INTRODUÇÃO À COMPUTAÇÃO

UD II - ALGORITMOS

ENTRADA E SAÍDA

Entrada e Saída

ELEMENTOS DE COMPETÊNCIA

- Empregar recursos para operar em ambientes humanizados, integrando as dimensões física, humana e informacional deste ambiente operacional.
- Tomar decisões e conduzir ações, em situações de crise.

Entrada e Saída

🎯 OBJETIVOS

1. Compreender a aplicação dos comandos e das funções de entrada e saída na lógica de programação. (CONCEITUAL/PROCEDIMENTAL)

Entrada e Saída

ATITUDES

1. Organização: capacidade de desenvolver atividades de forma sistemática e eficiente.
2. Dedicação: agir, realizando espontaneamente, com empenho e entusiasmo, as atividades necessárias ao cumprimento da missão.
3. Responsabilidade: capacidade de cumprir suas atribuições assumindo e enfrentando as consequências de suas atitudes e decisões.

Entrada e Saída

O ciclo básico de um algoritmo é formado por uma sequência lógica e finita de etapas cujo objetivo é transformar dados iniciais em um resultado desejado. Esse processo começa com a entrada de dados, passa pelo processamento — que pode envolver cálculos, decisões ou manipulações — e termina com a saída de informações que resolvem o problema proposto.

Comandos ou Funções de Entrada

O comando de entrada de dados permite a inserção, via teclado, de dados que serão utilizados pelo algoritmo. Os valores inseridos, normalmente, são armazenados em variáveis, que serão manipuladas pelo programa durante a sua execução. Em pseudocódigo, o comando Leia executa esta função.

Comandos ou Funções de Entrada

Este comando pode ser utilizado para a leitura de uma variável:

Leia(<variável>)

Exemplo:

Leia(idade)

Ou, na leitura de várias variáveis em sequência. Exemplo:

Leia(<variável$_1$>,<variável$_2$>)

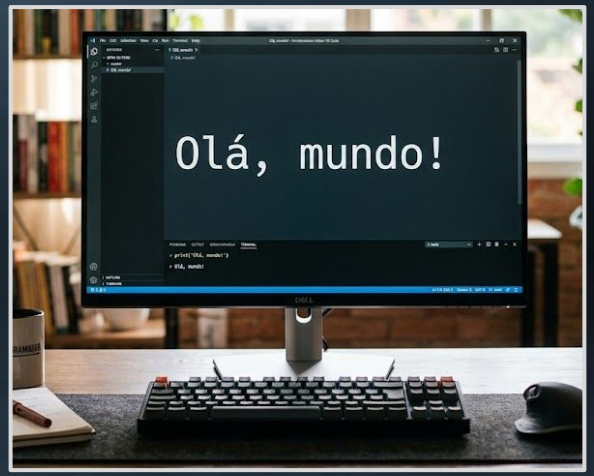
Exemplo:

Leia(nome, salario)

Comandos ou Funções de Saída

Mostram ao usuário os resultados do processamento dos algoritmos, bem como qualquer outro tipo de mensagem que se fizerem necessárias durante a execução do programa. Esses valores e mensagens são mostradas no monitor ou algum dispositivo de saída.

Em pseudocódigo o comando de saída é o Escreva.



Comandos ou Funções de Saída

Os comandos de saída exibem na tela os argumentos fornecidos a essas funções, e, quando fornecidos vários argumentos estes devem ser separados por vírgula. Os argumentos poderão ser:

- variável: o valor contido nesta variável é exibido.
- constante: o valor da constante é exibido.
- expressão: é resolvida e o seu resultado final é exibido.
- texto: a mensagem, entre as aspas que delimitam o texto, será exibida.

Comandos ou Funções de Saída

A sintaxe básica, para um parâmetro é: **Escreva(<argumento>)**

Escreva(idade) → 10

Escreva(PI) → 3.1415926

Escreva(“Olá, mundo!”) → Olá, mundo!

Escreva(x+10^2) → 108

Considere a idade = 10 e x = 8

Comandos ou Funções de Saída

Para a utilização de vários parâmetros:

Escreva(<argumento$_1$> , <argumento$_2$> , ..., <argumento$_n$>)

Considere a idade = 10; nome="Pedro"; lado1 = 8; lado2 = 7 e lado3 = 5

Escreva("A idade é:", idade) → 10

Escreva("Perímetro triângulo:", lado1+lado2+lado3, " cm") → Perímetro triângulo: 30 cm

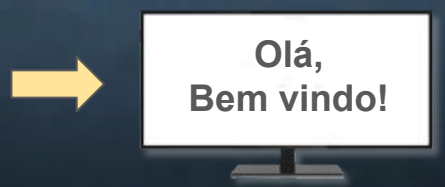
Escreva("Aluno ", nome) → Aluno Pedro

Escreva x escreval

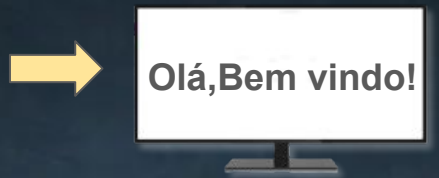
O comando Escreva mantém o cursor na mesma linha. Isso significa que o próximo comando de saída continuará escrevendo exatamente onde o anterior parou. O uso comum é para criar mensagens compostas ou solicitar dados onde a resposta do usuário deve ficar na frente da pergunta.

Há uma alternativa ao escreva que é o comando Escreval (“Escreva linha”), que mostra o conteúdo e, ao final, salta o cursor para o início da próxima linha.

```
Escreval( “Olá,” )
Escreval( “Bem vindo!” )
```

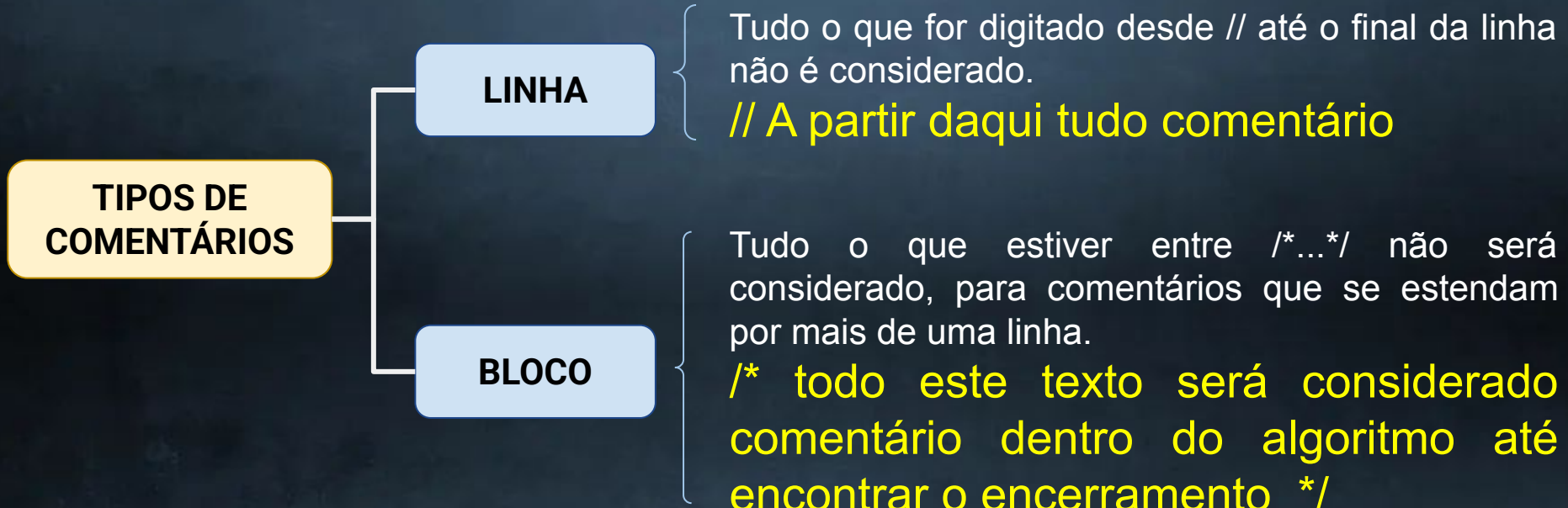



```
Escreva( “Olá,” )
Escreva( “Bem vindo!” )
```



Comentários

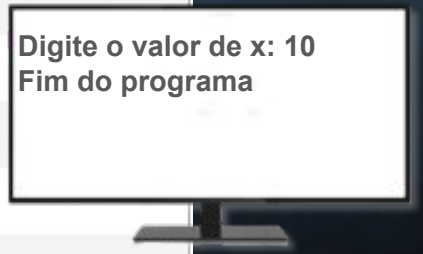
São informações inseridas nos algoritmos que servirão apenas como explicações e orientações sobre o algoritmo ou parte dele. O conteúdo desses comentários não é desconsiderado no momento da execução do algoritmo, isto é, o texto que o compõem não será exibido ou terá qualquer efeito sobre a execução do código.



Comentários

As linguagens de programação e pseudocódigos oferecem os dois tipos de comentário. No software Visualg somente comentários de linha.

```
Algoritmo "05_01 – exemplo de uso de comentário"
// Este é um primeiro exemplo de uso de comentários. No visualg, comentários
// em bloco exigem o símbolo de dupla barra no início em cada linha
Var
   x: inteiro
Inicio
   Escreva("Digite o valor de x:")
   Leia(x)          // realiza a leitura de um valor e armazena na variável "x"
   Escreva ("Fim do programa")
FimAlgoritmo
```



Estrutura básica de um algoritmo

Já se tem estruturas básicas para se escrever algoritmos simples.

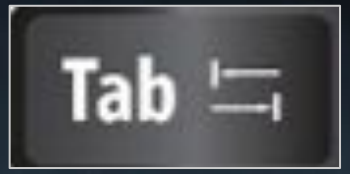
Porém, devemos observar que há uma estrutura básica que todo algoritmo deve ter:

```
Algoritmo <nome_algoritmo>
// <comentário>
Var <variável1>,<variável2>:<tipo de variável>
    <variável3>:<tipo de variável>
Início
    <comando1>
    <comando2>
    (...)
    <comando_n>
FimAlgoritmo
```

Código estruturado: indentação

Um código-fonte, seja em pseudocódigo ou em uma linguagem de programação, exige mais do que apenas a sintaxe correta; ele necessita ser bem estruturado, com seus blocos lógicos claramente delimitados.

Um recurso é a utilização da Indentação, um recuo do texto em relação à margem. Esse recurso cria uma hierarquia visual que organiza parágrafos (em textos comuns) ou blocos lógicos (em programação), melhorando drasticamente a legibilidade. O recuo pode ser feito usando espaços ou tabulação (tecla “Tab”), que insere automaticamente um espaçamento padrão.



Código estruturado: indentação

Código estruturado

```
Se x>1 então
|     Escreva("Número maior que zero")
Senão
|     Se x=0 então
|     |     Escreva("Número igual a zero")
|     Senão
|     |     Escreva("Número menor que zero")
|     Fimse
Fimse
```

Tab Tab Tab

Código não estruturado

```
Se x>1 então
Escreva("Número maior que zero")
Senão
Se x=0 então
Escreva("Número igual a zero")
Senão
Escreva("Número menor que zero")
Fimse
Fimse
```

Apresentação de um algoritmo completo


```
Algoritmo "05_02 Conversor_segundos"
// Esse algoritmo converte um valor total de segundos em horas, minutos e segundos.
Var
   total_segs, horas, minutos, segundos_restantes, segundos_restantes_final: inteiro
Inicio
   // Entrada de dados
   Escreva ("Digite a quantidade de segundos: ")
   Leia (total_segs)

   // Processamento
   horas ← total_segs \ 3600
   segundos_restantes ← total_segs % 3600
   minutos ← segundos_restantes \ 60
   segundos_restantes_final ← segundos_restantes % 60

   // Saída de dados
   Escreva (horas, "hora(s)," ,minutos, "minuto(s) e" ,segundos_restantes_final, "segundo(s).")
FimAlgoritmo
```

COMENTÁRIOS

DEFININDO VARIÁVEIS

ENTRADA DE DADOS

PROCESSAMENTO

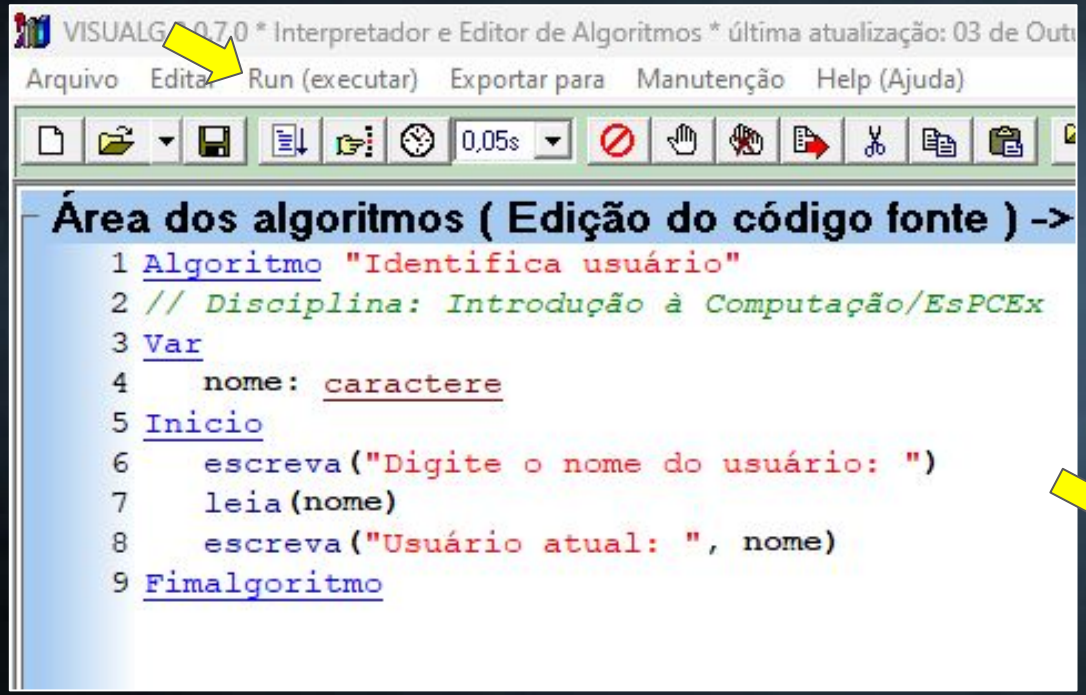
SAÍDA

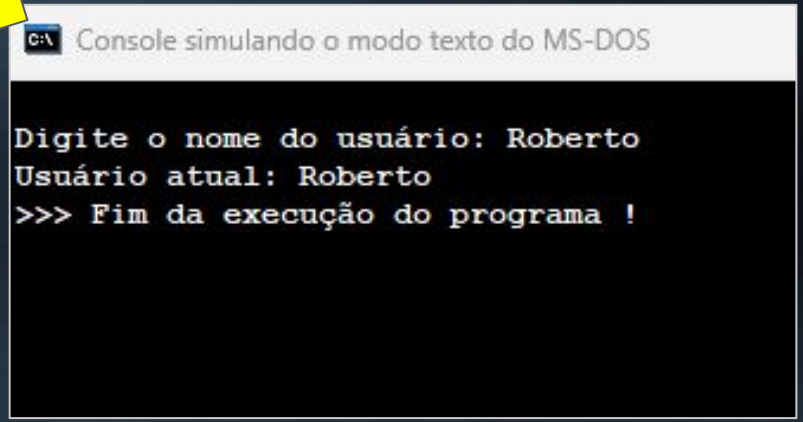
Exercícios em sala (1)

Desenvolva um algoritmo que leia o nome de um usuário e apresente no monitor a seguinte mensagem: “Usuário atual: Fulano”.

```
Algoritmo "Identifica usuário"
// Disciplina: IC/EsPCEx
Var
   nome: caractere
Inicio
   escreva("Digite o nome do usuário: ")
   leia(nome)
   escreva("Usuário atual: ", nome)
Fimalgoritmo
```

Exercícios em sala (1) usando Visualg





Exercícios em sala (2)

Desenvolva um algoritmo que leia o efetivo de um pelotão e a quantidade de munição por militar, apresentando a quantidade total de munição necessária para uma determinada instrução.

Exercícios em sala (2)

```
Algoritmo "Quantidade de munição"
// Exercício 2 em sala - IC
Var
   efetivo, munPerCapita: inteiro
Inicio
   escreva("Efetivo do pelotão: ")
   leia(efetivo)
   escreva("Munição por militar: ")
   leia(munPerCapita)
   escreva(“Munição necessária: ”, efetivo * munPerCapita)
Fimalgoritmo
```

Exercícios em sala (2) usando Visualg

VISUALG 3.0.7.0 * Interpretador e Editor de Algoritmos * última atualização: 03 de Outubro de 2015 * Escola

Arquivo Edit... un (executar) Exportar para Manutenção Help (Ajuda)

0,05s

Área dos algoritmos (Edição do código fonte) -> Nome do a

```
1  Algoritmo "Quantidade de munição"
2  // Exercício 2 em sala - IC
3  Var
4     efetivo, munPerCapita: inteiro
5  Inicio
6     escreva("Efetivo do pelotão: ")
7     leia(efetivo)
8     escreva("Munição por militar: ")
9     leia(munPerCapita)
10    escreva("Munição necessária: ", efetivo * munPerCapita)
11 Fimalgoritmo
12
```

Console simulando o modo texto do MS-DOS

```
Efetivo do pelotão: 30
Munição por militar: 15
Munição necessária:  450
>>> Fim da execução do programa !
```