Universidade Positivo

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

Aula: ALP02 - Tipos de dados. Identificadores. Constantes e variáveis.

Professor: Fabricio Olivo
fabricio@up.edu.br

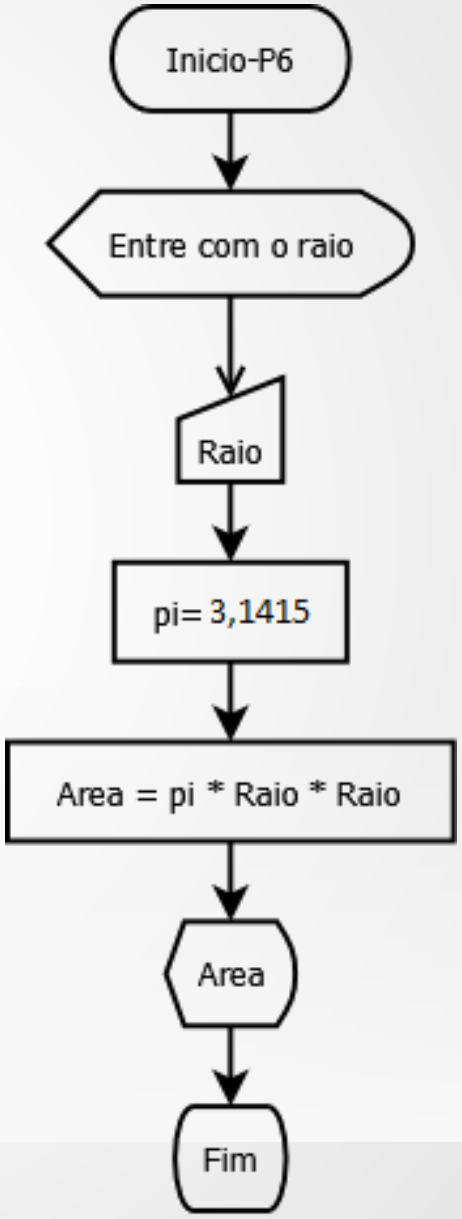
ATIVIDADE DA AULA ANTERIOR

ATIVIDADE 1: FLUXOGRAMA ÁREA DA CIRCUNFERÊNCIA

- Escrever um **algoritmo** do tipo **fluxograma** que **leia** o valor do raio de uma circunferência, **calcule** sua área pela equação,

  (**area = π * r^2** ⇔ **area = PI * raio * raio**),

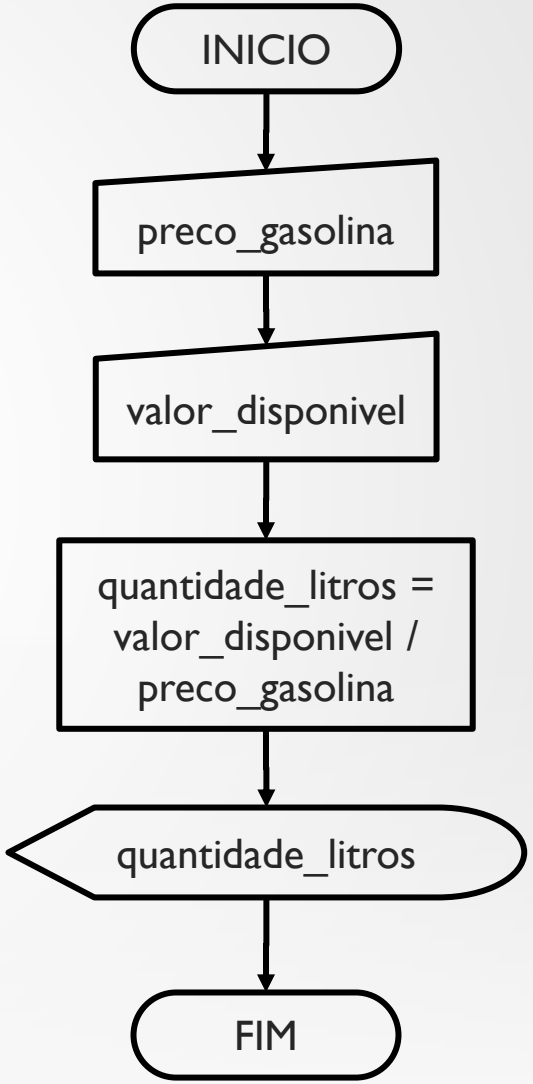
- e **mostre** o resultado na tela.



ATIVIDADE DA AULA ANTERIOR

ATIVIDADE 2: FLUXOGRAMA LITROS COMBUSTÍVEL

- Escrever um **algoritmo** do tipo **fluxograma** que **leia** o preço do litro da gasolina e o valor disponível para abastecimento, **calcule** a quantidade de litros que irá para o tanque, e **mostre** o resultado na tela.



ALGORITMO

É uma

sequência finita e suficientemente precisa de instruções

que

detalha o processo de solução de um problema

e que,

quando executada por uma outra pessoa

ou

por uma máquina

a partir dos mesmos dados de entrada,

leva à solução do problema.

PROPRIEDADES DE ALGORITMOS

Finitude	Um algoritmo deve ter um número finito de passos
Exatidão	As etapas que compõem um algoritmo devem ser claramente definidas e ordenadas, sem margem para interpretações ambíguas
Entradas e saídas determinadas	Todas as entradas e saídas devem ser explicitadas. Um algoritmo pode ter zero ou mais entradas, mas deve ter ao menos uma saída
Efetividade	O algoritmo deve solucionar os problemas a que se propõe
Eficiência	Deve-se sempre buscar a melhor combinação de três fatores: tempo, esforço e recursos necessários

PARTES DE UM ALGORITMO

- **Entrada**: são fornecidas as informações necessárias para que o algoritmo possa ser executado.
- **Processamento**: são executadas todas as instruções, avaliadas todas as expressões algébricas, relacionais e lógicas e todas as estruturas de controle existentes no algoritmo.
- **Saída**: os resultados do processamento são enviados para um ou mais dispositivos de saída, como: monitor, impressora ou a própria memória do computador.

PROGRAMAÇÃO DE COMPUTADORES

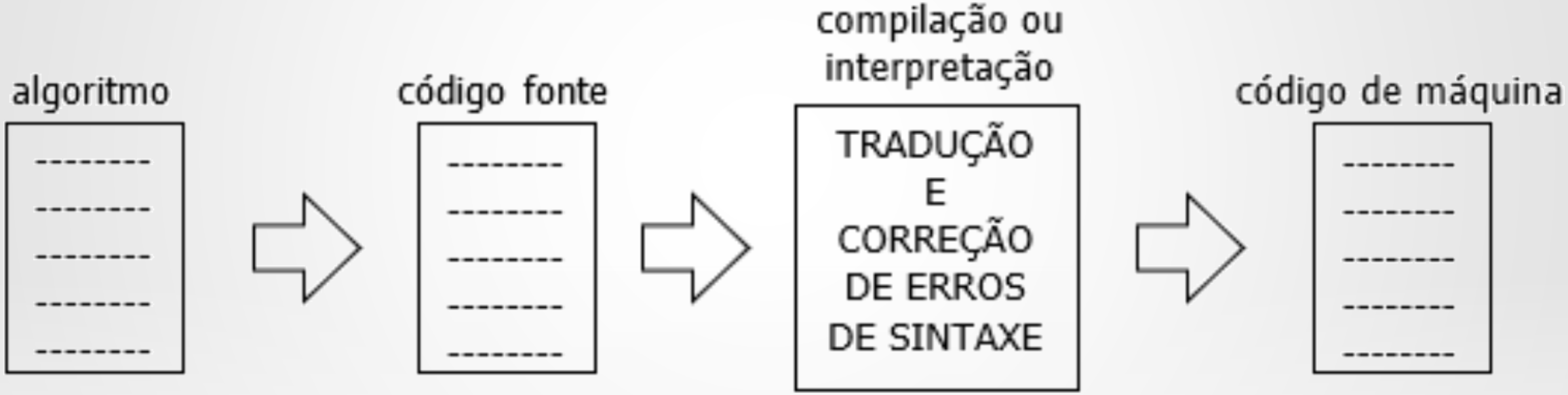
- Exige do programador um **raciocínio**
  - coerente,
  - organizado,
  - lógico e
  - criativo.
- Exige do programador o **exercício** de
  - abstração,
  - modelagem,
  - sequenciamento e
  - modularização do problema a ser solucionado.
- Exige do programador MUITO trabalho prático!
- Por isso, ...

... SÓ SE APRENDE PROGRAMAR,

PROGRAMANDO!

PROGRAMAÇÃO DE COMPUTADORES

- O processo de construção de um programa é similar para qualquer linguagem de programação.
- Em geral, o ciclo de construção de uma aplicação engloba quatro etapas:




```
ALGORITMO ola_mundo
INICIO
    ESCREVA("Ola mundo!");
FIMALGORITMO
```

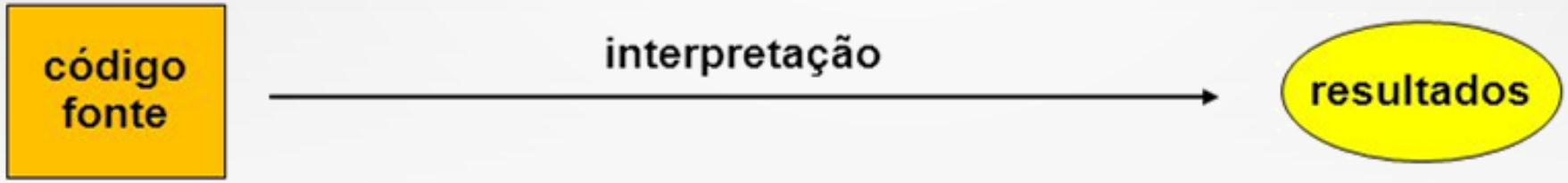
```
int main(){
    printf("Ola mundo!");
}
```

```
00011101010101010101101010
10101010110101010101010101
11110101010101010101101101
01010110101010101010101010
10101010111011010001110011
01010101101011111010100000
```

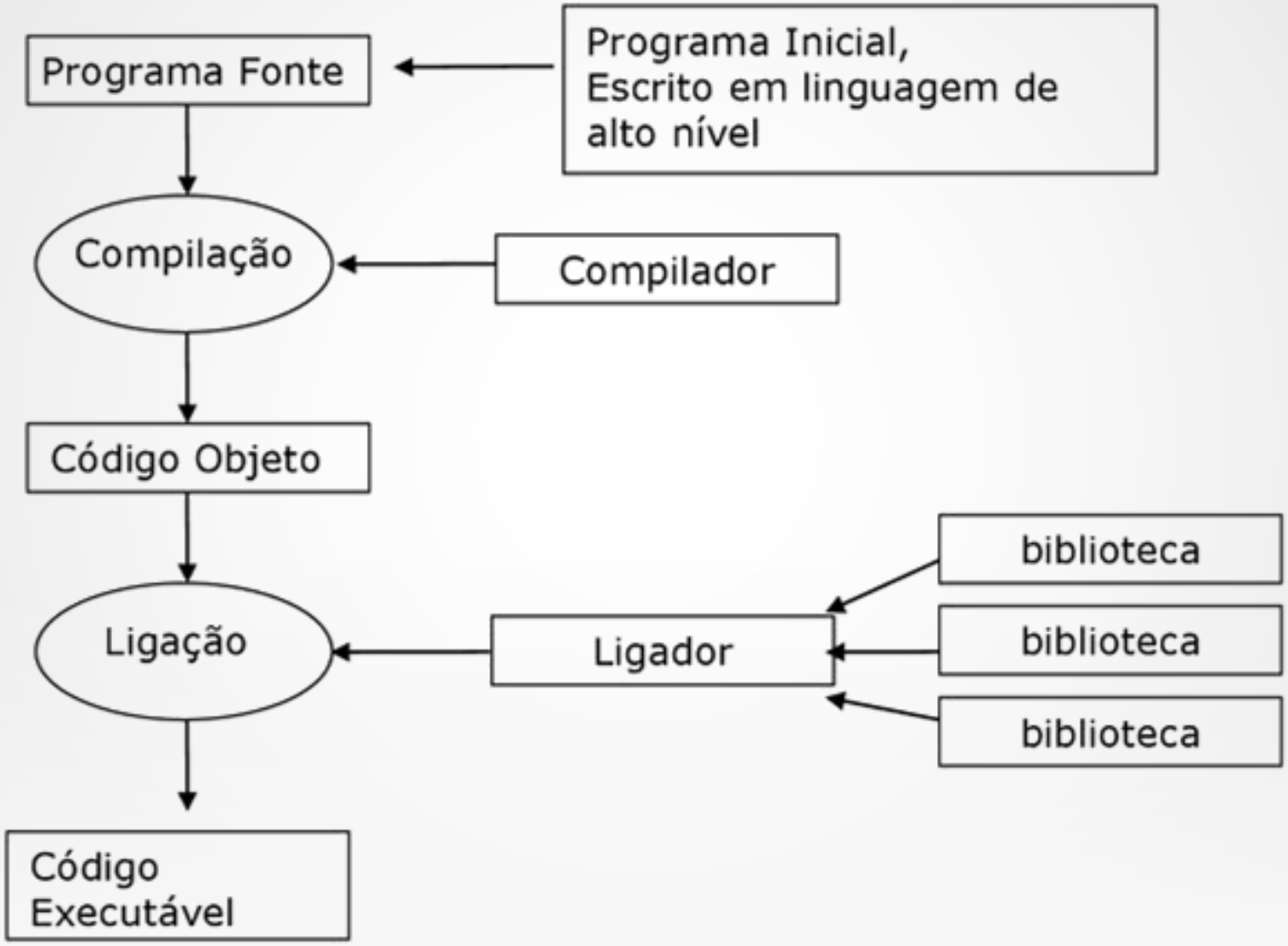
PROGRAMAÇÃO DE COMPUTADORES

- Compilação

- Interpretação



PROGRAMAÇÃO DE COMPUTADORES



PROGRAMAÇÃO DE COMPUTADORES

ALGORITMO -> CÓDIGO FONTE -> COMPILAÇÃO OU INTERPRETAÇÃO -> CÓDIGO DE MÁQUINA

- Primeira forma de representação da solução de um problema.
- Uso da lógica de programação.
- Sequência de passos coerentes e organizados que visa atingir um objetivo bem definido.
- Estrutura sequencial (começo-meio-fim).

```
ALGORITMO boas_vindas
VAR
    CARACTER: nome;
INICIO
    ESCREVA ("Digite seu nome: ");
    LEIA(nome);
    ESCREVA ("Ola, Bem Vindo ", nome);
FIMALGORITMO
```

PROGRAMAÇÃO DE COMPUTADORES

ALGORITMO -> **CÓDIGO FONTE** -> COMPILAÇÃO OU INTERPRETAÇÃO -> CÓDIGO DE MÁQUINA

- Tradução de um algoritmo para uma linguagem de programação específica.
- Precisa ser traduzido para linguagem de máquina para que possa ser executado.
- Linguagem de programação:
  - Conjunto de convenções e regras que especificam como transmitir informações entre pessoas e máquinas.
  - De forma simples, é composta por dois elementos: um conjunto de símbolos (vocabulário) e um conjunto de regras (gramática) para utilizá-la.
  - Possibilita que o hardware (parte física do computador) se comunique com a parte lógica do computador a partir dos programas ou aplicações.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char *argv[])
{
    int num, sr, flag, i;

    if (argc != 2) return 1;
    num = atoi(argv[1]);
    sr = (int)sqrt(num);
    if (num < 2)
        flag = 0;
    else
    {
        flag = 1;
        for (i=2; i<=sr; i++)
            if (num%i == 0)
            {
                flag = 0;
                break;
            }
    }
    if (flag) printf("%d e' primo\n", num);
    else printf("%d nao e' primo\n", num);
    return 0;
}
```

PROGRAMAÇÃO DE COMPUTADORES

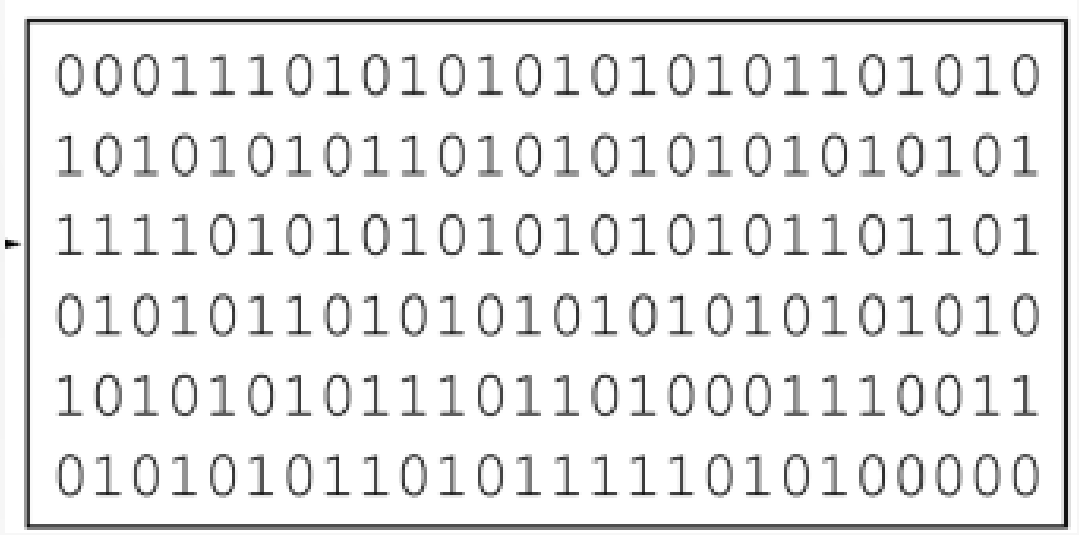
ALGORITMO -> CÓDIGO FONTE -> **COMPILAÇÃO OU INTERPRETAÇÃO** -> CÓDIGO DE MÁQUINA

- No processo de **compilação**, acontecem as **análises sintáticas do código** escrito pelo programador para gerar um **arquivo objeto**.
- No processo de **interpretação**, nenhum arquivo ou código é gerado, e sim uma **tradução instantânea** em tempo de execução do código escrito pelo programador.

PROGRAMAÇÃO DE COMPUTADORES

ALGORITMO -> CÓDIGO FONTE -> COMPILAÇÃO OU INTERPRETAÇÃO -> **CÓDIGO DE MÁQUINA**

- O **código de máquina** (também chamado de linguagem de máquina), popularmente conhecido como "**zeros e uns**", são as **instruções** que o **processador interpreta** e **executa**.
- São basicamente **números** que o **processador** do computador **decodifica** para **executar** as **operações** identificadas nas **instruções** escritas pelo **programador**.



TIPOS DE DADOS

Os dados são as informações a serem processadas por um computador.

INTEIROS: Representam dados numéricos positivos ou negativos sem parte fracionária.
Exemplos: 5, 0, -56.

REAIS: Representam dados numéricos positivos ou negativos com parte fracionária.
Exemplos: 5.9, 0.5, -30.7.

CARACTERE: Representam sequências de letras, números e símbolos especiais.
Deve ser indicada entre aspas duplas.
Também conhecido como alfanumérico, *string*, literal ou cadeia de caracteres.
Exemplos: "Ana Luísa", "CD98A", "Fone: 5555-6666", " ", "ab*10".

LÓGICO: Representam os valores VERDADEIRO e FALSO.
Esse tipo de dados também é conhecido como tipo **BOOLEANO**.

CARACTERÍSTICAS DE ARMAZENAMENTO

VARIÁVEIS

- Todo **dado** a ser **armazenado** na **memória** de um computador deve ser previamente **vinculado a um tipo (“tipado”)**, ou seja, primeiro é necessário saber qual o seu tipo para depois fazer o seu armazenamento adequado.
- Quando declaramos uma **variável** em um programa, estamos **definindo e reservando um espaço na memória** para **armazenar o valor** que aquela variável conterá em determinado tempo de execução do programa.
- Chama-se **variável** pois o valor contido nesse espaço de memória do computador **poderá variar** com o tempo, não sendo um valor fixo.

CARACTERÍSTICAS DE ARMAZENAMENTO

VARIÁVEIS

- Como a memória armazena inúmeros dados, cada um deve ser **identificado** com um **nome** (**identificador**).
- O **nome** de uma variável **identifica** uma região específica da memória onde um dado está armazenado.
- Esse nome:

1. Deve iniciar sempre com letra. 2. Deve ser formado apenas por letras, números ou *underline* (_). 3. Deve ser único. 4. Deve ser significativo.	5. Não pode conter espaços em branco. 6. Não pode ser o nome de uma palavra reservada de uma linguagem. 7. Não deve ser acentuado ou ter “ç”.

CARACTERÍSTICAS DE ARMAZENAMENTO

VARIÁVEIS

- Na declaração de variáveis, deve-se informar o **tipo** a ser usado e o **nome** da variável.
- Se mais de uma variável for declarada em uma linha, separa-las por vírgula “ **,** ”.
- Exemplos:

```
VAR
    INTEIRO: quantidade;
    CARACTERE: nome_cliente, data_compra;
    REAL: preco_produto, valor_dolar;
    LOGICO: resposta;
```

CARACTERÍSTICAS DE ARMAZENAMENTO

CONSTANTES

- Representam um dado que não pode ser alterado, que é **fixo** e **estável**, por exemplo, o valor do pi (3.1416).
- As constantes são representadas em algoritmos por números ou textos fixos.
- Exemplos:

```
CONST
    INTEIRO: QUANTIDADE;
    CARACTERE: NOME_CLIENTE, DATA_COMPRA;
    REAL: PI, PRECO_PRODUTO, VALOR_DOLAR;
    LOGICO: RESPOSTA;
```

PRÁTICA

1. Relacionar o tipo de cada um dos identificadores a seguir:
   a) 12
   b) “AMOR”
   c) “falso”
   d) 3,81 × 102
   e) Verdadeiro

2. Marque os identificadores válidos:
   a) ( ) soma
   b) ( ) X”y
   c) ( ) 3N
   d) ( ) N3
   e) ( ) temperatura ambiente

PALAVRAS RESERVADAS

- São identificadores que têm um significado especial na linguagem utilizada.
- Representam comandos e operadores.
- Também podem representar subprogramas já embutidos na linguagem.
- Não podem ser utilizadas como identificadores definidos pelo programador.
- Exemplos em pseudocódigo:
  - ALGORITMO, INICIO, FIM, SE, SENAO, ESCREVA, LEIA, FUNCAO, REPITA ...

PALAVRAS RESERVADAS

Palavras Reservadas

ALEATÓRIO	ALGORITMO	ARQUIVO
ASC	ATE	CARAC
CARACPNUM	CARACTERE	CASO
COMPR	COPIA	CRONOMETRO
DEBUG	E	ECO
ENQUANTO	ENTAO	ESCOLHA
ESCREVA	ESCREVAL	FACA
FALSO	FIMALGORITMO	FIMENQUANTO
FIMESCOLHA	FIMFUNCAO	FIMPARA
FIMPROCEDIMENTO	FIMREPITA	FIMSE
FUNCAO	INICIO	INT
INTEIRO	INTERROMPA	LEIA
LIMPATELA	LOGICO	MAIUSC
MINUSC	MOD	NAO
NUMPCARAC	OU	OUTROCASO
PARA	PASSO	PAUSA
POS	PROCEDIMENTO	REAL
REPITA	RETORNE	SE
SENAO	TIMER	VAR
VERDADEIRO	VETOR	XOU

COMENTÁRIOS

- Recursos que permitem a inclusão de esclarecimentos sobre o que o programa faz em determinado ponto do código ou seu objetivo geral.
- Todo o conteúdo dentro de um comentário é ignorado durante a execução de um programa.
- São delimitados por símbolos especiais que podem variar dependendo da linguagem.
- Podem compreender quaisquer sequências de caracteres.
- Em pseudocódigo, os comentários são delimitados pelos símbolos "/*" e "*/", ou "//".
- Ex.: /* isto é um comentário que pode abranger várias linhas */
  // isto é um comentário de uma única linha.

```
ALGORITMO nome_do_algoritmo
VAR
  /* declaração de variáveis */
INICIO
  /* inicialização de variáveis */
  /* desenvolvimento (fórmulas, estruturas de decisão ou repetição, ...) */
FIMALGORITMO
```

BOAS PRÁTICAS

- Evitar utilizar nomes genéricos demais para variáveis e constantes.
- Deve ficar claro o tipo de informação a que elas se referem.
  - Ex.: utilizar “altura” e “peso” em vez de “valorA” e “valorB”
- Utilizar letras **minúsculas** para **nomes** de **variáveis**.
  - Ex.: “largura”, “capacidade”, “identificador”, “ano”, “modelo”
- Utilizar letras **maiúsculas** para **nomes** de **constantes**.
  - Ex.: “PI”, “RG”, “TEMPO_DURACAO”, “ALTURA_MAXIMA”
- Caso uma variável tenha nome composto, utilizar “_” ou usar letra maiúscula somente na primeira letra da segunda palavra.
  - Ex.: “nome_sobrenome”, “nomeSobrenome”, “cidade_nascimento”, “cidadeNascimento”.

BOAS PRÁTICAS

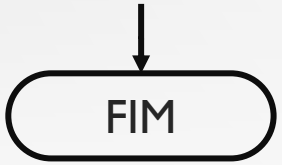
- Evitar utilizar nomes muito grandes para variáveis e constantes.
  - Ex.: em vez de “altura_maxima_permitida_veiculo”, usar “altura_maxima” ou “alturaMaxima”.
- Utilizar comentários para lembrar o que está sendo feito em cada ponto do programa.
- Sempre **indentar** o código!
  - Recuar o texto em relação à margem da folha.

```
ALGORITMO boas_vindas
VAR
    CARACTER: nome;
INICIO
    ESCREVA ("Digite seu nome: ");
    LEIA(nome);
    ESCREVA ("Ola, Bem Vindo ", nome);
FIMALGORITMO
```

ATIVIDADE DA AULA

ATIVIDADE 1: FLUXOGRAMA SOMA

- Escreva um **algoritmo** do tipo **Fluxograma** para **ler** dois números inteiros, **calcular** a soma dos números, e **imprimir** o resultado na tela.



ATIVIDADE DA AULA

ATIVIDADE 2: PSEUDOCÓDIGO PORTUGOL SOMA

- Escreva um **algoritmo** do tipo **pseudocódigo Portugol** para **ler** dois números inteiros, **calcular** a soma dos números, e **imprimir** o resultado na tela.

```
ALGORITMO soma
VAR

INICIO




FIMALGORITMO
```

ATIVIDADE DA AULA

ATIVIDADE 3: PSEUDOCÓDIGO PORTUGOL MÉDIA ARITMÉTICA

- Escreva um algoritmo do tipo pseudocódigo que **leia** quatro notas de um aluno, **calcule** a **média aritmética** dessas notas, e **apresente** na **tela** o **resultado**.

```
ALGORITMO media_aritmetica
VAR



INICIO









FIMALGORITMO
```

ATIVIDADE DA AULA

1. Escreva um **algoritmo** em **pseudocódigo** que **leia** um número inteiro **digitado** pelo usuário e **calcule** e **imprima** seu antecessor e seu sucessor.
   Por exemplo, suponha que o usuário digite o número 10, o algoritmo deverá imprimir na tela o os números 9 e 11.
2. Entregar as atividades no link do ambiente virtual observando o prazo.

BIBLIOGRAFIA BÁSICA

- MEDINA, Marco; FERTIG, Cristina. **Algoritmos e programação: teoria e prática**. 2. ed. São Paulo: Novatec, 2006. 384 p. ISBN 857522073X (broch.).
- FORBELLONE, A. L. V. **Lógica de programação: a construção de algoritmos e estruturas de dados**. 3. ed. São Paulo: Pearson Prentice Hall, 2005.
- MANZANO, José Augusto Navarro Garcia; OLIVEIRA, Jayr Figueiredo de. **Algoritmos: lógica para desenvolvimento de programação de computadores.** 22. ed. São Paulo: Ed. Érica, 2009.

BIBLIOGRAFIA COMPLEMENTAR

- EDELWEISS, Nina. **Algoritmos e programação com exemplos em Pascal e C**. Porto Alegre Bookman 2014 1 recurso on-line (Livros didáticos UFRGS 23). ISBN 9788582601907.
- ZIVIANI, Nivio. **Projeto de algoritmos: com implementações em Pascal e C**. 3. ed. rev. e ampl. São Paulo: Cengage Learning, 2012. xx, 639 p. ISBN 9788522110506 (broch.).
- MILETTO, Evandro Manara; OKUYAMA, Fabio Yoshimitsu; NICOLAO, Mariano (Org.). **Desenvolvimento de software I: con-ceitos básicos**. 1. ed. Porto Alegre: Bookman, 2014. 223 p. ISBN 9788582601457.
- ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi. **Fundamentos da programação de computadores: algoritmos, Pascal, C/C++ (padrão ANSI) e Java**. 3. ed. São Paulo: Pearson Education do Brasil, c2012. x, [4], 4569 p. ISBN 9788564574168 (broch.).
- SOUZA, Marco Antônio F. de; GOMES, Marcelo Marques; SOARES, Marcio Vieira; CONCILIO, Ricardo. **Algoritmos e lógica de programação: um texto introdutório para engenharia**. 2. ed. São Paulo: Cengage Learning, 2013. 234 p. ISBN 9788522111294.