NExT

Nova Experiência de Trabalho

Tatyane Calixto
tscs@cesar.org.br

Erick Simões
esm@cesar.org.br

C.E.S.A.R

Os breakout rooms já estão abertos!

AGENDA

19:05 - 19:30 Resolução de exercícios

Resolução de questões das listas de exercícios passados

19:30 - 22:00 Lista de exercícios de revisão

Lista de exercícios com questões que envolvem todos os conteúdos já abordados

19:05 - 22:00 21 - Save Room

Sala especial reservada aos estudantes que não estiveram presentes nas aulas ao vivo ou estão sentindo dificuldade com os primeiros conteúdos

PORTUGOL

PORTUGOL STUDIO

TUTORIAL

Para este curso, usaremos a linguagem de programação **Portugol**. Para criar e executar os códigos, adotaremos a IDE **Portugol Studio**.

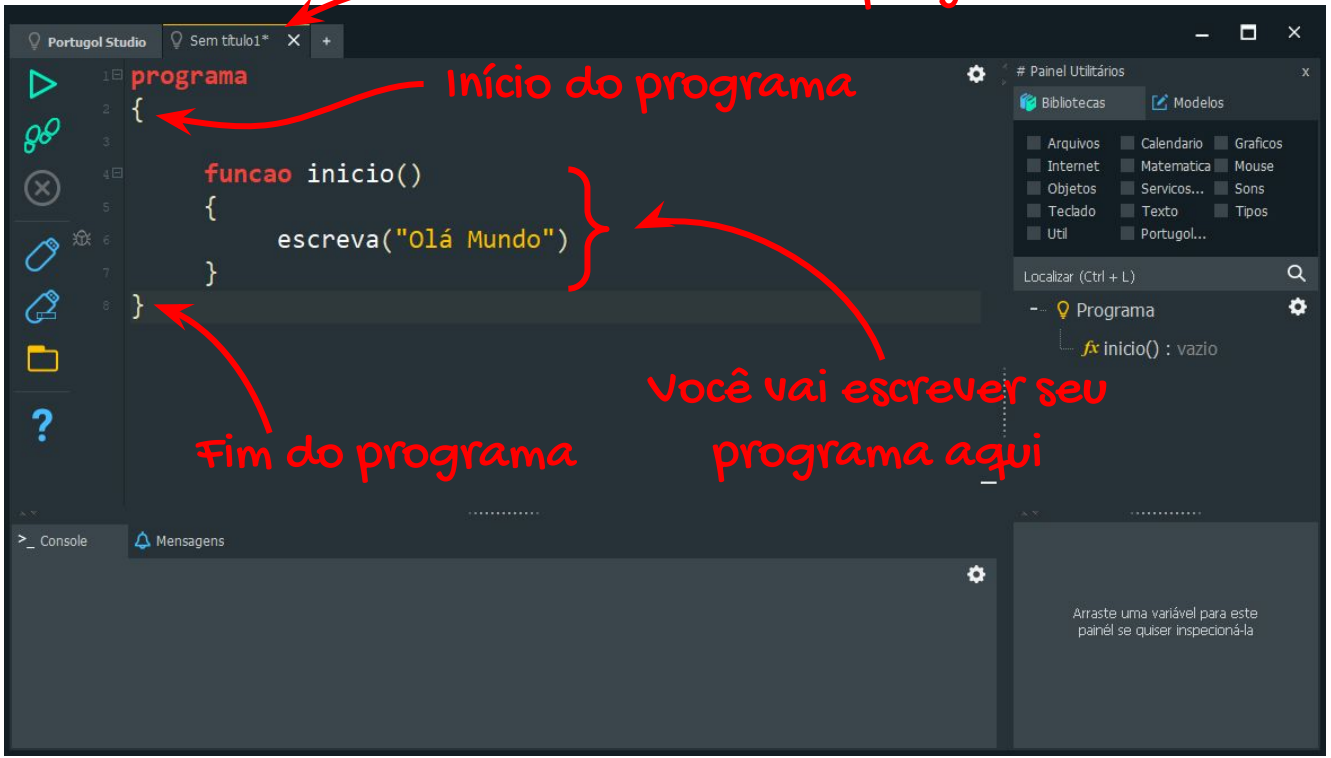
Portugol Studio:
http://lite.acad.univali.br/portugol/

Portugol WebStudio:
portugol-webstudio.cubos.io

Portugol Mobile (para Android):
https://play.google.com/store/apps/details?id=br.erickweil.portugolweb

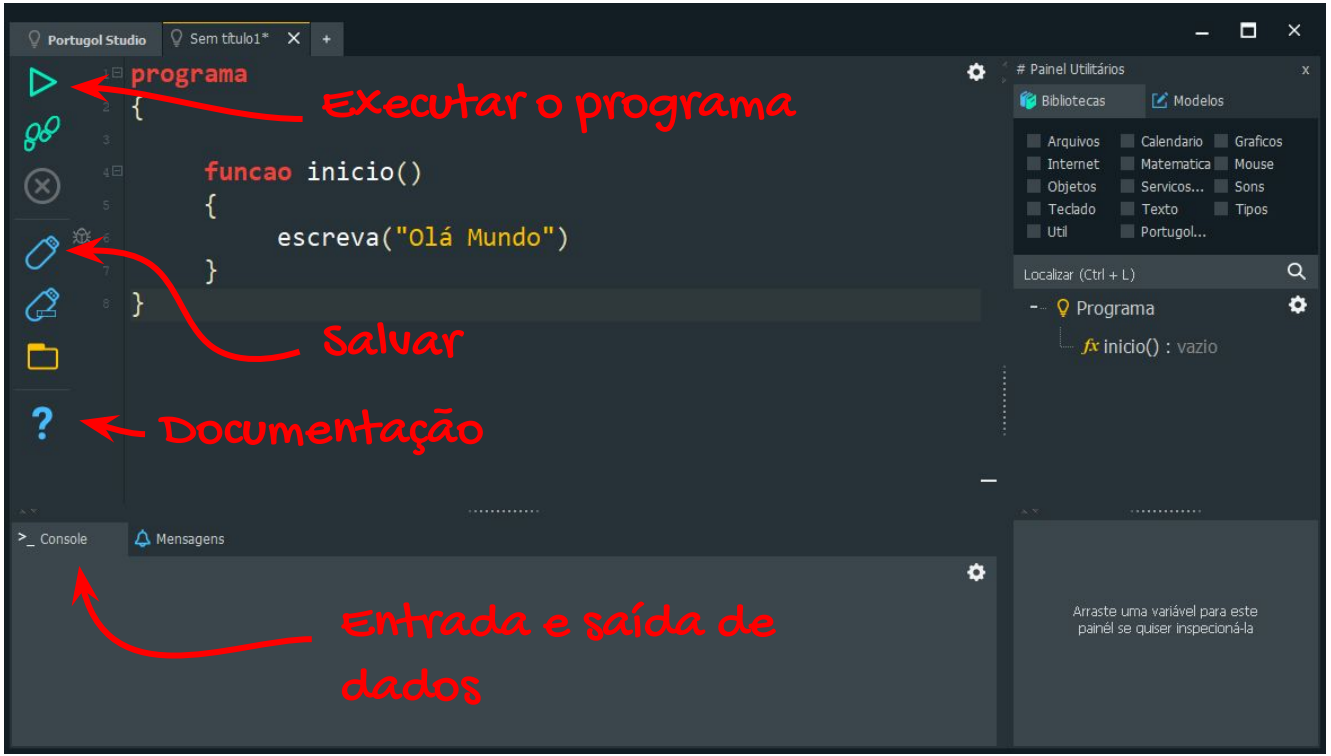
PORTUGOL STUDIO

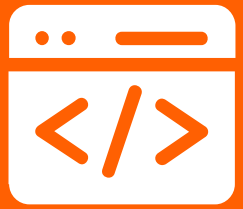
Conhecendo o Portugol Studio



PORTUGOL STUDIO

Conhecendo o Portugol Studio





PROGRAMAR

Programar
Variáveis

Mas antes...

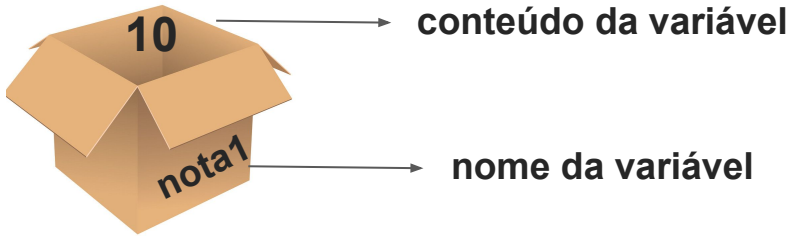
Vamos conhecer alguns conceitos básicos:

- **Comentários**
  - Pedaço de código que será ignorado;
  - Muito usado para explicar/documentar o programa ou para testes.

```
// comentário de linha

/*
    comentário de bloco
    contendo várias linhas
*/
```

Programar
Variáveis

- **Variáveis**
  - Na programação, uma variável é um objeto (uma posição, frequentemente localizada na memória) capaz de reter e representar um valor ou expressão.
  - "Espaço" em memória para guardar um valor durante a execução de um programa.
  - Devem ser declaradas antes de serem utilizadas



Programar
Tipos de Variáveis

- **Variáveis Numéricas**

Podem ser divididos em dois tipos:

- **Tipo inteiro** - Podem ser positivos, negativos ou nulos, mas não possuem componente decimal:
  Ex.: `5; -9; 0; 189; -800.`

- **Tipo real** - Podem ser positivos, negativos ou nulos, e possuem componente decimal, marcado pelo ponto (.):
  Ex.: `5.89; -6.8978; 0.00; 7.986; -1458.252.`

ℹ️ *Os inteiros são compatíveis com os reais, mas os reais não são compatíveis com os inteiros;*

ℹ️ *Os inteiros consomem menos espaço de armazenamento na memória.*

Programar
Variáveis

- **Variáveis Textuais**

Armazenam valores de texto. Podem ser divididos em dois tipos:

- **Tipo caracter** - Contém uma informação com apenas um caracter. Esse caracter pode ser uma letra, número ou pontuação.
  Ex.: `'a'; '7'; '!';`

- **Tipo cadeia** - Contém uma informação composta por vários caracteres, como um nome ou um endereço.
  Ex.: `"Em exemplo de texto"`

ℹ️ Caracteres são definidos usando aspas simples (‘a’) e cadeias são definidos usando aspas duplas (“exemplo”).

Programar
Variáveis

- **Variáveis Lógicas**

Armazena dados booleanos, ou seja, eles podem assumir dois valores possíveis: **`verdadeiro`** ou **`falso`**.

Dados Cadastrais
Nome: João Guilherme Idade: 30 Endereço: Rua João Pinho, 123 Peso: 85,5 Altura: 1,90 IMC: 23,7 Peso Ideal: (x) Sim ( ) Não

IMC	Classificação
abaixo de 18,5	abaixo do peso
entre 18,6 e 24,9	Peso ideal (parabéns)
entre 25,0 e 29,9	Levemente acima do peso
entre 30,0 e 34,9	Obesidade grau I
entre 35,0 e 39,9	Obesidade grau II (severa)
acima de 40	Obesidade III (mórbida)

Programar
Variáveis

Declaração e atribuição de variáveis

No código, a criação de uma variável é chamada de **declaração**. Ao colocar um valor na variável, chamamos de **atribuição** (ou inicialização). Geralmente, as declarações ocorrem no topo de programa.

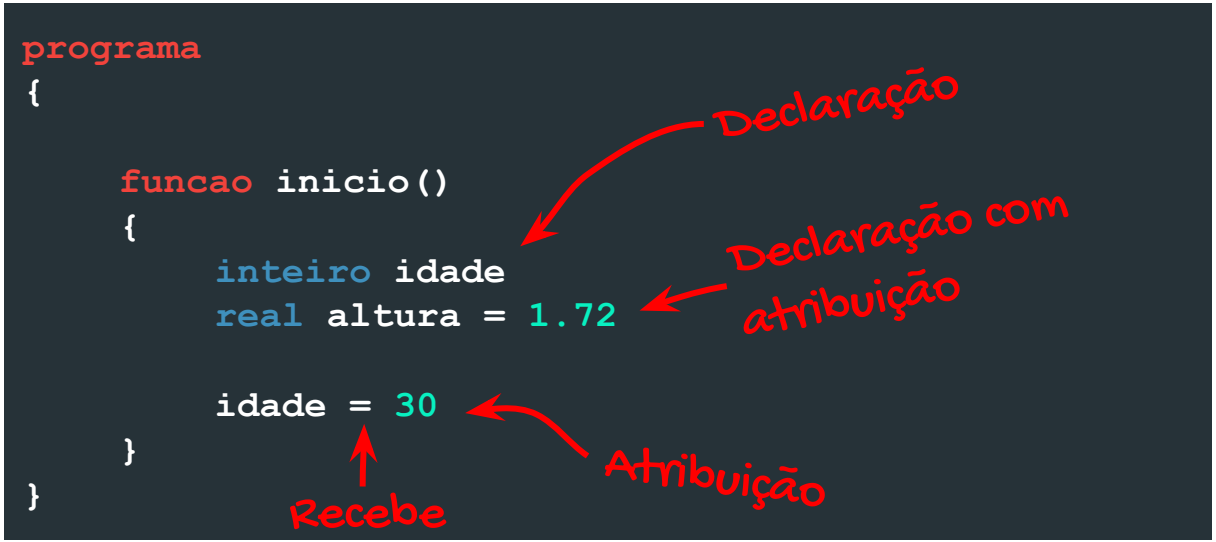
```
programa
{

    funcao inicio()
    {
        inteiro idade
        real altura
        caracter turma
        cadeia nome
        logico dirige
    }
}
```

Aqui as variáveis estão sendo declaradas, mas não estão recebendo valores

Programar
Variáveis

Declaração e atribuição de variáveis

No código, a criação de uma variável é chamada de **declaração**. Ao colocar um valor na variável, chamamos de **atribuição** (ou inicialização). Geralmente, as declarações ocorrem no topo de programa.



Programar
Exibir dados

Exibindo informações no console

O comando usado para exibir informações no console é o:

```
escreva("Olá mundo!")
```

É possível exibir o valor de uma variável no console passando-a como parâmetro:

```
inteiro idade = 30
escreva(idade)
```

Ainda é possível passar vários valores, separados por vírgula:

```
inteiro idade = 30
escreva("Sua idade é ", idade)
```

Programar
Receber dados

Recebendo informações no console

O comando usado para receber informações no console é o:

```
leia(nome_variavel)
```

Com esse comando, é possível receber um valor do usuário e atribuí-lo a uma variável:

```
inteiro idade
leia(idade)
escreva("Sua idade é ", idade)
```

Programar
Constantes

Constantes

Existem algumas situações em que precisamos trabalhar com um determinado parâmetro que não é alterado durante a execução do programa. Para isso existem as constantes.

Constante é um identificador cujo valor associado não pode não pode ser alterado pelo programa durante a execução.

```
const inteiro ALTURA_MAXIMA = 190
const real PI = 3.14
const real PLANCK = 6.62607
const real ACELERACAO_GRAVIDADE = 9.8
```

ℹ️ O nome de uma constante deve ser escrito todo em maiúsculo!

Programar
Resumo

Resumo:

Tipos de variáveis:

- `inteiro, real, caracter, cadeia, lógico`

Declaração de uma variável:

- `[tipo] [nome]`

Atribuição:

- `[nome] = [valor]`

Declaração com atribuição:

- `[tipo] [nome] = [valor]`

Variável x Constante:

- **Variável**: o valor pode mudar ao decorrer da execução do código
- **Constante**: o valor não muda depois que foi declarado

Exibir e receber informações:

- `escreva()`
- `leia()`

Variáveis

EXEMPLOS

EXEMPLO 1:

Declare duas variáveis inteiras chamadas “x” e “y”.

Receba do usuário, o valor de cada uma das variáveis.

Exiba os valores recebidos no console e sua soma.

Variáveis

EXEMPLOS

EXEMPLO 1:


```
programa
{

    funcao inicio()
    {
        inteiro x
        inteiro y

        leia(x)
        leia(y)

        escreva("x = ", x, " y = ", y, "\n")
        escreva("soma = ", x + y)
    }
}
```

Variáveis

EXEMPLOS

EXEMPLO 2:

Solicite ao usuário as seguintes informações:

- Nome completo;
- Ano de nascimento;
- Altura;
- Última letra do seu primeiro nome.

Exiba todas as informações recebidas no console.

Variáveis

EXEMPLOS

EXEMPLO 2:


```
programa
{

	funcao inicio()
	{
		cadeia nome
		inteiro ano
		real altura
		caracter ultima_letra

		escreva("Informe seu nome: ")
		leia(nome)
		escreva("Informe seu ano de nascimento: ")
		leia(ano)
		escreva("Informe sua altura: ")
		leia(altura)
		escreva("Informe a última letra do seu nome: ")
		leia(ultima_letra)

		escreva(nome, ' ', ano, ' ', altura, ' ', ultima_letra)
	}
}
```

Variáveis

EXEMPLOS

EXEMPLO 3:

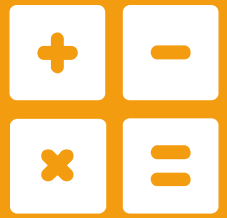
Solicite ao usuário seu ano de nascimento. Com base nesta informação, informe a idade do usuário.

Variáveis

EXEMPLOS

EXEMPLO 3:


```
programa
{

    funcao inicio()
    {
        inteiro ano_nascimento, idade

        escreva("Informe o ano do seu nascimento: ")
        leia(ano_nascimento)

        idade = 2022 - ano_nascimento

        escreva("Idade: ", idade)
    }
}
```



OPERAÇÕES BÁSICAS

Operações aritméticas

Operações Aritméticas Básicas

É possível realizar operações aritméticas entre variáveis numéricas. Para isso, usamos os operadores aritméticos:

`+, -, *, /, %`

```
10 + 5 // soma
10 - 5 // subtração
10 * 5 // multiplicação
10 / 5 // divisão
10 % 5 // resto da divisão
```

Operações Aritméticas

EXEMPLOS

EXEMPLO 4:

Declare duas variáveis inteiras chamadas “n1” e “n2”.

Receba do usuário um valor para cada variável.

Exiba a soma, subtração, multiplicação, divisão e resto da divisão dos valores no console.

Operações Aritméticas

EXEMPLOS

EXEMPLO 4:


```
programa
{

    funcao inicio()
    {
        inteiro n1, n2

        escreva("Informe o 1° número: ")
        leia(n1)
        escreva("Informe o 2° número: ")
        leia(n2)

        escreva("Soma: ", n1 + n2, "/n")
        escreva("Subtração: ", n1 - n2, "/n")
        escreva("Multiplicação: ", n1 * n2, "/n")
        escreva("Divisão: ", n1 / n2, "/n")
        escreva("Resto da divisão: ", n1 % n2, "/n")
    }
}
```

Operadores
Relacionais

Operadores Relacionais

- Realizam comparações e retornam valores lógicos: **verdadeiro** ou **falso**;

Operadores Relacionais	Portugol Studio
Maior	>
Menor	<
Maior ou igual	>=
Menor ou igual	<=
Igual	==
Diferente	!=

Variáveis

EXEMPLOS

EXEMPLO 5:

Receba dois valores inteiros do usuário. Exiba o resultado dos operadores relacionais entre esses números (>, <, >=, <=, ==, !=).

Variáveis

EXEMPLOS

EXEMPLO 5:

```
programa
{

    funcao inicio()
    {
        inteiro n1, n2

        leia(n1, n2)

        escreva("n1 > n2: ", n1 > n2, "\n")
        escreva("n1 < n2: ", n1 < n2, "\n")
        escreva("n1 >= n2: ", n1 >= n2, "\n")
        escreva("n1 <= n2: ", n1 <= n2, "\n")
        escreva("n1 == n2: ", n1 == n2, "\n")
        escreva("n1 != n2: ", n1 != n2, "\n")
    }
}
```

Operadores
Lógicos

Operadores Lógicos

- Servem para combinar resultados de expressões retornando se o resultado final é VERDADEIRO ou FALSO.

Operadores Lógicos	Portugol Studio	Significado
Conjunção	**e**	O resultado será **verdadeiro** se uma parte **e** a outra parte forem verdadeiras
Disjunção	**ou**	O resultado será **verdadeiro** se uma parte **ou** a outra parte forem verdadeiras
Não Lógico	**nao**	O resultado será a inversão do valor lógico. Se for **verdadeiro**, torna-se **falso**.

Variáveis

EXEMPLOS

EXEMPLO 6:

Receba o gênero e a idade do usuário e verifique se ele deve realizar o alistamento militar.

Variáveis

EXEMPLOS

EXEMPLO 6:

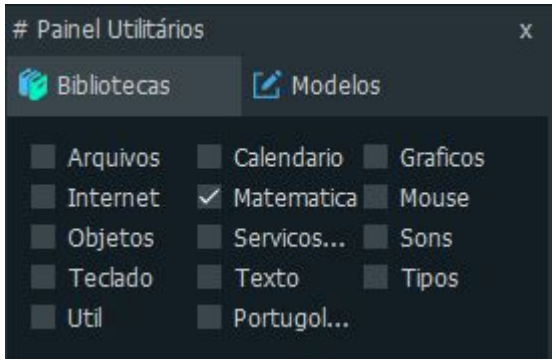
```
programa
{

     funcao inicio()
     {
          cadeia genero
          inteiro idade

          leia(genero)
          leia(idade)

          escreva("Deve se alistar ", genero == "masculino"
e idade == 18)
     }
}
```

BIBLIOTECAS

Bibliotecas
Matemática

Bibliotecas

- Chamamos de biblioteca um conjunto de soluções já desenvolvidas para resolver um problema conhecido.
- O Portugol Studio oferece uma série de bibliotecas já implementadas para uso.
- Algumas delas são:
  - Matematica
  - Texto
  - Calendario
  - Util



Bibliotecas
Matemática

Bibliotecas

- Para usar uma biblioteca, basta incluí-la no início do código:

```
programa
{
	inclua biblioteca Matematica

	funcao inicio()
	{
		escreva(Matematica.potencia(10, 2))
	}
}
```

```
programa
{
	inclua biblioteca Matematica --> mat

	funcao inicio()
	{
		escreva(mat.potencia(10, 2))
	}
}
```

Variáveis

EXEMPLOS

EXEMPLO 7:

Solicite ao usuário seu ano de nascimento. Colete o ano atual e, com base nesta informação, informe a idade do usuário.

Variáveis

EXEMPLOS

EXEMPLO 7:


```
programa
{
    inclua biblioteca Calendario --> cal
    funcao inicio()
    {
        inteiro ano_nascimento, idade

        escreva("Informe o ano do seu nascimento: ")
        leia(ano_nascimento)

        idade = cal.ano_atual() - ano_nascimento

        escreva("Idade: ", idade)
    }
}
```

Variáveis

EXEMPLOS

EXEMPLO 8:

O programa deve ler um número inteiro e depois imprimir:

- O antecessor desse número;
- O sucessor desse número;
- O número elevado a 4ª potência;
- A raiz quadrada desse número (aproximadamente)

Variáveis

EXEMPLOS

EXEMPLO 8:

```
programa
{
	inclua biblioteca Matematica --> mat
	funcao inicio()
	{
		inteiro num
		leia(num)

		escreva("Antecessor: ", num - 1, "\n")
		escreva("Sucessor: ", num + 1, "\n")
		escreva("Potência: ", mat.potencia(num, 4), "\n")
		escreva("Raiz: ", mat.raiz(num, 2.0), "\n")
	}
}
```

Estruturas de Decisão

Estrutura Condicional

Estruturas de Decisão

Decisão Simples

Instrução de Decisão Simples

- Utiliza a seguinte sintaxe:

```
se(condição) {
    // código a ser executado
}
```

- A expressão da ***condição*** é avaliada;

- Se o resultado da avaliação é **`verdadeiro`**, os comandos dentro do bloco **indentado** são executados

Variáveis

EXEMPLOS

EXEMPLO 9.1:

Construa um programa que peça a entrada de dois números inteiros x e y.

- Se os números forem iguais, imprima: *"São iguais"*.
- Se x for maior que y, imprima: *"A entrada x é maior que entrada y"*.
- Se y for maior que x, imprima: *"A entrada x é menor que entrada y"*.

Variáveis

EXEMPLOS

EXEMPLO 9.1:

```
funcao inicio()
{
	inteiro x, y

	leia(x)
	leia(y)

	se(x == y){
		escreva("São iguais")
	}

	se(x > y){
		escreva(x, " é maior que ", y)
	}

	se(x < y){
		escreva(x, " é menor que ", y)
	}
}
```

Estruturas de Decisão

Decisão Composta

Instrução de Decisão Composta

Estrutura condicional “se-senao”:

- Após definir o parâmetro do “se” o que não ocorrer dentro desse parâmetro podemos colocar no “senao” e declarar novas instruções para esse parâmetro.

Sintaxe:

```
se(condição) {
    // Código a ser executado
    // se a condição for verdadeira
} senao {
    // Código a ser executado
    // se a condição NÃO verdadeira
}
```

Estruturas de Decisão

Decisão Composta

Instrução de Decisão Aninhada

Sintaxe:

```
se(condição) {
    // Código a ser executado
} senao se(condição){
    // Código a ser executado
} senao se(condição){
    // Código a ser executado
} senao se(condição){
    // Código a ser executado
} senao se(condição){
    // Código a ser executado
} senao{
    // Código a ser executado caso
    // nenhum outro caso tenha sido
    // atendido
}
```

Variáveis

EXEMPLOS

EXEMPLO 9.2:

Construa um programa que peça a entrada de dois números inteiros x e y.

- Se os números forem iguais, imprima: *"São iguais"*.
- Se x for maior que y, imprima: *"A entrada x é maior que entrada y"*.
- Se y for maior que x, imprima: *"A entrada x é menor que entrada y"*.

Variáveis

EXEMPLOS

EXEMPLO 9.2:

```
funcao inicio()
{
	inteiro x, y

	leia(x)
	leia(y)

	se(x > y) {
		escreva(x, " é maior que ", y)
	} senao se(x < y) {
		escreva(x, " é menor que ", y)
	} senao {
		escreva("São iguais")
	}
}
```

Estruturas de Decisão

escolha-caso

Instrução de Decisão Composta

Sintaxe:

```
escolha(variável) {
    caso valor1:
        // Código a ser executado
        pare
    caso valor2:
        // Código a ser executado
        pare
    caso valor3:
        // Código a ser executado
        pare
    caso contrario:
        // Código a ser executado
}
```

! A estrutura escolha é compatível apenas com os tipos inteiro e caracter!

escolha-caso

EXEMPLOS

EXEMPLO 10:

Escreva um programa que receba um operador de soma, subtração, multiplicação ou divisão, e dois números inteiros. Mostre o resultado da operação em seguida.

Exemplo de entrada	Exemplo de saída
+ 3 2	5
- 8 10	-2

escolha-caso

EXEMPLOS

EXEMPLO 10:

```
funcao inicio()
{
      caracter operador
      inteiro num1, num2
      leia(operador)
      leia(num1, num2)

      escolha(operador) {
            caso '+':
                  escreva(num1 + num2)
                  pare
            caso '-':
                  escreva(num1 - num2)
                  pare
            caso '*':
                  escreva(num1 * num2)
                  pare
            caso '/':
                  escreva(num1 / num2)
                  pare
}
```

Finished

Acabou \o/

C . E . S . A . R

Pessoas impulsionando inovação.

Inovação impulsionando negócios.

Tatyane Calixto
tscs@cesar.org.br

Erick Simões
esm@cesar.org.br

e a melhor equipe de monitores
da CESAR School 💟