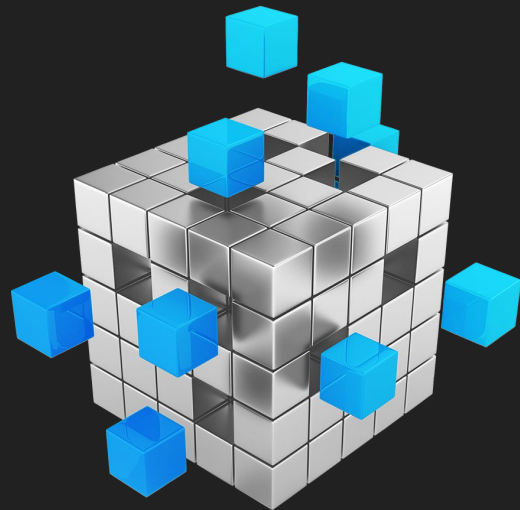
- **Cada um dos elementos conhecidos de um problema**
- **Registro do atributo de um ente, objeto ou fenômeno**

# Tipos de Dados

Dados são informações brutas a serem processadas por um computador.

São divididos em quatro tipos principais:

- Numéricos
- Caracteres
- Lógicos
- Referência (Compostos)



# Tipos Numéricos

Lidam com números. Podem ser subdivididos em dois tipos:

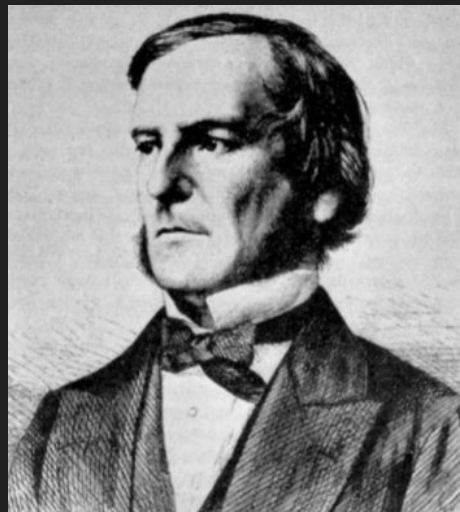
- **Inteiro** – números inteiros, positivos e negativos, como 5, 250 e -95.
- **Real** – números positivos, negativos e fracionários, como 5.0, 45.23, -20.6 e  $1/3$

# Tipos de Caractere

- Podem ser caracteres isolados ou sequências de caracteres que contenham letras, números e símbolos.
- Os caracteres devem ser representados sempre entre aspas ( ' ' ou " ") no código.
- Podemos usar termos como cadeia, string, alfanumérico ou char.

# Tipo Lógico

- São dados cujos valores somente podem assumir um de dois valores (estados lógicos): **verdadeiro** (True / 1) ou **falso** (False / 0).
- Também conhecido como tipo Booleano (*Boolean*), em homenagem a **George Boole** (criador da álgebra booleana).



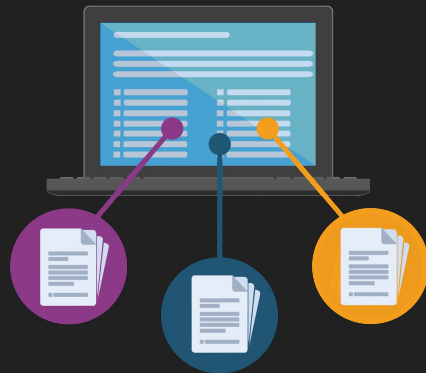
# Variáveis

# Variável

Local reservado na memória RAM do computador utilizado para armazenar temporariamente os dados que são utilizados pelo programa.

As variáveis possuem algumas características, como:

- Identificação ("nome")
- Endereço
- Tipo de dado
- Tamanho
- Valor ("conteúdo")

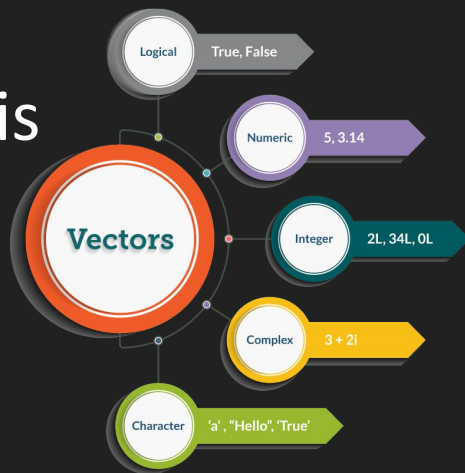




001 Idade	002	003	004	005
65				
006	007	008 Nome	009	010
		Ana		
011	012	013	014	015
016	017	018	019	020
021	022	023	024	025

# Nomes de variáveis - Convenções

- Podem ter um ou mais caracteres;
- O primeiro caractere sempre é uma letra;
- Não podem ser palavras reservadas da linguagem
- Não pode ter espaços em branco;
- Não podem ser usados caracteres especiais ou símbolos;
- Podem ser usados números;



# Declaração e Atribuição de Variáveis

# Declaração de variáveis

Para que uma variável possa ser usada em um programa, ela deve primeiro ser declarada, para que seja reservado espaço na memória para armazenamento de seus dados.

*tipo\_de\_dado nomes\_das\_variáveis*

- As variáveis podem ser declaradas no início do programa ou em outros locais do programa (escopo).

# Exemplo de declaração

programa

cadeia nomeAluno

caracter conceito

logico alunoAprovado

real N1, N2, N3, N4 *// Variáveis globais*

funcao inicio()

{

real media *// Variável local*

*código principal do programa*

}

}

# Atribuição de valores a variáveis

Para atribuir um valor a uma variável, usamos um ***operador de atribuição***:

**=** (sinal de igual)

Sintaxe:

**variável = valor**

# Exemplo de atribuição de valores

```
funcao inicio() {  
    nomeAluno = "Fábio"  
    N1 = 6.25  
    conceito = 'R'  
    media = 4.50  
    alunoAprovado = falso  
}
```

# Escopo de Variáveis



# Escopo das Variáveis

O Escopo de uma variável se refere ao local onde ela é declarada e acessada, e ao seu tempo de vida (duração).

As variáveis somente podem ser usadas dentro do escopo no qual foram criadas.



# Escopo das Variáveis

<b>Variáveis Globais</b>	<b>Variáveis Locais</b>
Visíveis em todo o programa (escopo amplo)	Visíveis somente no local onde foram declaradas (escopo restrito)
Declaradas no início do programa, ou em um local especial	Declaradas dentro de funções e métodos.
Existe durante todo o tempo de vida do programa.	Existe somente enquanto a função na qual foi declarada estiver sendo executada
Permitem o compartilhamento de dados entre funções	Dados somente podem ser usado dentro da função

# Exemplo

## programa

```
// Variável global
real numero = 10.0
funcao inicio()
{
    // Variável local
    real media = 12.0
    escreva(numero + "\n")
    escreva(media + "\n")
    escreva(calcula() + "\n")
    // escreva(num1 + "\n")
}
```

```
funcao real calcula()
{
    // Variável local
    real num1 = 5.0
    retorne num1 * numero
    // retorne num1 * media
}
}
```

# Constantes

# Constantes

- **Constante** é uma posição na memória cujo valor **não muda** ao longo da execução do programa.
- Por exemplo, o valor de **Pi** é uma constante, pois é sempre o mesmo (3,1415...)
- Recomenda-se nomear constantes apenas com letras maiúsculas.

# Declaração de Constantes

## Sintaxe

`const tipo NOME_CONSTANTE`

## Exemplo

`const real PI = 3.1415`



# Entrada e Saída de Dados

# Entrada de dados

## Instrução **leia**

Lê valores digitados no teclado e os armazena em variáveis na memória.

```
leia(variável1, variável2,...)
```

Exemplo:

```
leia(valor) //Armazena um dado na variável valor
```



# Saída de dados

## Instrução **escreva**

- Escreve dados na tela do computador. Os dados podem ou não estarem armazenados em variáveis

**escreva(valor1, valor2,...)**

## Exemplos:

```
escreva("Pizza")      //Escreve na tela a palavra Pizza
escreva(500)         //Escreve o valor 500
escreva(valor)       //Escreve o conteúdo da variável valor
escreva("1\n")        //Escreve o valor 1 como caractere e vai para a próxima linha
```

# Exemplo de Entrada e Saída de dados

```
programa {  
    inteiro idade  
    cadeia nome  
    real salario  
  
    funcao inicio()  
        escreva("Digite seu nome:\n ")  
        leia(nome)  
        escreva("Entre com sua idade:  
            \n")  
        leia(idade)  
        escreva("Informe seu  
            salário:\n ")  
  
        leia(salario)  
        escreva("Nome: ", nome,  
            "\n")  
        escreva("Idade: ", idade,  
            "\n")  
        escreva("Salário: ",  
            salario)  
    }  
}
```

# Operador de Concatenação

(operador literal / de caractere)

**Concatenação:** junção de duas ou mais sequências de caracteres (strings), formando uma nova sequência.

**Operador:** +

*Exemplo:*

```
cadeia texto = " Treinamentos "  
escreva("Bóson" + texto + "em Tecnologia!")
```

# Concatenação com valores numéricos

```
limpa()
```

```
cadeia texto = " Treinamentos "
```

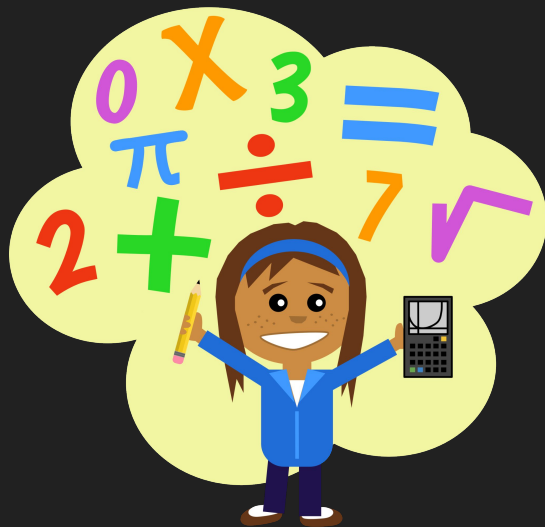
```
escreva("Bóson" + texto + "é a número " + 1)
```

*(A função **limpa()** é usada para limpar a tela (saída do console))*

# Operadores Aritméticos

# Operadores Aritméticos

Usados na realização de cálculos aritméticos simples, usando as quatro operações básicas da matemática mais operações como o módulo.



# Operadores Aritméticos

Operador	Operação
+	Soma
-	Subtração / Negação
*	Multiplicação
/	Divisão
%	Módulo (MOD)

# Expressões Aritméticas

$$x = 2 * 3$$

$$y = 5 * 2 * 2$$

$$z = 4 \% 2$$

$$w = 8 / 4$$

$$k = 7 / 2$$

$$m = w + x * 3$$

$$n = (m - k) / 2.0$$



# Exemplo: programa para somar dois números

```
programa {  
    inteiro x, y, z  
    funcao inicio() {  
        escreva("Entre com um número: ")  
        leia(x)  
        escreva("Entre com outro número: ")  
        leia (y)  
  
        // Realizar a soma dos dois números:  
        z = x + y  
        escreva ("A soma dos números é ", z)  
    }  
}
```

# Exemplo: Calcular $\Delta$ da equação de segundo grau

```
programa {  
    inteiro a, b, c, delta  
    funcao inicio() {  
        escreva("Digite o valor de a: ")  
        leia(a)  
        escreva("Digite o valor de b: ")  
        leia(b)  
        escreva("Digite o valor de c: ")  
        leia(c)  
        escreva("A equação é " + a + "x2 + " + b + "x + " + c + " = 0")  
  
        // Calcular  $\Delta$   
        delta = b * b - 4 * a * c  
        escreva ("\nDelta da equação é ", delta)  
    }  
}
```

# Operadores de Incremento e Decremento

# Incremento e Decremento

- **Incrementar: aumentar o valor de um item (por padrão, em 1 unidade)**
- **Decrementar: diminuir o valor de um item (por padrão, em 1 unidade)**

Operador	Exemplo de Uso	Operação Equivalente
<b>++</b>	<b>x++</b>	<b><math>x = x + 1</math></b>
<b>--</b>	<b>x--</b>	<b><math>x = x - 1</math></b>

# Operador de Incremento - Exemplo

```
inteiro n1
funcao inicio() {
    escreva("Digite um número: ")
    leia(n1)
    escreva("\nNúmero escolhido: " + n1)
    n1++
    escreva("\nNovo valor de n1: " + n1)
    n1--
    n1--
    escreva("\nAgora n1 vale: " + n1)
}
```

# Operadores Relacionais

# Operadores Relacionais

- Permitem estabelecer uma relação entre dois valores - seus *operandos*
- Comparam o valor à esquerda com o valor à direita do operador
- Funcionam com quaisquer tipos de dados - desde que os operandos sejam do mesmo tipo.
- Retornam um valor lógico dependendo do resultado da comparação
- Também conhecidos como *Operadores de Comparação*

# Operadores Relacionais

Operador	Operação
==	Igual a
!=	Diferente de
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a



# Expressões relacionais

`a == 5`

`b == 6`

`a < b`

`a == b`

`a != b`

`b != a + 1`

`a * 2 >= b`

`inteiro a = 5`

`inteiro b = 5`

*As expressões relacionais sempre retornam um valor lógico (**verdadeiro** ou **falso**)*

# Exemplo - Operadores Relacionais

```
programa {  
    logico x, y, z  
    inteiro n1, n2  
  
    funcao inicio() {  
        escreva("Digite um número: ")  
        leia(n1)  
        escreva("Digite outro número: ")  
        leia (n2)
```

```
        escreva("\nSão iguais?\n")  
        x = n1 == n2  
        escreva(x + "\n")  
  
        escreva("\nSão diferentes?\n")  
        y = n1 != n2  
        escreva(y + "\n")  
  
        escreva("\n" + n1 + " maior que "  
+ n2 + "?\n")  
        z = n1 > n2  
        escreva(z + "\n")  
    }  
}
```

# Operadores Lógicos *e ou não*


# Operadores Lógicos ou Booleanos

- Permitem trabalhar com múltiplas condições relacionais na mesma expressão
- Os mais usados são: **e**, **ou**, **nao**.
- Retornam valores lógicos (*verdadeiro / true* ou *falso / false*).

# Operador Lógico E

Condição A	Condição B	Resultado
Falso	Falso	Falso
Verdadeiro	Falso	Falso
Falso	Verdadeiro	Falso
Verdadeiro	Verdadeiro	Verdadeiro

Tabela-verdade  
do operador  
lógico e



Somente retorna **verdadeiro** se todas as condições de entrada forem verdadeiras.

A = falso

B = verdadeiro

C = A e B

*C é igual a falso*

# Operador Lógico OU

Condição A	Condição B	Resultado
Falso	Falso	Falso
Verdadeiro	Falso	Verdadeiro
Falso	Verdadeiro	Verdadeiro
Verdadeiro	Verdadeiro	Verdadeiro

Somente retorna ***falso*** se todas as condições de entrada forem falsas.

A = falso

B = verdadeiro      *C é igual a verdadeiro*

C = A ou B

# Operador Lógico NAO

Entrada	Saída
Falso	Verdadeiro
Verdadeiro	Falso

**Inverte** a condição de entrada: verdadeiro se torna falso, e falso se torna verdadeiro. Trata-se de um operador unário **inversor**.

A = verdadeiro

B = nao A

*B é igual a falso.*

```
programa {  
    caracter j1, j2, j3  
    logico estado  
  
    funcao inicio() {  
        j1 = 'f'  
        j2 = 'f'  
        j3 = 'a'  
        escreva("Janela 01 aberta? " + (j1 == 'a'))  
  
        escreva("\nAlguma janela aberta? ")  
        estado = j1 == 'a' ou j2 == 'a' ou j3 == 'a'  
        escreva(estado)  
  
        escreva("\nAlarme desligado? ", nao estado)  
  
        escreva("\nTodas as janelas abertas? ")  
        estado = (j1 == 'a' e j2 == 'a' e j3 == 'a')  
        escreva(estado)  
    }  
}
```

## Problema de exemplo

*3 janelas: j1, j2 e j3.*

*f = fechada*

*a = aberta*