



ALGORITMOS E LÓGICA DE PROGRAMAÇÃO

~ LINGUAGEM C ~

Material de Apoio

Prof^a Andréia Miranda Domingues



SUMÁRIO

PROCESSAMENTO DE DADOS.....	6
ENTRADA.....	6
PROCESSAMENTO	6
SAÍDA	6
ORGANIZAÇÃO BÁSICA DE UM COMPUTADOR	8
Unidades de Entrada e de Saída.....	8
Unidade de Memória	8
Unidade Lógica e Aritmética.....	8
Unidade de Controle	8
INSTRUÇÕES PRIMITIVAS - PROCESSAMENTO DE DADOS.....	8
ENTRADA.....	8
SAÍDA	8
FORMAS DE REPRESENTAÇÃO DOS ALGORITMOS	9
DESCRIÇÃO NARRATIVA.....	9
PSEUDOCÓDIGO OU PORTUGUÊS ESTRUTURADO	11
FLUXOGRAMA OU DIAGRAMA DE BLOCOS	13
TESTE DE MESA.....	16
OPERADORES	18
OPERADORES ARITMÉTICOS.....	18
OPERADORES RELACIONAIS	18
OPERADORES LÓGICOS.....	18
CÁLCULOS DE PORCENTAGEM	20
LINGUAGEM C	23
VARIAÇÕES DA LINGUAGEM C.....	23
CARACTERÍSTICAS DA LINGUAGEM C	23
CONSTANTES.....	24

VARIÁVEIS	24
TIPOS DE VARIÁVEIS.....	24
PALAVRAS RESERVADAS	25
FUNÇÕES DA LINGUAGEM C	25
FUNÇÃO main()	25
FUNÇÃO printf()	25
CARACTERES ESPECIAIS.....	25
CARACTERES de FORMATAÇÃO	26
FUNÇÃO scanf()	26
MINI TUTORIAL DEV C++ ou DEV CPP (DEV C PLUS PLUS)	28
COMENTÁRIOS EM LINHAS DE CÓDIGO	28
ESTRUTURAS DE CONTROLE – Tomadas de Decisão - Desvio Condicional	31
DESVIO CONDICIONAL SIMPLES.....	31
DESVIO CONDICIONAL COMPOSTO	32
DESVIO CONDICIONAL ENCADEADO ou ANINHADO	33
OPERADORES LÓGICOS AND, OR e NOT	36
INCREMENTO	39
INCREMENTO PRÉ-FIXADO	39
INCREMENTO PÓS-FIXADO	39
DECREMENTO	39
OPERADORES DIV e MOD.....	40
OPERADOR DIV (Quociente da Divisão)	40
OPERADOR MOD (Resto da Divisão)	40
FUNÇÕES MATEMÁTICAS	41
FUNÇÃO POW() - <u>POTENCIAÇÃO</u>	41
FUNÇÃO SQRT() - <u>RADICIAÇÃO</u>	41
ESTRUTURA CONTROLE - LAÇOS DE REPETIÇÃO – TESTE NO INICIO ..	42

CONCEITOS – String e Formatação.....	47
Função gets() – Digitação de textos (strings)	47
ESTRUTURA CONTROLE - LAÇOS DE REPETIÇÃO – TESTE NO FINAL...	48
ESTRUTURA DE REPETIÇÃO	50
VETORES	53
ESTRUTURA DE CONTROLE – DECISÃO - MÚLTIPLA ESCOLHA.....	62

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 1

Apresentação dos tópicos da disciplina explicando-os com aplicações do cotidiano para que o aluno possa visualizar o que ele será capaz de desenvolver ao finalizar o semestre.

BIBLIOGRAFIA BÁSICA

MANZANO, José Augusto Navarro Garcia & **OLIVEIRA**, Jair Figueiredo de. Algoritmos: Lógica para o desenvolvimento de programação. São Paulo: Ed. Érica, 2004.

MIZRAHI, Victorine Viviane. Treinamento em Linguagem C – Módulo I. São Paulo: Ed. Makron Books, 1990.

<p>Introdução a Representação de Algoritmos com Descrição Narrativa abordando o tema: Troca de uma caixa d'água</p>
--

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 2

Os problemas computacionais podem ser solucionados seguindo-se um ou mais caminhos. Para isso, utiliza-se a lógica de programação com instruções que visam a solução de um dado problema.

A lógica de programação é representada por algoritmos. Esses fazem parte também da nossa vida diária e constantemente fazemos uso deles. Exemplos:

- Indicações são dadas para se chegar até um endereço específico
- Plantas são usadas para a construção de prédios
- Orientações são seguidas para a montagem de um aparelho

PROCESSAMENTO DE DADOS

Ao desenvolver um algoritmo é preciso entender como os dados são processados.

ENTRADA

Os dados de entrada são geralmente fornecidos pelo usuário através do teclado, mas podem também ser oriundos de outro sistema. A seguir, eles são enviados para armazenamento na memória do computador em locais denominados variáveis.

PROCESSAMENTO

Trata-se dos procedimentos necessários para se chegar ao resultado final. Muitas vezes são cálculos com variáveis e ocorrem na memória na Unidade Lógica e Aritmética (ULA).

SAÍDA

Após o processamento, os dados (respostas esperadas) são apresentados na tela ou enviados para a impressora.

INFORMATIVO → DESENVOLVIMENTO DE SOFTWARES

De modo geral, os projetos podem ser classificados como:

Bem sucedidos – O projeto é finalizado no prazo, dentro do orçamento e contendo todas as funcionalidades especificadas.

Comprometidos – O projeto é finalizado e um software operacional é entregue, porém o orçamento e o prazo ultrapassam os limites estipulados, e, além disso, o software entregue possui menos funcionalidades do que o especificado.

Fracassados – O projeto é cancelado em algum momento durante o desenvolvimento.

Entretanto, a empresa *Standish Group International*, que apresenta **Resultados do Desenvolvimento de Projetos**, mostra que somente 1/3 deles se enquadram na categoria **Bem Sucedidos**.

Desse modo, é primordial considerarmos a importância de se desenvolver programas com dados bem estruturados.

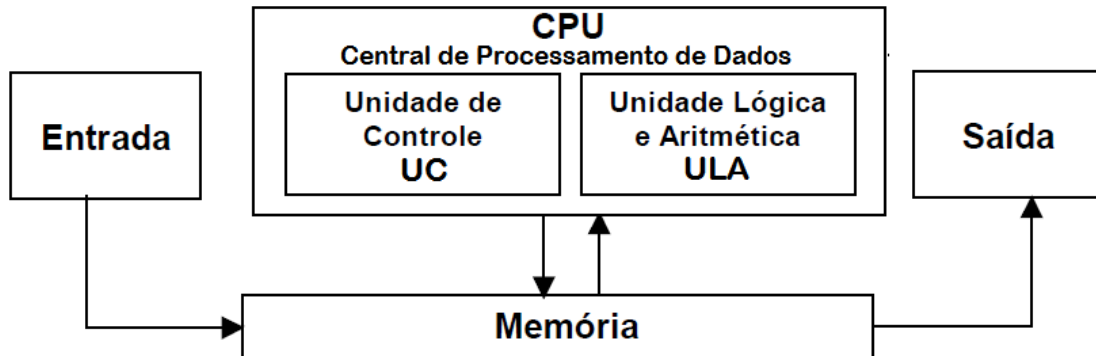
RESUMO

- **Lógica:** é a forma de organizar os objetivos raciocinando de maneira coerente sobre a solução de um problema.
- **Algoritmo:** é composto de instruções necessárias para o desenvolvimento de um programa de forma organizada e lógica.

VARIÁVEIS: São espaços alocados (reservados) na memória do computador para armazenar informações. Na maioria das vezes são do tipo **Inteiro**, **Real**, **Char**, entre outros, como veremos mais adiante.

ORGANIZAÇÃO BÁSICA DE UM COMPUTADOR

A estrutura de um computador pode ser esboçada através do seguinte esquema:



❖ Desse modo temos que:

Unidades de Entrada e de Saída

Recebe e exibe dados, além de realizar a comunicação com o meio exterior através dos periféricos de entrada (teclado e outros) e de saída (monitor e outros).

Unidade de Memória

Armazena informações que podem ser dados ou programas.

Unidade Lógica e Aritmética

Recebe os dados da memória para processá-los quando uma instrução aritmética ou lógica é executada.

Unidade de Controle

Determina a execução e a interpretação das instruções e controla o fluxo de dados.

INSTRUÇÕES PRIMITIVAS - PROCESSAMENTO DE DADOS

ENTRADA

Instrução: LEIA

SAÍDA

Instrução: ESCREVA

PASSOS PARA SE CONSTRUIR UM ALGORITMO

1. Ler atentamente o enunciado até compreendê-lo plenamente;
2. Identificar as entradas de dados do enunciado que serão fornecidas via teclado pelo usuário;
3. Definir os processamentos para transformar as entradas em saídas específicas;
4. Identificar as saídas que devem ser apresentadas para se atingir o resultado final, ou seja, o objetivo do algoritmo.

FORMAS DE REPRESENTAÇÃO DOS ALGORITMOS

Um algoritmo pode ser representado através de:

- Descrição Narrativa
- Pseudocódigo ou Português Estruturado
- Fluxograma ou Diagrama de Blocos

DESCRIÇÃO NARRATIVA

Nesta forma de representação os algoritmos são expressos diretamente em linguagem natural contendo:

- Linhas numeradas
- Verbo no infinitivo
- Complemento sucinto para o verbo

OBS: Caso seja necessário utilizar outro verbo provavelmente isso deva ser representado com uma nova ação a ser definida em uma nova linha de comando.

EXEMPLOS DE DESCRIÇÃO NARRATIVA

Exemplo 1

Descrição narrativa para **TROCAR UMA LÂMPADA**

- 1 - Iniciar
 - 2 - Pegar uma escada;
 - 3 - Buscar a lâmpada nova;
 - 4 - Posicionar a escada embaixo da lâmpada queimada;
 - 5 – Desligar o interruptor;
 - 6 - Subir na escada com a lâmpada nova;
 - 7 - Retirar a lâmpada queimada;
 - 8 - Colocar a lâmpada nova;
 - 9 - Descer da escada com a lâmpada queimada;
 - 10 – Ligar o interruptor;
 - 11 - Guardar a escada;
 - 12 - Descartar a lâmpada queimada;
 - 13 – Finalizar
-

Exemplo 2

Descrição Narrativa para **CALCULAR A MÉDIA DO ALUNO**

- 1 - Iniciar
 - 2 - Ler NOTA1;
 - 3 - Ler NOTA2;
 - 4 - Calcular $TOTAL = (NOTA1 + NOTA2)$;
 - 5 - Calcular $MEDIA = TOTAL / 2$;
 - 6 - Escrever (MEDIA);
 - 7 – Finalizar
-

EXERCÍCIOS

1. Trocar o pneu de um carro
2. Conceder desconto de estudante no cinema

FORMAS DE REPRESENTAÇÃO DOS ALGORITMOS

PSEUDOCÓDIGO OU PORTUGUÊS ESTRUTURADO

- Nesta forma de representação, os algoritmos são expressos numa linguagem simplificada, também conhecida popularmente como PORTUGOL.
- É possível executar as linhas de um pseudocódigo através do *software VISUALG*.
- Um recuo (tabulação) é inserido à esquerda dos comandos dos algoritmos entre o INICIO e o FIM de cada procedimento. Essa tabulação (TAB) é também chamada de **indentação** utilizada para definir o começo e o final de cada procedimento.
- Durante a elaboração, primeiramente definimos um nome para o algoritmo, conforme os exemplos a seguir.

EXEMPLO 1

Pseudocódigo para **TROCAR LÂMPADA**

PROGRAMA TROCA_LAMPADA

Início

```
Escreva ("Pegar uma escada");  
Escreva ("Buscar a lâmpada nova");  
Escreva ("Posicionar a escada embaixo da lâmpada queimada");  
Escreva ("Subir na escada");  
Escreva ("Retirar a lâmpada queimada");  
Escreva ("Colocar a lâmpada nova");  
Escreva ("Descer da escada");
```

Fim

EXEMPLO 2Pseudocódigo para **CALCULAR MÉDIA****PROGRAMA CALC_MEDIA**

VAR

AV1, AV2, TOTAL, MEDIA: Real;

Inicio

Escreva ("Digite a Nota1:");

Leia (AV1);

Escreva ("Digite a Nota2:");

Leia (AV2);

TOTAL \leftarrow AV1 + AV2;MEDIA \leftarrow TOTAL / 2;

ou

MEDIA \leftarrow (AV1 + AV2) / 2;

Escreva ("Sua Média é", MEDIA);

Fim

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 3

FORMAS DE REPRESENTAÇÃO DOS ALGORITMOS

FLUXOGRAMA OU DIAGRAMA DE BLOCOS

Nesta forma de representação os algoritmos são expressos numa linguagem gráfica e simbolizam instruções/ações a serem executadas em um programa.

Segue tabela contendo as simbologias mais utilizadas para as instruções mais relevantes de um processo.

DESCRIÇÃO	REPRESENTAÇÃO GRÁFICA
Início/ Fim	
Entrada de Dados	
Saída de Dados	
Conector de Processos (Usa-se letra para especificação de processos)	
Conector de Páginas (Usa-se o número da página em que o processo continua)	
Processamento	
Decisão	
Repetição	
Direção do Fluxo	

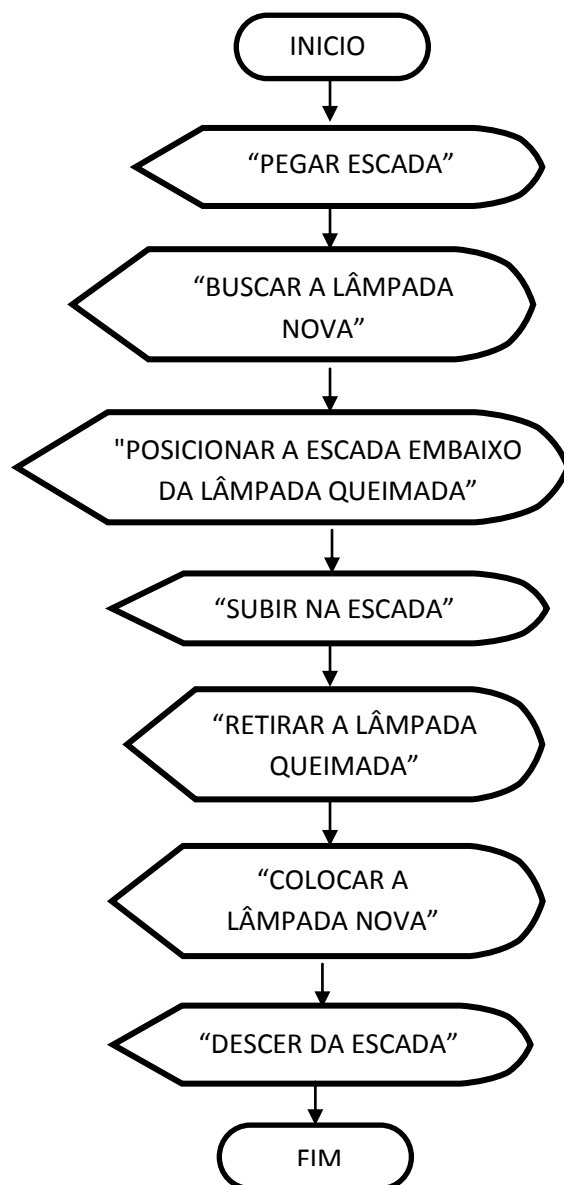
Existem alguns softwares utilizados para se construir Fluxogramas:

- MS WORD (Inserir Figuras Fluxograma)
- MS VISIO
- GLIFFY (on line)

Veja a seguir fluxogramas com as soluções dos problemas da **TROCA DA LÂMPADA** e **CÁLCULO DA MÉDIA** mencionados anteriormente.

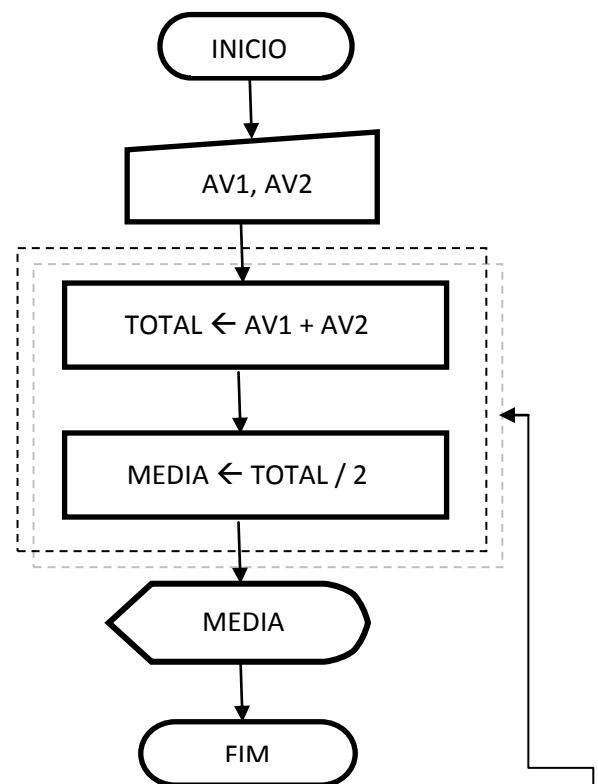
Exemplo 1

Fluxograma para **TROCAR DA LÂMPADA**

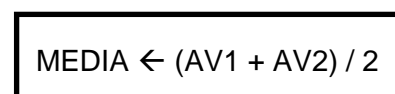


Exemplo 2

Fluxograma para **CÁLCULAR A MÉDIA**



Os dois procedimentos destacados na caixa pontilhada acima podem ser substituídos por um procedimento único conforme descrito a seguir:



OBS: Note a importância dos parênteses.

EXEMPLO: Caso as variáveis A e C e B e D sejam declaradas como inteiro e real, respectivamente, tem-se:

PROGRAMA TESTE

VAR

A, C: INTEIRO;

B, D: REAL;

INICIO

$A \leftarrow 9 / 2;$

$B \leftarrow 9 / 2;$

$C \leftarrow 9.0 / 2;$

$D \leftarrow 9.0 / 2;$

ESCREVA ("A = ", A);

ESCREVA ("B = ", B);

ESCREVA ("C = ", C);

ESCREVA ("D = ", D);

FIM

Observe que serão apresentadas na tela as saídas para A, B, C e D conforme segue:

A = 4

B = 4.000000

C = 4.000000

D = 4.500000

EXERCÍCIOS – Pseudocódigo e Fluxograma

- 1) Entre com dois números inteiros / Calcule o quadrado de cada um deles / Some os valores calculados / Apresente o resultado.
- 2) Some dois números inteiros
Calcule o dobro da soma e apresente-o.
- 3) Entre com dois números inteiros (A e B).
Efetue a troca dos valores digitados para essas variáveis.
DICA: Acrescente uma variável (C) para realizar os procedimentos.
Apresente os valores trocados das variáveis A e B.
- 4) Elaborar um programa que calcule e apresente o volume de uma caixa retangular, por meio da fórmula:
$$\text{VOLUME} \leftarrow \text{COMPRIMENTO} * \text{LARGURA} * \text{ALTURA}$$

OBS: Utilizar Números Reais
- 5) Elaborar um programa que apresente o valor da conversão em dólar (US\$) de um valor lido em reais (R\$). O programa deverá solicitar o valor da cotação do dólar e também a quantia em reais disponível pelo usuário.

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 4

TESTE DE MESA

- Após desenvolver um algoritmo é importante que seja realizado o seu teste lógico. Este procedimento é chamado de **Teste de Mesa** (ou Método Chinês).
- Nele as instruções do algoritmo são seguidas passo a passo (linha a linha) pelo programador no papel, ou seja, sob a mesa (Teste de Mesa) para prever a existência de eventuais erros.
- Assim, são observadas as **ocorrências das variáveis na memória do computador.**

EXEMPLO 1

Pseudocódigo - *Calcular Média 4 Notas de Alunos*

PROGRAMA MÉDIA

VAR

P1, P2, P3, P4, MEDIA : REAL;

INICIO

Escreva ("Entre com a Nota1: ");

Leia (P1);

Escreva ("Entre com a Nota2: ");

Leia (P2);

Escreva ("Entre com a Nota3: ");

Leia (P3);

Escreva ("Entre com a Nota4: ");

Leia (P4);

$MEDIA \leftarrow (P1+P2+P3+P4)/4;$

Escreva ("Sua média é: ", MEDIA);

FIM

Observe a seguir um **teste de mesa** que **simula** o cálculo da média dos alunos.

OCORRÊNCIAS NO MONITOR

Digite a Nota 1: 3
Digite a Nota 2: 5
Digite a Nota 3: 7
Digite a Nota 4: 9
Sua média é: 6

MEMÓRIA DO COMPUTADOR

P1	P2	P3	P4	MEDIA
3	5	7	9	6

Dessa forma, é possível simular as instruções e saber se elas retornarão ou não o resultado esperado.

EXERCÍCIO

Informe os valores finais para X, Y e Z

PROGRAMA XYZ

VAR

X, Y, Z : INTEIRO;

INICIO

X ← 1;

Y ← 2;

Z ← Y – X;

X ← X + 1;

Z ← Z - 1;

Y ← X + Y + Z;

Escreva (X, Y, Z);

FIM

X	Y	Z

EXERCÍCIOS

1) Observe o algoritmo abaixo e informe os valores finais para as variáveis A e X:

PROGRAMA VALORES

VAR

Y, Z, TOTAL, A, I, W, X : INTEIRO;

INICIO

Y ← 2;

TOTAL ← 10;

W ← 7;

A ← -4;

I ← 80;

X ← 4;

X ← I / TOTAL;

X ← X + 1;

A ← A + I;

Z ← 10;

FIM

2) Identifique os dados de Entrada (**E**), Processamento (**P**) e Saída (**S**):

(a) Total ← (Qtde_Peça x Valor_Unit) _____

(b) Leia (Cod_Peça) _____

(c) Escreva (Cod_Peça, Total) _____

(d) SubTotal ← (Qtde_Peça x 100) _____

(e) Leia (Valor_Unit) _____

(f) Leia (Qtde_Peça) _____

3) Dado o algoritmo a seguir, faça o teste de mesa e informe o valor para Delta (D) considerando:

A ← 2 B ← 5 C ← 3

ALGORITMO DELTA

VAR

A, B, C, D: INTEIRO;

INICIO

ESCREVA ("Digite o valor para A:");

LEIA (A);

ESCREVA ("Digite o valor para B:");

LEIA (B);

ESCREVA ("Digite o valor para C:");

LEIA (C);

D ← (B*B) – (4*A*C);

ESCREVA ("O valor de Delta é", D);

FIM

4) Faça um algoritmo que apresente a idade do usuário a partir do ano de nascimento e ano atual digitado por ele.

OBS: considere o ano com 4 dígitos.

- Apresente Pseudocódigo e Fluxograma.

- Faça o Teste de Mesa para os anos: 1957, 1971 e outro ano de nascimento definido por você.

OPERADORES

Os operadores são utilizados para calcular e comparar dados e podem ser:

- Operadores Aritméticos;
- Operadores Relacionais;
- Operadores Lógicos.

OPERADORES ARITMÉTICOS

São utilizados para realizar operações a fim de se obter resultados numéricos. Os símbolos estão descritos a seguir:

OPERAÇÃO	SÍMBOLO
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Módulo	% (Resto de uma divisão)

OPERADORES RELACIONAIS

Comparam valores retornando como resposta: Verdadeiro ou Falso

DESCRIÇÃO	SÍMBOLO
Igual a	=
Diferente de	< >
Maior que	>
Menor que	<
Menor ou igual a	< =
Maior ou igual a	> =

OPERADORES LÓGICOS

Retornam resposta verdadeiro ou falso e podem ser:

AND → .E.

OR → .OU.

NOT → .NÃO.

AND : É verdadeiro se todas as condições forem verdadeiras;

OR : É verdadeiro se pelo menos uma condição for verdadeira;

NOT : Inverte o valor da condição, ou seja, considera **VERDADEIRO** como **FALSO** e vice-versa.

A tabela a seguir mostra todos os valores possíveis para os 3 operadores (AND, OR e NOT) considerando T (True = verdadeiro) e F (False = Falso).

1º VALOR	OPERADOR	2º VALOR	RESULTADO
T	AND	T	T
T	AND	F	F
F	AND	T	F
F	AND	F	F
T	OR	T	T
T	OR	F	T
F	OR	T	T
F	OR	F	F
T	NOT	--	F
F	NOT	--	T

EXEMPLO: Suponha as variáveis **A=5 B=8 C=1**

Observe uma comparação entre as variáveis na tabela a seguir:

EXPRESSÕES					RESULTADO
A = B	F	AND	B > C	T	FALSO
A < > B	T	OR	B < C	F	VERDADEIRO
A > B	F	NOT	--	-	VERDADEIRO
A < B	T	AND	B > C	T	VERDADEIRO
A > = B	F	OR	B = C	F	FALSO
A < = B	T	NOT	--	-	FALSO

EXERCÍCIOS

1. Para **A=4, B=1 e C=3** informe verdadeiro ou falso para as expressões:

- | | | | |
|------------|-----|-------------|-----|
| a) A=B | () | d) C<(B+A) | () |
| b) A < > B | () | e) A>=(C+B) | () |
| c) A>(B+C) | () | f) A<=B | () |

2. Sabendo que **A=3, B=7 e C=4**, informe se as condições/ expressões a seguir são verdadeiras ou falsas.

- | | | | |
|---------------|-----|---------------|-----|
| a) (A+C) > B | () | d) (B+A) <= C | () |
| b) B >= (A+2) | () | e) (C+A) > B | () |
| c) C = (B-A) | () | | |

3. Agora considere: **A=5, B=4, C=3 e D=6**

- | | | | |
|----------------------------------|-----|------------------------|-----|
| a) (A>C) .AND. (C<=D) | () | c) (A>=C) .AND. (D>=C) | () |
| b) (A+B) > 10 .OR. (A+B) = (C+D) | () | d) .NOT. (A<=D) | () |

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 5**CÁLCULOS DE PORCENTAGEM**

Seguem exemplo de métodos para cálculos de porcentagem:

(1) Para calcular o quadrado de um número temos:

$$X^2 = X * X$$

(2) Para calcular a porcentagem temos:

$$X\% = (X / 100)$$

EXEMPLOS

(a) $1\% = 1/100 = 0,01$

(b) $5\% = 5/100 = 0,05$

(c) $10\% = 10/100 = 0,10$ ou $0,1$

(d) $15\% = 15/100 = 0,15$

(e) $50\% = 50/100 = 0,50$ ou $0,5$

(f) Calcule o **TOTAL** de 10% de **AUMENTO** sobre o **PREÇO** de um produto

$$\text{AUMENTO} = \text{PREÇO} * 10/100$$

ou

$$\text{AUMENTO} = \text{PREÇO} * 0,1$$

Assim temos que:

$$\text{TOTAL} = (\text{PREÇO} + \text{AUMENTO})$$

(g) Calcule quanto **PAGAR** para 25% de **DESCONTO** sobre o **VALOR** de uma peça

$$\text{DESCONTO} = \text{VALOR} * 25/100$$

ou

$$\text{DESCONTO} = \text{VALOR} * 0,25$$

Assim temos que:

$$\text{PAGAR} = (\text{VALOR} - \text{DESCONTO})$$

EXERCÍCIOS AULA 5 → Pseudocódigo e Fluxograma

EXERCÍCIO 1

Ler Horas Trabalhadas (HT)

Ler Valor da Hora de trabalho (VH)

Ler Percentual de Desconto do INSS (PD)

Calcular o Salário Bruto $SB = HT * VH$

Calcular o Total de Desconto $TD = SB * PD / 100$

Calcular o Salário Líquido $SL = SB - TD$

Ao final apresentar os seguintes valores:

- SALÁRIO BRUTO
- TOTAL DE DESCONTO
- SALÁRIO LÍQUIDO

EXERCÍCIO 2

Ler a Temperatura em Graus Centígrados (C) e apresentar conversão para Graus Fahrenheit (F), utilizando a seguinte fórmula: $F = (9 * C + 160) / 5$

EXERCÍCIO 3

Calcule e Apresente o SALDO bancário a partir dos valores lidos CRÉDITO e DÉBITO.

Faça o teste de mesa de 3 ocorrências.

EXERCÍCIO 4

Calcular e apresentar o **Volume** do Cilindro – Fórmula = $\pi \times \text{Raio}^2 \times \text{Altura}$

EXERCÍCIO 5

Calcule o **TOTAL** do **PREÇO** de um produto após **AUMENTO** de 9%. Apresente os seguintes valores:

PREÇO PADRÃO:

AUMENTO:

TOTAL COBRADO:

EXERCÍCIO 6

Calcule o valor a **PAGAR** por um produto cujo **VALOR** recebeu **DESCONTO** de 5%. Apresente os seguintes valores:

VALOR PADRÃO:

DESCONTO:

TOTAL COBRADO:

EXERCÍCIO 7

Calcule o **TOTAL** do **PREÇO** de um produto após um **AUMENTO** em porcentagem (%) digitado pelo usuário. Apresente os seguintes valores:

PREÇO PADRÃO:

AUMENTO:

TOTAL COBRADO:

EXERCÍCIO 8

Calcule o valor a **PAGAR** por um produto cujo **VALOR** recebeu um **DESCONTO** em porcentagem (%) digitado pelo usuário. Apresente os 3 valores:

VALOR PADRÃO:

DESCONTO:

TOTAL COBRADO:

EXERCÍCIO 9

Faça o algoritmo que calcule o **Estoque Médio** de peças identificadas pelo **Código** sabendo-se que corresponde a **quantidade mínima** mais a **quantidade máxima** dividido por 2.

Apresentar na tela → **Estoque Produto Código: 99 Média 9999**

Elabore o **Teste de mesa** para os valores apresentados a seguir:

Qtde Mínima 2 e Qtde Máxima 8

Qtde Mínima 300 e Qtde Máxima 750

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 6

LINGUAGEM C

A linguagem C foi criada em 1972 e tornou-se amplamente utilizada por sua flexibilidade e portabilidade. Foi derivada das linguagens *Algol 68* e *BCPLC/B*.

VARIAÇÕES DA LINGUAGEM C

C - Linguagem com códigos implementados de forma estruturada e que foi utilizada para desenvolver SOs (Sistemas Operacionais) tais como UNIX, LINUX e WINDOWS.

Ela permite gerenciar os endereços de memória diretamente.

C++ - É uma evolução da linguagem C possibilitando uma Programação Orientada a Objetos (OOP - *Object-Oriented Programming*).

C# - (Lê-se: C-SHARP) - Linguagem desenvolvida para plataforma .NET, ou seja, uma tecnologia criada para criar projetos em linguagem C voltados para a WEB.

CARACTERÍSTICAS DA LINGUAGEM C

- É uma linguagem de alto nível com sintaxe estruturada que gera um programa fonte com extensão (.C)
- Os programas fontes são compilados gerando programas executáveis com extensão (.EXE) Eles são enxutos e geralmente mais velozes que a maioria dos gerados em outras linguagens.
- Permite compartilhar recursos de linguagens de alto nível com recursos de linguagens de baixo nível.

OBS1: Linguagens de Alto Nível: Programação legível e intuitiva com comandos que se aproximam da linguagem humana – Ex.: *Delphi*.

Linguagens de Baixo Nível: Programação direta nos microprocessadores e menos intuitiva, pois se aproxima da linguagem de máquina – Ex. *Assembly*.

OBS2: O ASSEMBLER é o compilador da linguagem ASSEMBLY.

CONSTANTES

Constante é um valor fixo que não se altera durante a execução do programa, tal como:

Formula da água	= H ₂ O	Voltagem eletrônica	= 5 Volts
Maioridade	= 18 anos	Valor de PI	= 3,1415...

→ No pseudocódigo as constantes são chamadas de modificadores. Assim, um valor é atribuído durante a sua declaração que é feita na linha que antecede a sua utilização (no meio do algoritmo).

EXEMPLO: CONST PI = 3,1415: Real;

→ Na Linguagem C, declaram-se constantes de duas formas:

(a) Como modificador

EXEMPLO: const float PI = 3.1415;

(b) Como diretiva (antes da seção de declaração das variáveis) sem ponto e vírgula no final

EXEMPLO: #define PI 3,1415 (lê-se: **sustenido define**)

VARIÁVEIS

Conforme mencionado anteriormente, variáveis são espaços reservados na memória do computador para armazenar um tipo específico de informação.

Elas armazenam um valor por vez, ou seja, seus valores podem se modificar (variar) durante a execução do programa.

O ideal é que as variáveis recebam nomes mnemônicos para facilitar a sua identificação.

TIPOS DE VARIÁVEIS

A seguir são listados os tipos de variáveis mais utilizados na Linguagem C:

TIPO	TAMANHO	APLICAÇÃO	FAIXA DE VALORES
char	1 byte	Caractere	-128 a 127
int	2 bytes	Contadores, Controle de laços, etc.	- 2.147.483.648 a 2.147.483.647
float	4 bytes	Reais (Precisão 6 dígitos)	10^{-38} a 10^{38}
double	8 bytes	Reais (Precisão 10 dígitos) aceita Números Científicos	10^{-308} a 10^{308}

**** OBS:** Tipo **Void** pode ser declarado para variáveis sem tipo definido em que, posteriormente, podem ser alteradas para um tipo específico.

PALAVRAS RESERVADAS

Existem expressões que não podem ser definidos como nomes de variáveis, pois tratam-se de palavras reservadas e portanto, restritas às linguagens de programação.

EXEMPLOS:

auto	class	else	public	if	char
break	do	long	new	white	double
case	const	for	delete	goto	float

FUNÇÕES DA LINGUAGEM C

FUNÇÃO main()

A função main() é a função principal e marca o ponto de início da execução do programa.

Os parênteses indicam que trata-se de uma função.

Um exemplo simples de programa em C é apresentado a seguir.

Exemplo:

```
#include <stdio.h>
main( )
{
    printf("Primeiro Programa - Hello, World!");
}
```

OBS: Note que as linhas de comandos aparecem entre chaves que correspondem ao INÍCIO e ao FIM dos comandos nas funções.

FUNÇÃO printf()

A função printf() é utilizada para saída de dados.

CARACTERES ESPECIAIS

Existem alguns códigos da linguagem C utilizados para caracteres que não podem ser inseridos diretamente pelo teclado conforme descritos a seguir.

\n – Nova linha

\t – TAB – Tabulação

\b – Retrocesso (*BackSpace*)

\“ – Aspas

\\ – Barra

\r – Retorno do cursor para o início da linha

\f – Salta página de formulário durante a impressão

CARACTERES de FORMATAÇÃO

A seguir são apresentados alguns códigos de formatação para exibir dados na tela.

%d – Números inteiros com Base Decimal (0 a 9).

%f – Ponto Flutuante (casas decimais) – **Ex.:** %.2f – Para formatar **2** casas decimais.

%c – Caractere simples - Código da Tabela ASCII (usa-se aspas simples ‘ ‘)

%s – Cadeia de caracteres ou texto (usa-se aspas duplas “ ”)

%i – Aceita números inteiros. Entretanto é específico para uso com Base Octal (8, 10 e 16).

Exemplo da Função printf()

Vide exemplo da função main().

FUNÇÃO scanf()

- A função scanf() é utilizada para entrada de dados.

- Os endereços de variáveis da função scanf() requerem o operador de endereço “&”.

Eles retornam o conteúdo de uma variável. Se o operador de endereço & não for utilizado será retornado o endereço de memória da variável e não o seu conteúdo, como esperado.

scanf() - CÓDIGOS de FORMATAÇÃO

Esses códigos são utilizados para formatação de conteúdos a serem lidos pela digitação do usuário e armazenados na memória.

%c	Caractere da Tabela ASCII	%f	Número em Ponto Flutuante
%d	Número inteiro	%s	Cadeia de String - Texto
%e	Número em Notações científicas	%lf	Número Double

Exemplo de linha de código scanf()

```
scanf("%f", &SALDO);
```

EXEMPLO DE PROGRAMA

```
#include <stdio.h>
main ( )
{
    int A, B, C;

    printf ( " Entre com um valor:" );
    scanf ( "%d", &A);
    printf ( "Entre com outro valor:" );
    scanf ( "%d", &B);
    C = A + B;
    printf ( "A soma é: %d" , C);
}
```

Exemplos de formatações para printf()

```
printf ( "%d" , C);
printf ( "A soma de %d e %d é %d" , A, B, C);
printf ( "%d + %d = %d \n" , A, B, C);
```

EXERCICIO1 - Pseudocódigo/Fluxograma/Linguagem C

Efetuar o cálculo e apresentar o valor de uma prestação em atraso, utilizando a fórmula:

prestação \leftarrow valor + (valor*(taxa/100)*tempo)

EXERCICIO2 - Linguagem C

Elaborar os enunciados dos exercícios das Páginas 21 e 22 para execução no Laboratório.

MINI TUTORIAL DEV C++ ou DEV CPP (DEV C PLUS PLUS)


Execute o programa em **INICIAR /PROGRAMAS /BLOODSHED DEV-C++ /DEV-C++**

Será iniciado o ambiente de programação do Dev-C++

Será visualizada uma janela contendo a *Dica do Dia* - leia feche-a.

Clique sobre os comandos:

ARQUIVO / NOVO / ARQUIVO FONTE

ou pressionar o ícone  localizado abaixo do comando *Editar* da barra de menu. Aparecerá uma tela com duas áreas vazias, sendo que o programa será digitado na maior delas (à direita).

BIBLIOTECAS: A diretiva **#include** permite o acesso às bibliotecas específicas.

A biblioteca principal (cabeçalho padrão) para reconhecimento dos comandos de entrada e de saída (scanf e printf) é chamada **<stdio.h>** que corresponde a **Standard Input-Output Header**.

A biblioteca padrão **<stdlib.h>** corresponde a **Standard Library** pode ser inserida para reconhecer outros comandos no **DEV CPP** como **system("...")**, por exemplo.

Também a biblioteca matemática **<math.h>** é utilizada em algoritmos que utilizam funções matemáticas trigonométricas, logarítmicas, etc.

COMENTÁRIOS EM LINHAS DE CÓDIGO

Os comentários nas linhas dos códigos em Linguagem C (**DEV C++**) podem ser inseridos de 2 formas:

// Comentários em linha aqui

/* Comentários em bloco aqui */

ALTERANDO TÍTULO DA JANELA DE EXECUÇÃO E CORES DE TELA/ TEXTO

Valores hexadecimais (**0 a 9** – Fundo + **A a F** Texto)

system ("title TESTE");

system ("color 5D"); // Exemplo: Fundo aparecerá em Roxo e Texto em Lilás

system ("color 1F"); // Exemplo: Fundo aparecerá em Azul e Texto em Branco

Para efeito de teste digite o seguinte programa:

```
/* Programa 1 - Teste */
#include <stdio.h>
main ( )
{
    printf ("Programa desenvolvido por => Seu nome\n");
    system ("pause");
}
```

OBS 1: Dependendo da configuração do computador, talvez seja necessário pressionar a barra de espaço após digitar o caractere " para que ele apareça na tela.

OBS 2: Confira sempre se não existe erros de digitação.

OBS 3: Depois de digitar o programa é preciso **Salvar, Compilar e Executar**.

► **SALVANDO O PROGRAMA**

Clique sobre os comandos **ARQUIVO / SALVAR COMO...** ou sobre o

ícone 


Na caixa **Salvar em:** selecione o local para armazenar seus arquivos;

Digite o nome do programa em C na caixa **Nome:** **Teste** (por exemplo)

Selecione na caixa **Tipo** a opção **C source files (*.c)** e clique sobre o botão **SALVAR**

OBS 4: Conforme a descrição do exemplo acima, seu programa digitado foi salvo na pasta selecionada, com o nome Teste e possui extensão.C


► **COMPILANDO O PROGRAMA**

Para compilar o programa utilize os comandos **EXECUTAR / COMPILAR** ou o ícone .

OBS 5: Depois que o programa for compilado aparecerá um janela mostrando as ocorrências do **Processo de Compilação** e espera-se que na caixa **Errors** dessa janela contenha o valor 0. Isso indica que seu programa foi compilado corretamente e que um arquivo executável (**.EXE**) também foi criado com sucesso no mesmo local.

OBS 6: Nesse momento é possível fechar o arquivo e encerrar o programa DEV CPP. Entretanto, costuma-se conferir se a execução está conforme esperada executando o programa desenvolvido.

► **EXECUTANDO O PROGRAMA**

Ao verificarmos que o programa não apresenta erros podemos executá-lo através dos comandos: **EXECUTAR / EXECUTAR** ou através do ícone .

OBS 7: Uma janela do **DOS** apresentará a execução do programa.

► **SOBRE O SYSTEM PAUSE**

OBS 8: Observe que depois que o programa termina, ainda na janela do **DOS**, aparecerá a mensagem **Pressione qualquer tecla para continuar . . .** Ela aparece por conta do comando **system ("pause")** e, muito sugestiva, aguarda que seja pressionada qualquer tecla, ou que a janela seja fechada para encerrar definitivamente a execução do programa.

OBS 9: Para não aparecer a mensagem **Pressione qualquer tecla para continuar . . .** utilize o comando **system("pause > null")** também seguido de ponto e vírgula.

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 7

ESTRUTURAS DE CONTROLE – Tomadas de Decisão - Desvio Condicional

LÓGICA

SE... ENTÃO... SENÃO

IF... THEN ... ELSE

OBS: A instrução ENTÃO sempre é utilizada nos pseudocódigos e também é aplicável para várias linguagens. Porém, o comando correspondente **THEN** não é aplicável particularmente na Linguagem C.

SINTAXE:

SE <condição> **ENTÃO**

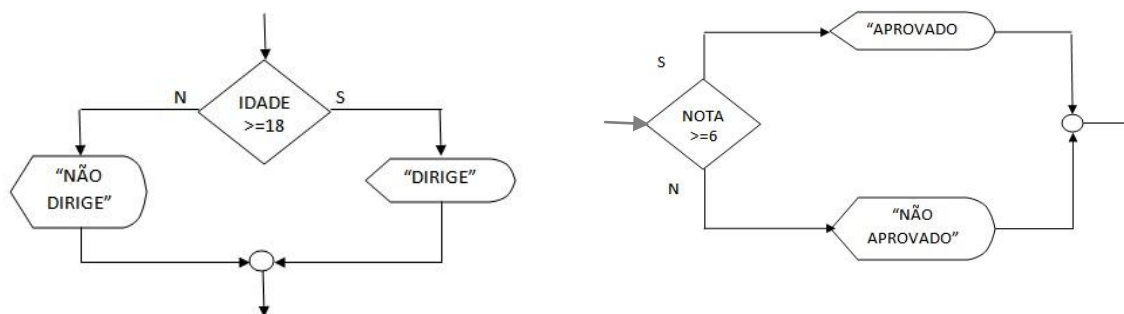
Instruções para SIM;

SENÃO

Instruções para NÃO;

FIM_SE

MODELOS DE FLUXOGRAMA (DIAGRAMAS DE BLOCOS)



DESVIO CONDICIONAL SIMPLES

Pseudocódigo e Fluxograma para:

- Receber 2 valores para as variáveis A e B.
- Efetuar a soma dos valores A e B e atribuir o resultado à variável C.
- Se o resultado da soma for maior que 10, apresentar o resultado.

**PORTUGUÊS ESTRUTURADO
(PSEUDOCÓDIGO)**

PROGRAMA SOMA

VAR

A, B, C: INTEIRO;

INICIO

LEIA (A);

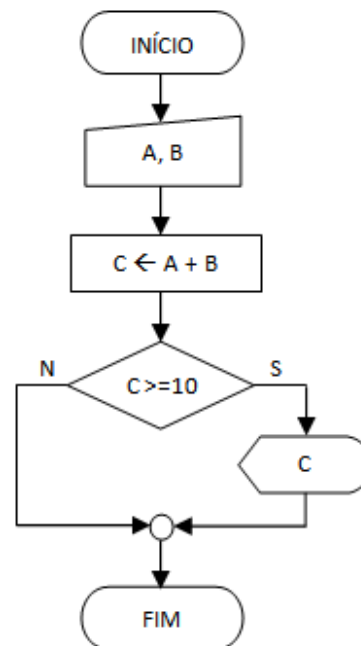
LEIA (B);

 $C \leftarrow A+B$;**SE** ($C \geq 10$) **ENTÃO**

ESCREVA (C);

FIM_SE

FIM

**DIAGRAMA DE BLOCOS
(FLUXOGRAMA)****DESVIO CONDICIONAL COMPOSTO****Pseudocódigo e Fluxograma para:**

- Receba 2 valores (C e D).
- Calcular a subtração (C-D) para Y.
- Comparar Y é igual ou superior a 50
 Se sim $R \leftarrow Y + 10$
 Senão $R \leftarrow Y + 20$
- Ao final apresente o valor de R.

**PORTUGUÊS ESTRUTURADO
(PSEUDOCÓDIGO)**

PROGRAMA SUBTRAÇÃO

VAR

C, D, Y, R: INTEIRO;

INICIO

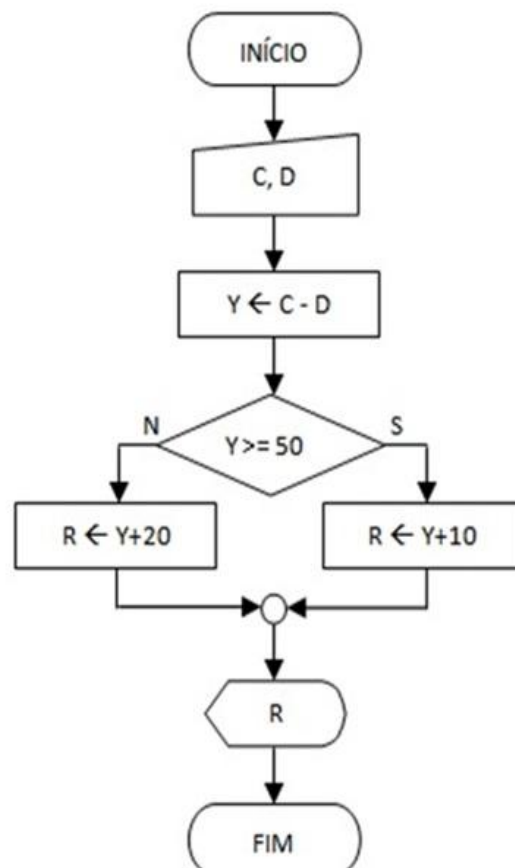
LEIA (C);

LEIA (D);

 $Y \leftarrow C - D$;**SE** ($Y \geq 50$) **ENTÃO** $R \leftarrow Y+10$;**SENÃO** $R \leftarrow Y+20$;**FIM_SE**

ESCREVA (R);

FIM

**DIAGRAMA DE BLOCOS
(FLUXOGRAMA)**

DESVIO CONDICIONAL ENCADEADO ou ANINHADO

- Verifica condições sucessivas, ou seja, uma ação executada pode gerar novas condições.
- Dessa forma, utiliza-se uma condição dentro de outra condição.
- Essas estruturas são chamadas de **Estruturas Encadeadas ou Aninhadas**

EXEMPLO – Pseudocódigo e Fluxograma

Considere a seguinte situação:

Na entrada para um evento, pedir que usuário informe: **1** - Feminino ou **2** - Masculino.

Se **1** com **mais de 25 anos** Apresentar texto **Entrada VIP** Caso contrário **Desconto 50%**

Se **2** com **mais de 30 anos** Apresentar texto **Desconto de 20%** Caso contrário **Desconto 10%**

PROGRAMA EVENTO
VAR

S, I : INTEIRO;

INICIO

LEIA (S);

LEIA (I);

SE (S = 1) ENTÃO

SE (I > 25) ENTÃO

ESCREVA ("VIP");

SENÃO

ESCREVA ("50% Desconto");

FIM_SE

SENÃO

SE (I > 30) ENTÃO

ESCREVA ("20% Desconto");

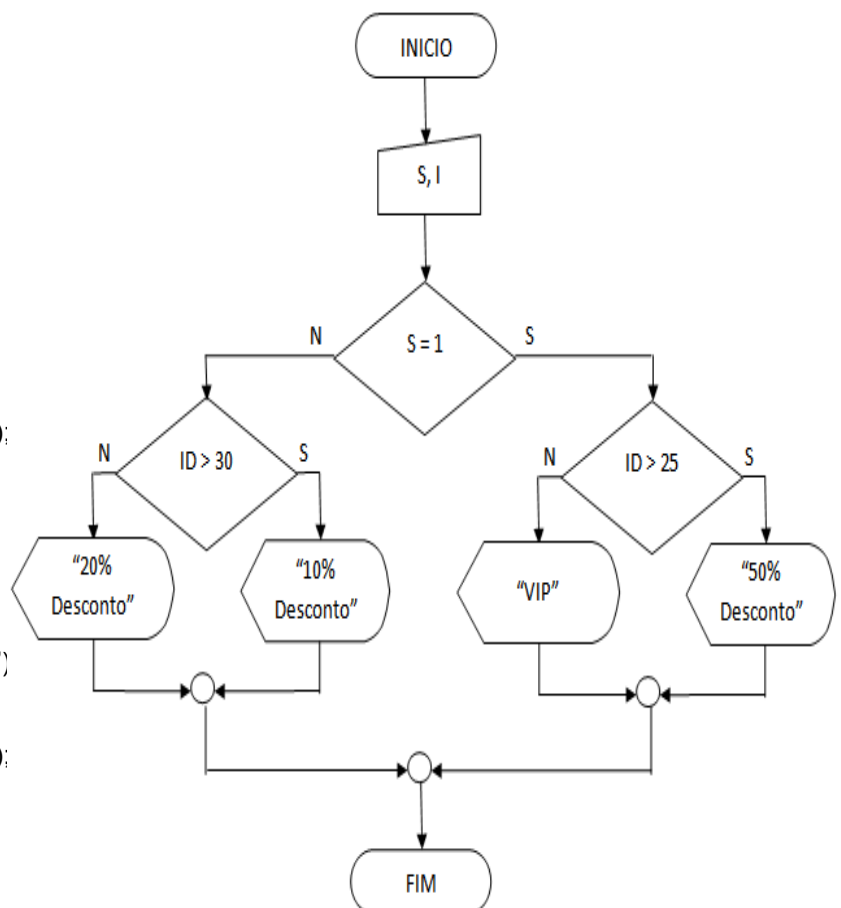
SENÃO

ESCREVA ("10% Desconto");

FIM_SE

FIM_SE

FIM



EXEMPLOS EM LINGUAGEM C

A seguir são apresentados os exemplos anteriores **em Linguagem C**.

DESVIO CONDICIONAL SIMPLES

```
#include <stdio.h>
main()
{
    int a, b, c;
    printf("Digite um valor: ");
    scanf("%d", &a);
    printf("Digite outro valor: ");
    scanf("%d", &b);
    c=a+b;
    if(c>=10)
        printf("O valor da Soma e: %d \n", c);
    system("pause");
}
```

DESVIO CONDICIONAL COMPOSTO

```
#include <stdio.h>
main()
{
    int c, d, y, r;
    printf("Digite um valor: ");
    scanf("%d", &c);
    printf("Digite outro valor: ");
    scanf("%d", &d);
    y=c-d;
    if(y>=50)
        r=y+10;
    else
        r=y+20;
    printf("O valor de R e: %d \n", r);
    system("pause");
}
```

DESVIO CONDICIONAL ENCADEADO ou ANINHADO

```
#include <stdio.h>
main()
{
    int I, S;
    printf("Digite o Sexo (1 - Feminino ou 2 - Masculino): ");
    scanf("%d", &S);
    printf("Digite a Idade:");
    scanf ("%d", &I);
    if (S == 1)
    {
        if (I>25)
            printf("50%% Desconto\n");
        else
            printf("Entrada VIP\n");
    }
    else
    {
        if (I>30)
            printf("10%% Desconto\n");
        else
            printf("20%% Desconto\n");
    }
    system("pause");
}
```

EXERCÍCIOS – Pseudocódigo / Fluxograma / Linguagem C

OBS: Os exercícios a seguir possuem as mesmas estruturas dos **Exemplos 1, 2 e 3**, respectivamente, e que podem ser utilizados para as resoluções.

(1) De acordo com o SEBRAE, uma indústria com até 19 funcionários é classificada como **MicroEmpresa**.

- Ler a quantidade de funcionários e apresentar a definição acima caso corresponda.

(2) Os alunos de uma disciplina com carga horária de 80h/semestre são reprovados caso tenham mais que 20 faltas.

- Ler a quantidade de faltas e apresentar se o aluno está **Reprovado por Faltas** ou **Não Reprovado por Faltas**

(3) Ler a quantidade de arestas

Apresentar a quantidade de arestas digitada com a mensagem correspondente:

< 3 - Não é Figura Geométrica

= 3 - É um triângulo

> 3 - Não é um triângulo

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 8

OPERADORES LÓGICOS AND, OR e NOT

TESTE LÓGICO	PSEUDOCÓDIGO	LINGUAGEM C	ATRIBUIÇÃO	COMPARAÇÃO	
AND	.E.	&&	$A \leftarrow B$	A=B	A≠B
OR	.OU.		A=B	A==B	A!=B
NOT	.NÃO.	!	ASSIM: SE A=5	LINGUAGEM C	
			TEM-SE: B=5	IF (A==B)	IF (A!=B)

TESTES LÓGICOS → PSEUDOCÓDIGOS (PORTUGUÊS ESTRUTURADO) e LINGUAGEM C

PROGRAMA TESTAR_LÓGICA_AND_E

```

VAR
N : INTEIRO;
INICIO
    LEIA (N);
    SE ((N >= 20) .E. (N <= 90)) ENTÃO
        ESCREVA ("O NÚMERO ESTÁ
        NA FAIXA DE 20 A 90");
    SENÃO
        ESCREVA ("O NÚMERO ESTÁ
        FORA DA FAIXA DE 20 A 90");
    FIM_SE
FIM
  
```

```

#include <stdio.h>
main()
{
    int n;

    printf ("Entre com um Valor:");
    scanf ( "%d", &n);

    if ((n>=20) && (n<=90))
        printf ("O valor %d esta entre 20 e 90 \n\n", n);
    else
        printf ("O valor %d esta fora da faixa 20 e 90 \n\n", n);

    system ("pause");
}
  
```

PROGRAMA TESTAR_LÓGICA_OR_OU

```

VAR
S : INTEIRO;
INICIO
    ESCREVA ("DIGITE 1 – SAQUE OU 2 –
    SALDO");
    LEIA (S);
    SE ((S = 1) .OU. (S = 2)) ENTÃO
        ESCREVA ("OPERAÇÃO BANCÁRIA
        VÁLIDA");
    SENÃO
        ESCREVA ("OPERAÇÃO BANCÁRIA
        INVÁLIDA");
    FIM_SE
FIM
  
```

```

#include <stdio.h>
main()
{
    int s;
    printf ("DIGITE: 1 - SAQUE OU 2 - SALDO\n\n");
    scanf ( "%d", &s);

    if ((s==1) || (s==2))
        printf ("OPERACAO BANCARIA VALIDA\n\n");
    else
        printf ("OPERACAO BANCARIA INVALIDA\n\n");

    system ("pause");
}
  
```

PROGRAMA TESTAR_LÓGICA_NOT_NÃO

```

VAR
A,B,C,X : INTEIRO;
INICIO
    A ← 4;
    B ← 2;
    LEIA (X);
    SE (.NÃO. (X > 5)) ENTÃO
        C ← (A + B) * X;
    SENÃO
        C ← (A – B) * X;
    FIM_SE
    ESCREVA (C);
FIM
  
```

```

#include <stdio.h>
main()
{
    int a,b,c,x;

    a=4;
    b=2;
    printf ("Digite um valor:");
    scanf ("%d", &x);

    if (!(x>5))
        c=(a+b)*x;
    else
        c=(a-b)*x;

    printf ("%d\n\n",c);

    system ("pause"); }
  
```

IMPORTANTE: É preciso inserir **chaves adicionais** (INICIO e FIM) para blocos de comandos com saídas (*Verdadeiro* ou *Falso*) que contenham mais de uma linha de instrução.

FLUXOGRAMAS DE DECISÕES EM TESTES LÓGICOS

Exemplo Fluxograma AND	Exemplo Fluxograma OR	Exemplo Fluxograma NOT
<pre> graph TD A{ "(NUM >= 20) .AND. (NUM <= 90)" -- N --> B[] A -- S --> C[] style B fill:none,stroke:none style C fill:none,stroke:none </pre>	<pre> graph TD A{ "(S = 1) .OU. (S = 2)" -- N --> B[] A -- S --> C[] style B fill:none,stroke:none style C fill:none,stroke:none </pre>	<pre> graph TD A{ ".NÃO. (X > 5)" -- N --> B[] A -- S --> C[] style B fill:none,stroke:none style C fill:none,stroke:none </pre>

EXERCÍCIOS → PSEUDOCÓDIGO / FLUXOGRAMA / LINGUAGEM C SALA DE AULA + LABORATÓRIO

ELABORAR OS SEGUINTE ENUNCIADOS:

(1) TESTE LÓGICO COM AND

Ler um número e informar texto conforme segue:

Valor na Faixa entre 0 e 10

Valor fora da Faixa entre 0 e 10

(2) TESTE LÓGICO COM OR

Ler opção 1 (*Funcionário*) **OU** 2 (*Ex-Funcionário*) e apresentar a mensagem:

Pessoa TEM ou JÁ TEVE vínculo com a empresa.

Apresentar: **Pessoa nunca teve vínculo com a empresa**
(Caso resposta não seja 1 ou 2)

(3) TESTE LÓGICO COM NOT

Ler um valor numérico inteiro e verificar se ele **NÃO** é maior que 3. Apresentar as mensagens correspondentes:

- **Valor é maior que 3**
- **Valor não é maior que 3**

EXERCÍCIOS ADICIONAIS

1) A Prefeitura de Poá-SP abriu uma linha de crédito para os funcionários estatutários. O valor máximo da prestação não poderá ultrapassar 30% do salário bruto. Fazer um algoritmo que permita entrar com o salário bruto e o valor da prestação, e informar se o empréstimo pode ou não ser concedido.

2) Considere as seguinte condições para Cálculo do Reajuste de Salário:

Salário < 500 → 15% de Reajuste
Salário <= 1000 → 10% de Reajuste
Salário > 1000 → 5% de Reajuste

3) Elabore um programa para manter um equilíbrio no estoque de produtos de uma empresa observando:

Se **Estoque<500 peças** – Calcular 30% **a mais** que a qtde de peças informada

Se **Estoque<=1000 peças** – Calcular 20% **a mais** que a qtde de peças informada

Se **Estoque>1000 peças** – Calcular 10% **a mais** que a qtde de peças informada

- Ao final, ***apresentar a quantidade que deverá ser produzida.***

4) Elabore um programa que Leia idade e exiba as seguintes mensagens de acordo com a idade sobre pagamento de passagens:

- “Crianças até 5 anos – Não Pagam Passagem”
 - “ Adultos com mais de 65 anos – Isentos de Pagamento de Passagem”
 - “Passageiros entre 6 e 64 anos – Pagam Passagem”
-

5) Elabore um algoritmo que lê um número inteiro e informa se o número lido é negativo ou positivo. O valor zero será considerado positivo.

6) Elabore um algoritmo para determinar se um dado número N (recebido através do teclado) é POSITIVO, NEGATIVO ou NULO.

7) Elabore um algoritmo que leia a idade de uma pessoa e informe a sua classe eleitoral:

- não eleitor (abaixo de 16 anos);
 - eleitor obrigatório (entre a faixa de 18 e menor de 65 anos);
 - eleitor facultativo (de 16 até 18 anos e maior de 65 anos, inclusive).
-

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 9

INCREMENTO

Variáveis podem ser incrementadas, ou seja, acumulam-se (acrescentam-se) valores em através do processo de incremento (adição).

Exemplos:

A=A+1

A++ (em Linguagem C)

DECREMENTO

As variáveis também podem ser decrementadas, ou seja, retiram-se (diminuem-se) valores através do processo de decremento (subtração).

Exemplos:

A=A-1

A-- (em Linguagem C)

INCREMENTO e DECREMENTO - PRÉ-FIXADO

Durante a atribuição, primeiramente, a variável A será incrementada/decrementada e, desse modo, a variável B receberá a atribuição com o incremento/decremento.

Exemplo Incremento:

A=10

B=++A

Exemplo Decremento:

A=10

B=--A

Após a execução das linhas no exemplo acima, na memória teremos respectivamente:

A=11

B=11

A=9

B=9

INCREMENTO e DECREMENTO - PÓS-FIXADO

Durante a atribuição, primeiramente, a variável B receberá a atribuição sem o incremento/decremento e, a seguir, a variável A será incrementada/decrementada.

Exemplo Incremento:

A=10

B=A++

Exemplo Decremento:

A=10

B=A--

Após a execução das linhas no exemplo acima, na memória teremos respectivamente:

A=11 **ou seja** B=10

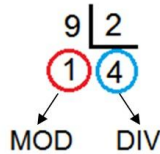
B=10 A=11

A=9 **ou seja** B=10

B=10 A=9

OPERADORES DIV e MOD

A Figura a seguir apresenta o procedimento de cálculo de uma divisão de 9 por 2. Nela são destacados o quociente (resultado) da divisão =4 e também o resto da divisão =1. Para se obter esses valores é preciso utilizar os operadores **DIV** e **MOD** conforme ilustradas na Figura abaixo.



OPERADOR DIV (Quociente da Divisão)

Recebe dois valores (dividendo e divisor) e calcula o quociente (resultado inteiro) da divisão.

Sintaxe Pseudocódigo (Uso do operador DIV)

Dividendo = 9

Divisor = 2

Quociente = Dividendo DIV Divisor

→ Teremos que a variável **Quociente** recebe o resultado (inteiro) do cálculo 9/2, ou seja, Quociente = 4

Exemplo Linguagem C (Uso de variável tipo inteiro e operador “/”)

```
int A, B, C;
```

```
A = 9;
```

```
B = 2;
```

```
C = A / B; (desse modo, C=4 → ver modelo no início dessa página)
```

OPERADOR MOD (Resto da Divisão)

Recebe dois valores (dividendo e divisor) e calcula o resto da divisão.

Sintaxe Pseudocódigo (Uso do operador MOD)

Dividendo = 9

Divisor = 2

Resto = Dividendo MOD Divisor

→ Teremos que a variável **Resto** recebe o valor (inteiro) restante do cálculo 9/2, ou seja, Resto = 1

Exemplo Linguagem C (Uso de variável tipo inteiro e operador “%”)

```
int A, B, C;
```

```
A = 9;
```

```
B = 2;
```

```
C = A % B; (desse modo, C=1 → ver modelo no início dessa página)
```

OBS: Declaração dessas variáveis como **float** resulta em erro de execução.

FUNÇÕES MATEMÁTICAS

POTENCIAÇÃO - FUNÇÃO POW()

Recebe dois valores (base e expoente) e calcula o resultado dessa exponenciação.

Sintaxe Pseudocódigo:

Base = 3

Expoente = 2

Resultado = POT(Base, Expoente)

Exemplo Linguagem C:

```
int X, Y, Z;
```

```
X=3;
```

```
Y=2;
```

```
Z = pow(X,Y);
```

Teremos que a variável Resultado recebe Base **3** multiplicado **2** vezes (Expoente), ou seja, Resultado = $3 * 3$, sendo Resultado = 9

RADICIAÇÃO (operação inversa da potenciação) - FUNÇÃO SQRT()

Recebe um valor e calcula o resultado de sua raiz quadrada.

Sintaxe Pseudocódigo:

Num = 9

Raiz = RAD(Num)

Exemplo Linguagem C:

```
int X, Y;
```

```
X=9;
```

```
Y=sqrt(X);
```

Teremos que a variável Raiz recebe o cálculo da raiz quadrada de Num, ou seja,

Raiz = 3

OBS: Função **cbrt()** para **raiz cúbica**.

EXERCÍCIOS

- 1) Elabore em Pseudocódigo e em Linguagem C os enunciados descritos a seguir:
 - a) Realizar cálculos de **Exponenciação** de valores digitados pelo usuário.
 - b) Realizar cálculos de **Raiz** de números digitados pelo usuário.

ESTRUTURA CONTROLE - LAÇOS DE REPETIÇÃO – TESTE NO INÍCIO

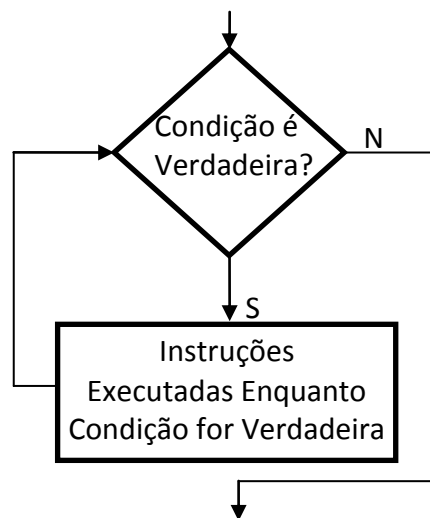
SINTAXE

ENQUANTO <condição verdadeira> **FAÇA**

Instruções para sim

FIM_ ENQUANTO

MODELO DIAGRAMA DE BLOCOS/ FLUXOGRAMA



IMPORTANTE:

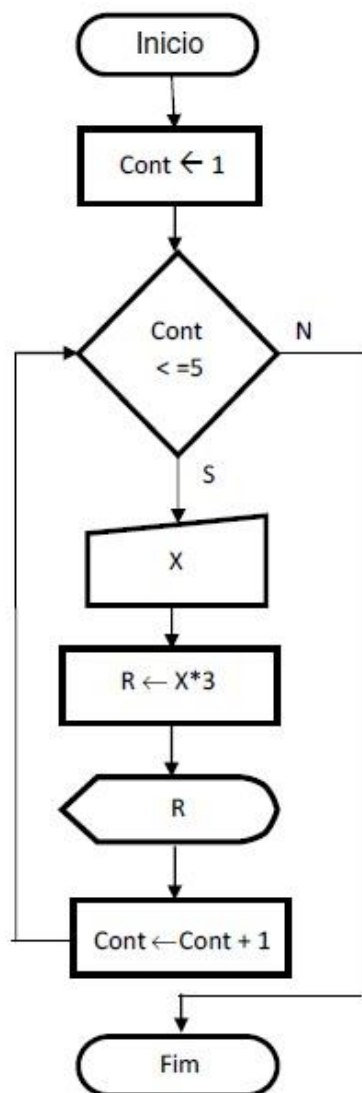
1. O teste de verificação será executado no início;
2. As instruções a seguir serão executadas enquanto a condições for verdadeira;
3. Se de inicio a condição for falsa, as instruções a seguir não serão executadas e o *looping* será finalizado.

Segue exemplo:

- Receber um valor;
- Multiplicá-lo por 3 e apresentar o resultado;
- Repetir o processo 5 vezes.

Análise do problema:

1. Criar uma variável para iniciar o contador;
2. Enquanto o valor do contador for menor ou igual a 5:
 - Ler um número;
 - Multiplicar por 3;
 - Apresentar o resultado;
 - Acrescentar 1 ao contador;
 - Novamente verificar se o contador atingiu o final.

EXEMPLO: FLUXOGRAMA**PORTUGUÊS ESTRUTURADO (PSEUDOCÓDIGO)**

PROGRAMA LOOPING

VAR

X, R, CONT : INTEIRO;

INICIO

CONT ← 1 ;**ENQUANTO (CONT <= 5) FAÇA**

LEIA (X);

R ← X * 3;

ESCREVA (R);

CONT ← CONT + 1;**FIM_ENQUANTO**

FIM

EXEMPLO: LINGUAGEM C

/* loop while */

#include <stdio.h>

main()

{

printf ("Execução de Loop While \n");

int x, r, cont;

cont = 1;**while (cont <= 5)**

{

printf ("\n digite um valor :");

scanf ("%d", &x);

r= x*3;

printf ("\n O triplo do valor é : %d
\n\n", r);**cont = cont + 1; // ou usar cont++;**

}

system ("pause");

}

EXERCÍCIOS → PSEUDOCÓDIGO / FLUXOGRAMA / LINGUAGEM C**SALA DE AULA + LABORATÓRIO**

- (1) Apresente o seu nome 10 vezes na tela
- (2) Apresente na tela os números entre 0 e 20
- (3) Apresente apenas os números pares entre 0 e 100 (ordem crescente)
- (4) Apresente apenas os números ímpares entre 0 e 100 (ordem decrescente)
- (5) Ler valores para A, B e C → Calcular e Apresentar a Média → Repetir o processo 3 vezes

EXEMPLOS WHILE DE VERIFICAÇÃO - PROCESSO CONTINUA OU NÃO?

```
#include <stdio.h>
main()
{
    int A, B, S, resp;
    resp = 1;
    while (resp==1)
    {
        printf("\n\nInforme o valor de A: ");
        scanf("%d", &A);
        printf("Informe o valor de B: ");
        scanf("%d", &B);
        S = A + B;
        printf("\nResultado: %d", S);
        printf("\nDeseja continuar? (1=SIM
                                     / 2=NAO): ");
        scanf("%d",&resp);
    }
    system("pause");
}
```

```
#include <stdio.h>
main()
{
    int A, B, S;
    char resp;
    resp = 'S'; // aspas simples para
                tipo char

    while (resp=='S')

        // Também pode-se prever digitação
        // de SIM com S maiúsculo 'S' OU
        // minúsculo 's':

        // while (resp=='S' || resp=='s')

        {
            printf("\n\nInforme o valor de A: ");
            scanf("%d", &A);
            printf("Informe o valor de B: ");
            scanf("%d", &B);
            S = A + B;
            printf("\nResultado: %d", S);
            printf("\nDeseja continuar? (S/N): ");
            scanf(" %c",&resp); //espaço antes %c
        }
    system("pause");
}
```

EXEMPLO WHILE DE VERIFICAÇÃO - PROCESSO CONTINUA OU NÃO?

```
#include <stdio.h>
main()
{
    int n1,n2,soma,resp;
    resp = 1;
    while(resp==1)
    {
        printf("\n\nSOMA - LOOP USUARIO\n");
        printf("\nDigite Primeiro Valor Inteiro: ");
        scanf("%d",&n1);
        printf("Digite Segundo Valor Inteiro: ");
        scanf("%d",&n2);
        soma = n1 + n2;
        printf("\nSOMA=> %d\n",soma);
        printf("\nDeseja continuar? (1=SIM / 2=NAO): ");
        scanf("%d",&resp);

        while((resp!=1) && (resp!=2)) // != Corresponde a Diferente e && Operador E
        {
            printf("\n\n ==>>> ATENCAO: Digite 1 para SIM ou 2 para NAO\n\n");
            scanf("%d",&resp);
        }
    }
    system("pause");
}
```

EXEMPLO DO WHILE DE VERIFICAÇÃO - PROCESSO CONTINUA OU NÃO?**EXEMPLO 1 – Definir continua SOMA**

```
#include <stdio.h>
main()
{
    int n1,n2,soma,resposta;
    do{
        system("cls"); // Apaga a tela
        printf("Digite Primeiro Valor: ");
        scanf("%d",&n1);
        printf("Digite Segundo Valor: ");
        scanf("%d",&n2);
        soma=n1+n2;
        printf("Soma: %d\n",soma);
        printf("Deseja continuar?
            (1 - SIM / 2 - NÃO)\n");
        scanf("%d",&resposta);
    } while(resposta==1);
    system("pause");
}
```

EXEMPLO 2 – Definir se n° é par ou ímpar

```
#include <stdio.h>
main()
{
    int n;
    do {
        printf("Digite um numero para
            saber se e Par ou Impar ou 0
            (zero) para sair: ");
        scanf("%d", &n);
        if( n%2 == 0 ) // Se o resto n/2 = 0
            printf("%d e Par\n", n);
        else
            printf("%d e Impar\n", n);
    } while( n != 0 ); // Enquanto n≠0
    system("pause");
}
```

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 10

Parte 1

REVISÃO AV1 – LABORATÓRIO E SALA DE AULA

(1) Faça o teste de mesa para o pseudocódigo a seguir e informe todos os valores para as variáveis X, Y e Z

ALGORITMO TESTE
 VAR
 X, Y, Z: INTEIRO;
 INICIO
 $X \leftarrow 1$;
 $Y \leftarrow 2$;
 $Z \leftarrow Y - X$;
 $X \leftarrow 5$;
 $Y \leftarrow X + Z$;
 $Z \leftarrow X * Y$;
 FIM

(2) Sejam A=2, B=3, C=4 responda Verdadeiro ou Falso

($A < > C$) AND ($A + 1 = B$) ()
 ($B = C$) AND ($C > = A$) ()
 ($A * B > C$) OR ($C < A$) ()
 ($C = B$) OR ($A > C$) ()
 NOT ($B - 1 = A$) ()
 NOT ($C < A$) ()

**** Para os enunciados 3, 4 e 5 elabore algoritmos em:**

- PSEUDOCÓDIGO (Português Estruturado)
- FLUXOGRAMA (Diagrama de Blocos)
- LINGUAGEM C

(3) FAÇA O CÁLCULO DA ÁREA DE CIRCUNFERÊNCIA ATRAVÉS DA FÓRMULA DADA POR πR^2

(4) EXERCÍCIOS COM INSTRUÇÃO SE / COMANDO IF

- (a) Elaborar um programa que informe se o número é Positivo ou Negativo
 (b) Ler 2 números inteiros e apresentá-los em Ordem Crescente e em Ordem Decrescente

(5) EXERCÍCIOS COM INSTRUÇÃO ENQUANTO / COMANDO WHILE
 (Fazer também Teste de Mesa para 3 ocorrências)

- (a) Apresentar o quadrado de todos os números inteiros de 15 a 200
 (b) Apresentar a soma de todos os ímpares de 0 a 10

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 12

CONCEITOS – String e Formatação

Uma **sequência de caracteres** é chamada de **STRING**.

Quando se declara uma variável do tipo **char** na Linguagem C, é possível utilizar a variável para:

- (i) armazenar apenas **um único caractere**;
- (ii) armazenar mais de um caractere, ou seja, uma **string**. Desse modo, é necessário pré-determinar um tamanho previsto para a variável, reservando assim, um espaço na memória para ela.

Exemplos:

char OP Variável OP que armazenará um único caractere a Opção: (1, A, S, +, etc.)

char NOME[30] Variável NOME que armazenará uma string de até 30 caracteres

Nesses casos, utiliza-se a **formatação** conforme descrito a seguir:

char OP **%c** (caractere único)

char NOME[30] **%s** (string = cadeia ou sequência de caracteres)

Função gets() – Digitação de textos (strings)

Para se ler valores para variáveis do tipo string utiliza-se a função **gets()** que lê caracteres, números e espaços em branco até que a tecla ENTER seja pressionada.

Desse modo, não se utiliza a **função gets** para leitura de um único caractere esperado (**Ex.:** OP).

EXEMPLO

```
#include <stdio.h>
main ()
{
char nome[30], endereco[20];
printf("Digite seu Nome Completo: ");
gets(nome);
printf("Digite seu Endereco: ");
gets(endereco);
printf("Saudacoes, %s \nVoce mora no Endereco: %s \n\n", nome, endereco);
system ("pause"); }
```

NOTA:

Não é possível utilizar a função scanf ("%s", &nome) para digitação de cadeias de caracteres com espaçamento em branco, pois qualquer espaço entre as palavras (Nome e Sobrenome) digitado pelo usuário sinalizará para a função scanf() que a entrada de dados terminou. Assim, será armazenada na variável somente a primeira parte digitada antes do espaço, ou seja, apenas o Nome sem o Sobrenome.

ESTRUTURA CONTROLE - LAÇOS DE REPETIÇÃO – TESTE NO FINAL

SINTAXE: **REPITA** ...*instruções*... **ATÉ QUE** <*condição verdadeira*>

- (a) O teste de verificação é executado no final do *looping*/laço;
- (b) Instruções depois do REPITA e antes ATÉ QUE são executadas até que condição verdadeira;
- (c) Portanto, as instruções são executadas enquanto a resposta do teste for FALSA;
- (d) Todas as instruções contidas no laço serão executadas pelo menos uma vez, pois elas aparecem antes do teste de verificação.

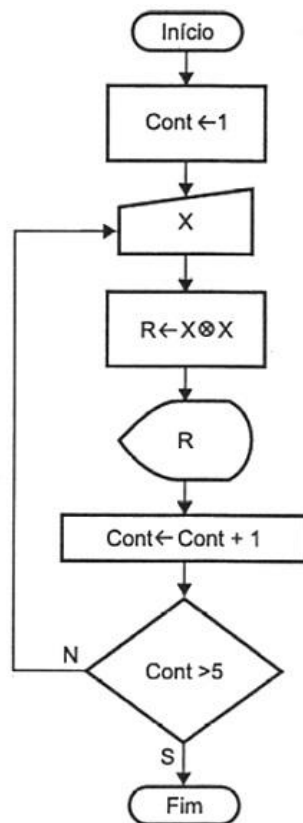
EXEMPLO

- Receber valor
- Calcular o quadrado dele
- Mostrar o resultado
- Repetir esse processo 5 vezes

ANÁLISE DO PROBLEMA

1. Criar a variável **CONTADOR**;
2. Ler um valor para a variável **X**;
3. Calcular o quadrado do valor **X** e armazenar o resultado **R**;
4. Apresentar o resultado de **R**;
5. Incrementar o **CONTADOR**;
6. Repetir os passos 2, 3, 4 e 5 até que o **CONTADOR** seja maior que 5.

OBSERVAÇÃO: É preciso ter cuidado para não reiniciar o contador. Deixe-o fora do laço.

→ **FLUXOGRAMA**→ **PORTUGUÊS ESTRUTURADO**

PROGRAMA LOPPING_2

VAR

X, R, CONT : INTEIRO;

INICIO

CONT ← 1;**REPITA**

LEIA (X);

R ← X * X;

ESCREVA ("QUADRADO É :", R);

CONT ← CONT + 1;**ATÉ QUE (CONT > 5)**

FIM

PSEUDOCÓDIGO (REPITA ATÉ QUE):A repetição continua enquanto a condição for *Falsa*.**EXEMPLO DE USO NA LINGUAGEM DELPHI:**

REPEAT... UNTIL (idem Lógica do Pseudocódigo)

EXEMPLO DE USO NA LINGUAGEM VB:

DO... LOOP UNTIL (idem Lógica do Pseudocódigo)

Comando DO... WHILE - Linguagem C→ *Similar ao REPITA ATÉ QUE – Teste Lógico no Final*

#include <stdio.h>

main()

{

int x, r, cont;

cont = 1;**do { // Observe sequência {entre chaves}**

printf ("Digite um número para saber o quadrado dele: ");

scanf ("%d",&x);

r = x * x;

printf ("O quadrado e: %d \n", r);

cont = cont+1;**} while (cont <= 5);** // Observe aqui o uso do ponto e vírgula

system ("pause");

}

LINGUAGEM C:Assim como no comando **WHILE**, a repetição continua *Enquanto* a condição for *Verdadeira*.**DO ... WHILE <condição verdadeira>****EXERCÍCIOS****Pseudocódigo e Fluxograma: REPITA... ATÉ QUE... Linguagem C: DO... WHILE****(1) Apresentar apenas os números pares de 10 a 20 (ordem crescente)****(2) Apresentar apenas os números ímpares de 30 a 50 (ordem decrescente)**

- (3) Apresentar apenas os múltiplos de 3 dos números de 0 a 50
- (4) Ler A, B e C / Calcular e Apresentar a Média / Executar esse processo 5x
- (5) Apresentar os quadrados de todos os números inteiros de 15 a 31
- 15 = 225 ... 31 = 961

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 13

ESTRUTURA DE REPETIÇÃO

SINTAXE: PARA *variável* DE *início* ATÉ *fim* PASSO *incremento* FAÇA
 < laço contendo instruções a serem repetidas >
 FIM PARA

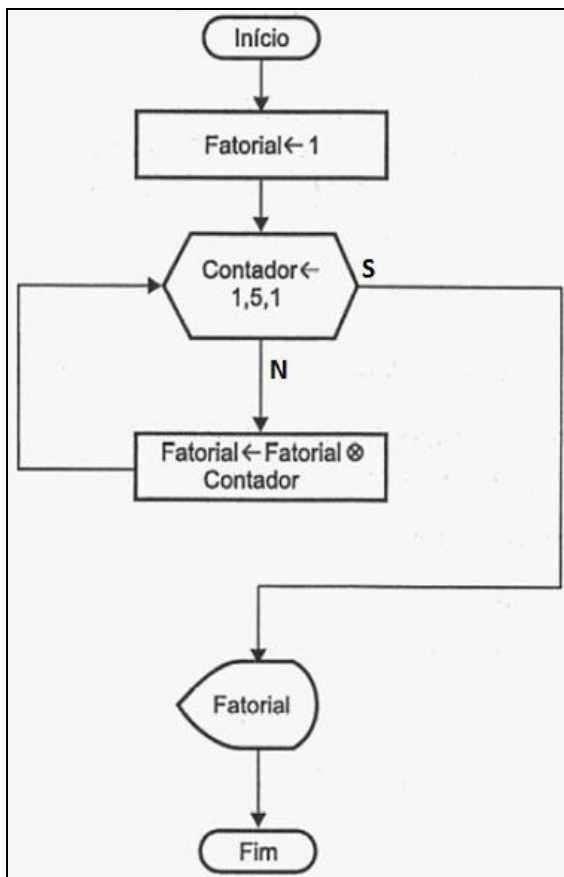
- (a) Será executado o conjunto de instruções entre a instrução **PARA** e a instrução **FIM PARA**.
- (b) Essa estrutura de repetição poderá ser utilizada todas as vezes que houver a necessidade de repetir trechos finitos, em que se conhecem os valores inicial e final dessa repetição.

EXEMPLO → Cálculo FATORIAL → $5! = \frac{5 \times 4 \times 3 \times 2 \times 1}{\text{uso do FOR}} = 120$

OBS: Para o cálculo FATORIAL utiliza-se o conceito de RECURSIVIDADE, pois a mesma variável é reutilizada para próximos cálculos.

ANÁLISE DO PROBLEMA

1. Criar uma variável de controle **CONTADOR**;
2. Criar uma variável para controle do cálculo **FATORIAL** que apresentará o resultado;
3. Definir **INICIO**, **FIM** e **PASSO** para a variável **CONTADOR**;
4. Armazenar na variável **FATORIAL** o valor atual dela multiplicado pela variável de controle **CONTADOR**;
5. Apresentar o resultado final do **FATORIAL**.

FLUXOGRAMA**PORTUGUÊS ESTRUTURADO**

PROGRAMA FATORIAL

VAR

CONTADOR, FATORIAL : INTEIRO;

INICIO

FATORIAL ← 1;**PARA CONTADOR DE 1 ATÉ 5 PASSO 1 FAÇA**

FATORIAL ← FATORIAL * CONTADOR;

FIM PARA

ESCREVA ("Fatorial de 5 é =", FATORIAL);

FIM

LINGUAGEM C

#include <stdio.h>

main()

{

int cont, fat ;

fat = 1;

for (cont=1 ; cont <=5 ; cont++)

{

fat = fat * cont;

}

printf ("Fatorial de 5 = %d \n", fat);

system ("pause");

}

EXEMPLO - PROGRAMA para apresentar na tela a TABELA ASCII

#include <stdio.h>

main()

{

int i;

system("cls"); // CLS = CLear Screen = Limpar a tela - Apaga os dados que estiverem na tela

for (i=0; i < 256; i++)

printf("%d = %c \n", i, i);

// OBS: Variável **i** será apresentada com formato Número (**%d**) e Caractere (**%c**)

system("pause");

}

EXEMPLO - SEQUÊNCIA DE FIBONACCI: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

main()

{

int a=0, b=0, c=0, cont;

a=1;

for (cont=1; cont<=15;cont++)

{

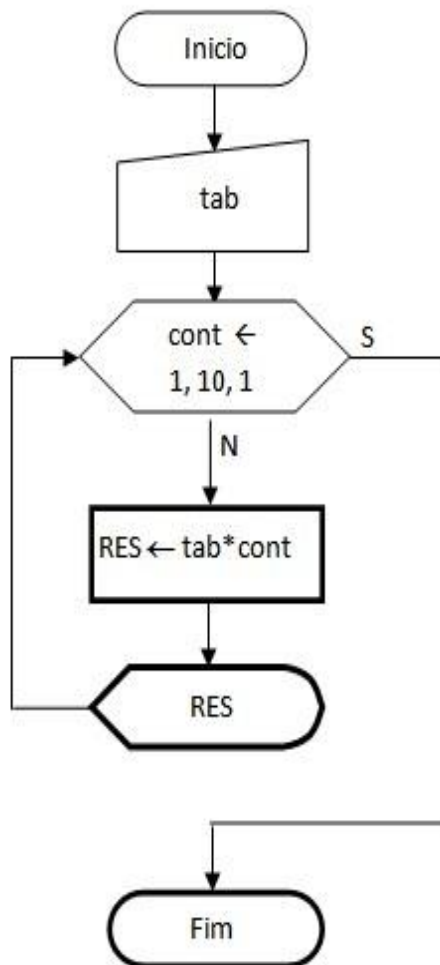
c=a+b;

printf("%d\n", c);

a=b;

b=c; }

system("pause"); }

EXEMPLO 2 → TABUADA DE UM NÚMERO DIGITADO PELO USUÁRIO**FLUXOGRAMA****PSEUDOCÓDIGO**

PROGRAMA TABUADA

VAR

TAB, RES, CONT: INTEIRO;

INICIO

ESCREVA ("Digite um número: ");

LEIA (TAB);

PARA CONT DE 1 ATÉ 10 PASSO 1 FAÇA

RES ← TAB * CONT;

ESCREVA (RES);

FIM_PARA

FIM

LINGUAGEM C

#include <stdio.h>

main()

{

int tab, cont, res;

printf ("Digite um numero:");

scanf("%d", &tab);

for (cont=1; cont <= 10; cont++)

{

res = tab*cont;

printf ("%d x %d = %d \n", tab, cont, res);

}

system("pause");

}

EXERCÍCIOS – Pseudocódigo, Fluxograma e Linguagem C

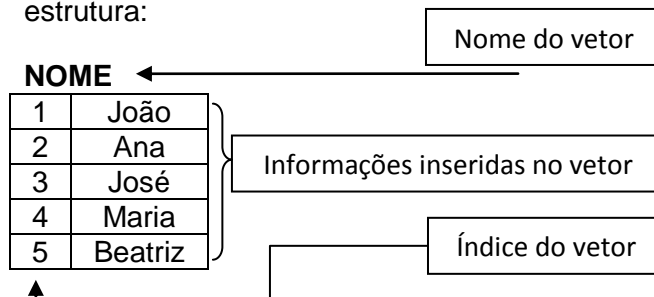
- (1) Apresente apenas os números pares de 80 a 200 (ordem crescente)
- (2) Apresente apenas os números ímpares de 0 a 20 (ordem decrescente)
- (3) Apresente apenas os múltiplos de 5 dos números de 0 a 100
- (4) Apresentar a Soma de todos os números ímpares de 0 a 10

VETORES

Podemos definir um **VETOR** como **UMA VARIÁVEL** dividida em partes. Cada parte pode receber uma informação diferente. Evidentemente, todas elas têm o mesmo tipo de dados (real, inteiro, caractere, etc.).

As partes de um vetor são identificadas por um número (índice) que se refere a posição (localização) de cada informação inserida no vetor. **Sintaxe:** **VETOR [ÍNDICE]** - **Exemplo:** **NOME [5]**

Assim, um vetor que armazene 5 nomes pode ser visualizado com a seguinte estrutura:



SINTAXE EM PSEUDOCÓDIGOS

NUM: CONJUNTO [0..9] DE INTEIROS

Vetor NUM com 10 posições (índices) tipo inteiro

NOME: CONJUNTO [0..49] DE CARACTERES [30]

Vetor NOME com 50 posições (índices) tipo caractere de tamanho 30 cada.

EXEMPLOS LINGUAGEM C

int NUM[10];

Vetor NUM com 10 posições (índices) declarado com tipo de dados inteiro

char NOME [50] [30];

Vetor NOME tipo caractere. O primeiro número indica a quantidade de posições (índices) e o segundo número indica o tamanho de cada uma dessas posições.

OBS: O primeiro elemento de um vetor é colocado no índice 0. Assim, para 10 elementos considera-se um vetor de 0 a 9 = 10 posições (índices) para os elementos.

EXEMPLO 1: VETOR COM 5 POSIÇÕES - INSERE E APRESENTA INFORMAÇÕES

PROGRAMA VALORES

VAR

VALOR: Conjunto [0..4] de Inteiro;

INICIO

VALOR[0] ← 7;

VALOR[1] ← 4;

VALOR[2] ← 9;

VALOR[3] ← 5;

VALOR[4] ← 2;

ESCREVA("Posicao 0 do vetor é ", VALOR[0]);

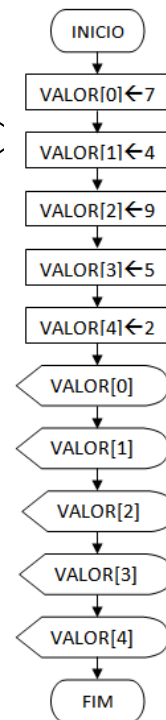
ESCREVA("Posicao 1 do vetor é ", VALOR[1]);

ESCREVA("Posicao 2 do vetor é ", VALOR[2]);

ESCREVA("Posicao 3 do vetor é ", VALOR[3]);

ESCREVA("Posicao 4 do vetor é ", VALOR[4]);

FIM

**LINGUAGEM C**

#include <stdio.h>

main()

{

int VALOR[5];

// declarando o vetor VALOR - **OBS: de 0 a 4 = 5 índices – A primeira posição de um vetor tem índice 0**

VALOR[0]=7; // inserindo informações no vetor VALOR

VALOR[1]=4;

VALOR[2]=9;

VALOR[3]=5;

VALOR[4]=2;

// Também poderia declarar e inserir informações diretamente no vetor VALOR como a seguir:

// int VALOR[5] = { 7, 4, 9, 5, 2 };

printf("Informacao na Posicao 0 do vetor = %d \n", VALOR[0]);

// apresentando informações no vetor VALOR

printf("Informacao na Posicao 1 do vetor = %d \n", VALOR[1]);

printf("Informacao na Posicao 2 do vetor = %d \n", VALOR[2]);

printf("Informacao na Posicao 3 do vetor = %d \n", VALOR[3]);

printf("Informacao na Posicao 4 do vetor = %d \n", VALOR[4]);

system("pause");

}

EXEMPLO 2: VETOR COM 5 POSIÇÕES - LAÇO FOR – INSERE E EXIBE TEXTO

PROGRAMA TEXTOS

VAR

i: Inteiro;

TEXTO: Conjunto [0..4] de
Caracteres[10];

INICIO

TEXTO [0] ← "BALA";

TEXTO [1] ← "COMUNICAR";

TEXTO [2] ← "DADO";

TEXTO [3] ← "ELEFANTE";

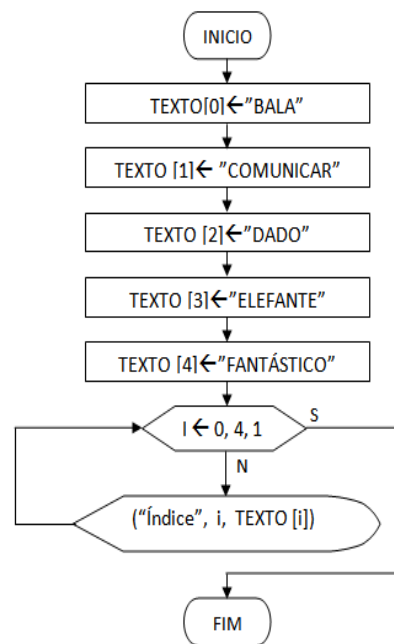
TEXTO [4] ← "FANTASTICO";

PARA i DE 0 ATÉ 4 PASSO 1 FAÇA

ESCREVA("Índice", i, TEXTO [i]);

FIM_PARA

FIM



#include <stdio.h>

main()

{

int i; // declaração da variável i que será o índice do vetor INFO

char TEXTO[5][10] = { "BALA", "COMUNICAR", "DADO", "ELEFANTE", "FANTASTICO" };

for(i = 0 ; i <= 4; i++) // **OBS: de 0 a 4 = 5 índices - Primeira posição do vetor tem índice 0**

{

printf("Informacao na Posicao %d do vetor = %s \n" , i , TEXTO [i]);

// apresentando informações do vetor TEXTO

}

system("pause");

}

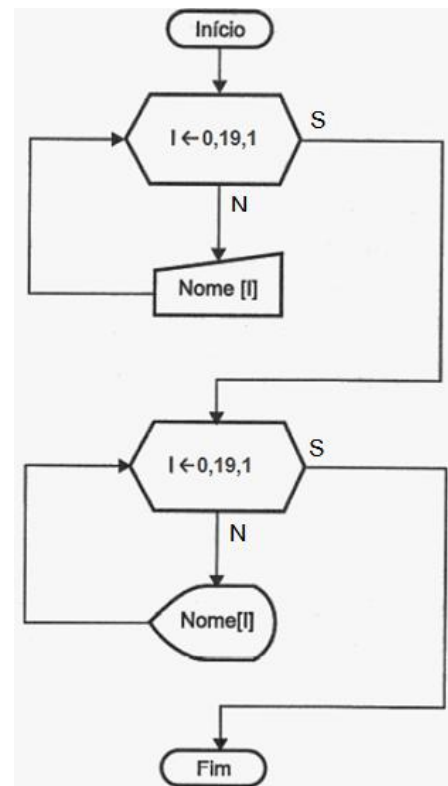
EXEMPLO 3: VETOR COM 20 NOMES DE TAMANHO 30 CADA. PREENCHE E EXIBE

```

PROGRAMA LISTA_NOME
VAR
  NOME: Conjunto [0..19] de Caracteres [30];
  i: Inteiro;
INICIO
  PARA i de 0 até 19 passo 1 faça
    LEIA (NOME[i]);
  FIM_PARA

  PARA i de 0 até 19 passo 1 faça
    ESCREVA (NOME[i]);
  FIM_PARA
FIM

```



```

#include <stdio.h>
main()
{
  char NOME[20][30]; // Vetor NOME - 20 posições sendo cada posição com tamanho 30
  int X;

  for(X=0; X<=19; X++)
  {
    printf("\nDigite Nomes [%d]:", X);
    gets(NOME[X]);
  }

  for(X=0; X<=19; X++)
  {
    printf("\n");
    printf("\t Nome  %s \n", NOME[X]);
  }

  printf("\n\n");
  system("pause");
}

```


EXEMPLO 4: VETOR COM NUMEROS PARES DE 0 A 20 / PREENCHE E EXIBE

PROGRAMA LISTA_PARES

VAR

NUM: Conjunto [0..10] de Inteiro;

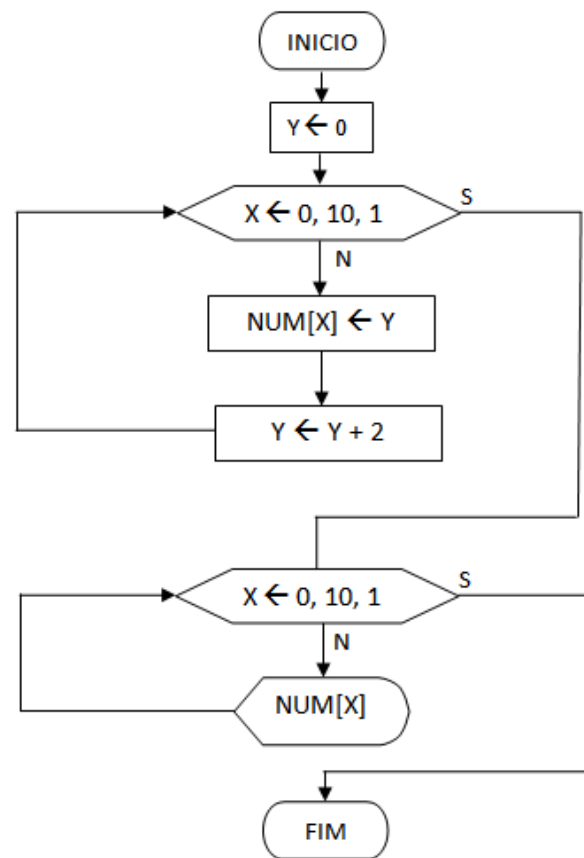
X, Y: Inteiro;

INICIO

Y ← 0;

PARA X de 0 até 10 passo 1 faça**NUM[X] ← Y;****Y ← Y+2;****FIM_PARA****PARA X de 0 até 10 passo 1 faça****ESCREVA (NUM[X]);****FIM_PARA**

FIM



#include <stdio.h>

main()

{

int NUM[11], X, Y=0; // Declaração do vetor VET de tamanho 11 (Pares de 0 a 20)

// Declaração da variável X para o índice do vetor

// Declaração da variável Y inicializada com valor =0 para calcular os números pares

for(X=0; X<=10; X++)

{

NUM[X] = Y; // Preenchimento do vetor VET com valores da variável Y (0 a 20)**Y = Y+2;****}****for(X=0; X<=10; X++)**

{

printf(" %d ", NUM[X]); // Apresentando os valores pares: 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20.**printf("\n\n");****}**

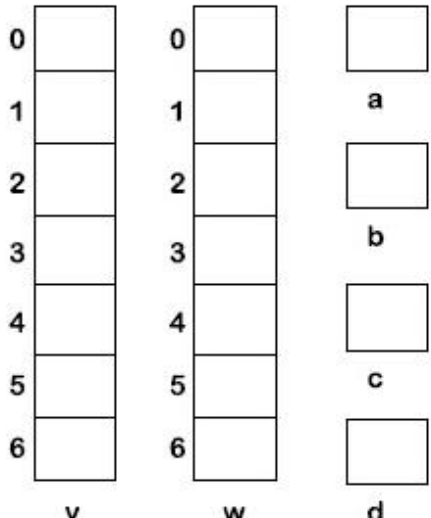
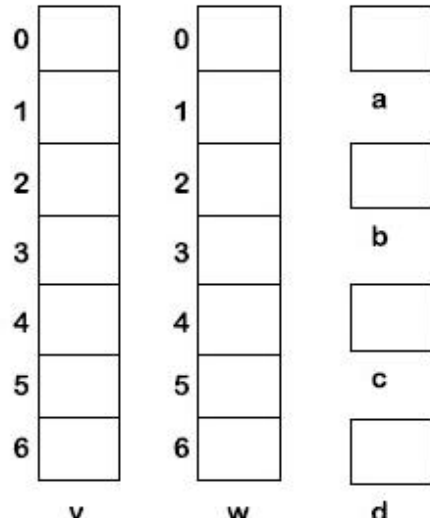
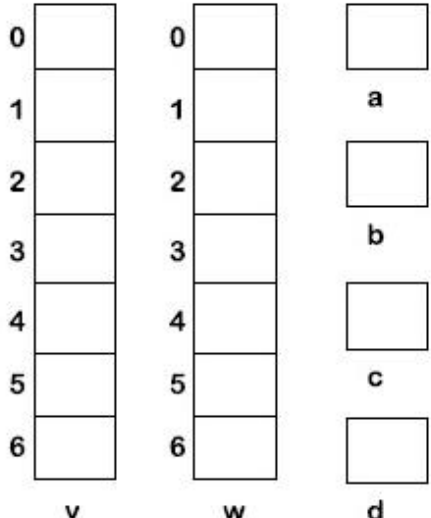
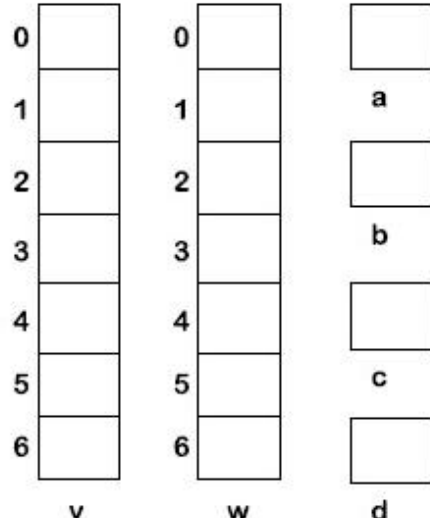
system("pause");

}

EXERCÍCIOS - TESTE DE MESA → VETORES

Preencha os quadros informando os valores finais das variáveis para cada trecho de código a seguir. Considere as variáveis inicialmente como vazias.

<p>(1)</p> <pre> a = 10; b = 20; c = (a + b) / 2; c = c - 40; v[4] = a + b + c; </pre>	<p>(2)</p> <pre> d = 43; a = d%8; v[0] = a + d; v[1] = d * 0.1; </pre>
<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: left;"> 0 1 2 3 4 5 6 v </div> <div style="text-align: left;"> 0 1 2 3 4 5 6 w </div> <div style="text-align: left;"> <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> a <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> b <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> c <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> d </div> </div>	<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: left;"> 0 1 2 3 4 5 6 v </div> <div style="text-align: left;"> 0 1 2 3 4 5 6 w </div> <div style="text-align: left;"> <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> a <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> b <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> c <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> d </div> </div>
<p>(3)</p> <pre> a = 2; while (a < 6) { v[a] = 10 * a; a = a + 1; } </pre>	<p>(4)</p> <pre> a = 7; b = a - 6; while (b < a) { v[b] = b * a; b = b + 2; } </pre>
<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: left;"> 0 1 2 3 4 5 6 v </div> <div style="text-align: left;"> 0 1 2 3 4 5 6 w </div> <div style="text-align: left;"> <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> a <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> b <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> c <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> d </div> </div>	<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: left;"> 0 1 2 3 4 5 6 v </div> <div style="text-align: left;"> 0 1 2 3 4 5 6 w </div> <div style="text-align: left;"> <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> a <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> b <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> c <div style="border: 1px solid black; width: 50px; height: 30px; margin-bottom: 5px;"></div> d </div> </div>

<p>(5)</p> <pre>for (a=0; a<3; a++) { v[a] = 5; w[a] = a; }</pre>	<p>(6)</p> <pre>a = 2; b = 5; for (c=0; c<3; c++) { v[c] = a; w[c] = c * v[c]; }</pre>
	
<p>(7)</p> <pre>d = 0; for (a=0; a<5; a++) { w[a] = d; d = d + a; }</pre>	<p>(8)</p> <pre>v[0] = 2; for (d=1; d<4; d++) v[d] = v[d-1] * 2; for (d=0; d<4; d++) w[d] = v[d] * 10;</pre>
	

EXERCÍCIOS – Pseudocódigo, Fluxograma e Linguagem C

- 1) Implementar um programa que preencha um vetor com múltiplos de 5 de 0 a 50 (ordem decrescente) e apresente-os em seguida.
- 2) Implementar um programa que preencha um vetor B resultante do vetor A conforme descritos a seguir:
 A = 10 20 30
 B = 100 200 300
- 3) Implementar um programa que preencha um vetor com 10 Notas digitadas pelo usuário. Apresente-as em seguida → usar função *scanf()*.

ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 14

Algoritmos e Lógica de Programação - REVISÃO AV2 (Parte1) – Pseudo e Fluxo

(1) EXERCÍCIO COM INSTRUÇÃO SE

- Ler um número inteiro e informar se ele é positivo ou negativo

(2) EXERCÍCIOS COM INSTRUÇÃO ENQUANTO *(repetição com verificação no início)*

- (a) Apresentar números pares entre 50 e 100 (ordem crescente)
- (b) Apresentar números ímpares entre 50 e 100 (ordem decrescente)

(3) EXERCÍCIOS COM INSTRUÇÃO REPITA ATÉ QUE *(repetição com verificação no final)*

- (a) Apresentar números pares entre 50 e 100 (ordem crescente)
- (b) Apresentar números ímpares entre 50 e 100 (ordem decrescente)

(4) EXERCÍCIOS COM INSTRUÇÃO PARA *(repetição com variável de controle)*

- (a) Apresentar números pares entre 50 e 100 (ordem crescente)
- (b) Apresentar números ímpares entre 50 e 100 (ordem decrescente)

Algoritmos e Lógica de Programação - REVISÃO AV2 (Parte2)

→ SALA DE AULA: *Pseudocódigo e Fluxograma (ENTREGAR do 1 ao 4)*
 → LABORATÓRIO: *Linguagem C*

(1) EXERCÍCIO – INSTRUÇÃO: SE ENTÃO SENÃO FIM SE → COMANDO IF

- Ler 2 números inteiros – Dependendo do números lidos, apresentar a mensagem para o usuário: >> A é maior que B ou >> B é maior que A

(2) EXERCÍCIO – INSTRUÇÃO: ENQUANTO FIM ENQUANTO → COMANDO WHILE *(repetição com verificação no início)*

- Ler Nome do Produto, Preço e Desconto
 - Calcular e Apresentar o Valor a Pagar

Executar esse processo 3 vezes

(3) EXERCÍCIO – INSTRUÇÃO: REPITA ATÉ QUE → COMANDO DO WHILE (*repetições - similares - verificação no final*)

- Ler valores reais para A, B e C
 - Calcular e Apresentar a Média
- } *Executar esse processo 5 vezes*

(4) EXERCÍCIO – INSTRUÇÃO: PARA FIM PARA → COMANDO FOR (*repetição com variável de controle*)

- (a) Apresentar todos os números de 0 e 9 (decrescente)
- (b) Altere o programa para que apresente, no final, a Soma entre esses números
(OBS: O resultado da Soma será 45)

(5) TESTE DE MESA – VETOR

- a) Faça Pseudocódigo e Fluxograma do código em Linguagem C descrito abaixo
→ *Siga os exemplos Aula FOR*
- b) Execute o teste de mesa do programa abaixo que preenche um vetor com os **múltiplos de 3** de 0 a 15 e que, em seguida, os apresenta em ordem crescente e decrescente.

```
#include <stdio.h>
main()
{
    int VET[6], X, Y=0;
    // Declaração do vetor VET de tamanho 6 (conterá 6 posições, ou seja, 6 índices para 6 informações)

    for(X=0; X<=5; X++)
        // Preenchimento do vetor de índice X (0 a 5) em que cada índice receberá os valores da variável Y (múltiplos de 3)
        {
            VET[X]=Y;
            Y=Y+3;
        }

    for(X=0; X<=5; X++)
    {
        printf(" %d ", VET[X]);
        // Apresentando Múltiplos de 3 (crescente) – Insira um espaço após %d para os valores ficarem lado a lado
    }

    printf("\n\n\n");

    for(X=5; X>=0; X--)
    {
        printf(" %d ", VET[X]);
        // Apresentando Múltiplos de 3 (decrescente) – Insira um espaço após %d para os valores ficarem lado a lado
    }

    printf("\n\n\n");
    system("pause"); }
```

(6) EXERCÍCIO – VETOR → (Utilize o modelo anterior) → Pseudocódigo, Fluxograma e Linguagem C

→ Elaborar um programa que preencha um vetor com **múltiplos de 10** de 0 a 100 e em seguida apresente-os em ordem crescente e decrescente.

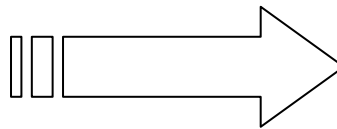
ALGORITMOS E LÓGICA DE PROGRAMAÇÃO – AULA 17

ESTRUTURA DE CONTROLE – DECISÃO - MÚLTIPLA ESCOLHA

- ★ O comando **SWITCH CASE** é similar ao comando **IF** para escolha de variáveis no direcionamento das instruções.
- ★ Entretanto, quando se tem várias tomadas de decisão, o comando **SWITCH CASE** possui maior flexibilidade usando um formato limpo e claro.
- ★ O comando **break** é colocado no final de cada opção definida na Estrutura de Controle do comando Switch Case encerrando-o quando qualquer dessas opções for selecionada pelo usuário.
- ★ O comando **default** é executado caso nenhuma das opções definidas na Estrutura de Controle do comando Switch Case tenha sido selecionada pelo usuário.

SINTAXE PSEUDOCÓDIGO:

```
VAR OPCAO: CHAR;
ESCOLHA (OPCAO)
    CASO '1':
        INSTRUÇÃO 1;
    CASO '2':
        INSTRUÇÃO 2;
    CASO '3':
        INSTRUÇÃO 3;
    CASO CONTRÁRIO : INSTRUÇÃO 4;
FIM_ESCOLHA
```



SINTAXE EM LINGUAGEM C:

```
main()
{
    int A;
    ...
    ...
    ...

    printf ("Digite a Opção:");
    scanf ("%d", &A);

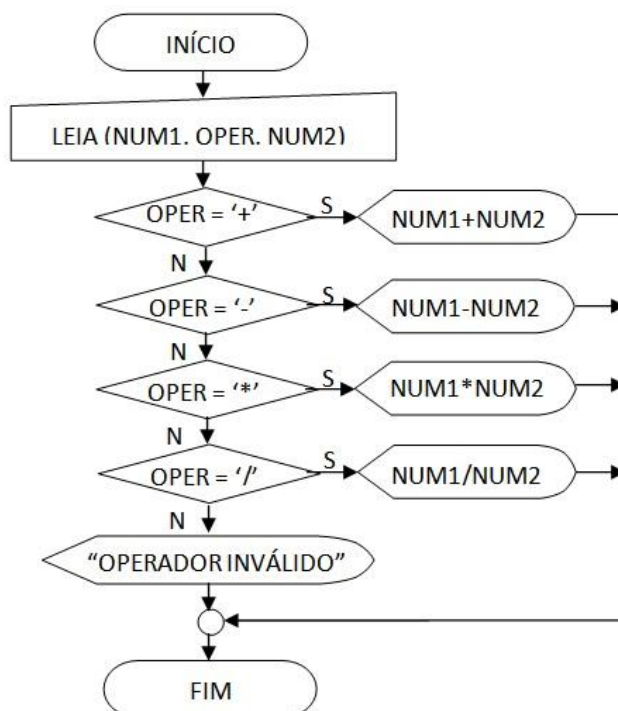
    switch(A)
    {

        case 1:
            .....
            break;
        case 2:
            .....
            break;
        case 3:
            .....
            break;

        default:
            .....

    }
}
```

EXEMPLO FLUXOGRAMA → CALCULADORA



EXEMPLO SWITCH CASE - CALCULADORA

PORTUGUÊS ESTRUTURADO

```
PROGRAMA CALCULADORA
VAR  NUM1, NUM2 : REAL;
     OPER : CHAR;
INICIO
    ESCREVA ( "DIGITE VALORES PARA
              NUMERO 1, OPERADOR, NUMERO2" );
    LEIA (NUM1, OPER, NUM2);
    ESCOLHA (OPER)
        CASO '+' :
            ESCREVA (NUM1+NUM2);
        CASO '-' :
            ESCREVA (NUM1-NUM2);
        CASO '*' :
            ESCREVA (NUM1*NUM2);
        CASO '/' :
            ESCREVA (NUM1/NUM2);
        CASO CONTRÁRIO: ESCREVA
            ("OPERADOR INVALIDO");
    FIM_ESCOLHA
FIM
```

EXEMPLO LINGUAGEM C

```
#include <stdio.h>
main()
{
    float num1, num2;
    char oper;
    printf ("Digite==> Primeiro Numero,
            um Operador e o Segundo Numero: ");
    scanf ( "%f %c %f", &num1, &oper, &num2);
    switch(oper)
    {
        case '+':
            printf ("Soma = %.1f \n\n", num1 + num2);
            break;
        case '-':
            printf ("Subtração = %.1f \n\n", num1- num2);
            break;
        case '*':
            printf ("Multiplicação = %.1f \n\n", num1* num2);
            break ;
        case '/':
            printf ("Divisão = %.1f \n\n", num1/num2);
            break;
        default:
            printf ("Operador Invalido \n\n");
    }
}
```

EXEMPLOS EM LINGUAGEM C

```
#include <stdio.h>
main( )    // Programa menciona o estado civil de acordo com a opção selecionada
{
    int num;

    printf("INFORME O ESTADO CIVIL \n\n1 - Solteiro ou 2 - Casado\n\n");
    scanf("%d", &num);

    switch(num)
    {
        case 1: printf("\nVoce informou ser Solteiro\n\n"); break;
        case 2: printf("\nVoce informou ser Casado\n\n"); break;
        default: printf("\n Opcao incorreta \n\n");break;
    }

    system("pause");

}
```

```
main()    // De acordo com a escolha calcula soma ou media entre 2 valores digitados
{
    int entrada;
    float num1, num2;
    printf ("Digite primeiro numero: ");
    scanf ("%f", &num1);
    printf ("Digite segundo numero: ");
    scanf ("%f", &num2);
    printf("\n\nDigite 1 para calcular a soma de dois numeros\n");
    printf("\nDigite 2 para calcular a media\n\n");
    scanf("%d",&entrada);

    switch(entrada){
        case 1:
            printf("\nVc escolheu a opcao de somar dois numeros\n");
            printf("\nA soma entre %.1f e %.1f = %.1f\n\n", num1, num2, num1+num2);
            break;
        case 2:
            printf("\nVc escolheu a opcao de calcular a media\n");
            printf("\nA media entre %.1f e %.1f = %.1f\n\n", num1, num2, (num1+num2)/2);
            break;
        default:
            printf("Nenhuma das opcoes validas foi selecionada\n\n");
            break;
    }

    system ("pause");
}
```



```
#include <stdio.h>
main()
{
// De acordo com o peso digitado para a Terra será informado o peso correspondente ao
planeta selecionado

int planetaEscolhido ;
float pesoNaTerra, gravidadeRelativa, pesoNoPlaneta;
    printf("PLANETAS \n\n");
    printf("1 - Mercurio \n");
    printf("2 - Venus \n");
    printf("3 - Marte \n");
    printf("4 - Jupiter \n");
    printf("5 - Saturno \n");
    printf("6 - Urano \n");
    printf("7 - Netuno \n");

    printf(" \nSelecione o planeta a saber o seu peso nele: ");
    scanf("%d", &planetaEscolhido);

    printf("\nInforme seu peso na Terra: ");
    scanf("%f", &pesoNaTerra);

    switch(planetaEscolhido)
    {
    case 1:
        gravidadeRelativa = 0.376; break;
    case 2:
        gravidadeRelativa = 0.903; break;
    case 3:
        gravidadeRelativa = 0.3083; break;
    case 4:
        gravidadeRelativa = 2.340; break;
    case 5:
        gravidadeRelativa = 1.160; break;
    case 6:
        gravidadeRelativa = 1.150; break;
    case 7:
        gravidadeRelativa = 1.190; break;
    default:
        gravidadeRelativa = 0; break;
    }

    if (gravidadeRelativa != 0)
    {
        pesoNoPlaneta = pesoNaTerra * gravidadeRelativa;
        printf(" \nSeu peso no planeta %d e: %f \n\n", planetaEscolhido,
pesoNoPlaneta);
    }
    else
        printf (" \nErro na escolha do planeta – Selecione de 1 a 7 \n\n" );
    system("pause");
}
```

EXERCÍCIOS → PSEUDOCÓDIGO, FLUXOGRAMA E LINGUAGEM C

1) Selecione uma Opção:

- 1 – Inclusão
- 2 – Consulta
- 3 – Alteração
- 4 – Exclusão
- 5 – Impressão
- 6 – Sair

OBS: Caso o usuário selecione a opção 2 apresentar na tela a seguinte mensagem:

➔ **VOCÊ SELECIONOU A OPÇÃO CONSULTA**

2) Selecione uma Nota:

- A – 10,0 e 9,0
- B – 8,0 e 7,0
- C – 6,0 e 5,0
- D – 4,0 e 3,0
- E – 2,0 e 1,0

OBS: Caso o usuário selecione a opção C apresentar na tela a seguinte mensagem:

➔ **SUA NOTA ESTÁ ENTRE 6,0 E 5,0**

3) Selecione a sigla do estado:

- 1 - RJ
- 2 - RR
- 3 - SE
- 4 - SP
- 5 - RS
- 6 - PE
- 7 - ES

OBS: Caso o usuário selecione a opção 4 apresentar na tela a seguinte mensagem:

➔ **VOCÊ SELECIONOU O ESTADO DE SÃO PAULO**

4) Selecione um adicional para acompanhar o seu pedido:

- 1 – Queijo Extra
- 2 – Batata Palha
- 3 – Milho
- 4 – Molho Especial

OBS: Caso o usuário selecione a opção 1 apresentar na tela a seguinte mensagem:

SEU ADICIONAL SERÁ QUEIJO EXTRA

5) Selecione o tipo da sua senha:

- A – Empréstimos
- B – Pagamentos
- C – Financiamentos

OBS: Caso o usuário selecione a opção C apresentar na tela a seguinte mensagem:

SUA SENHA CORRESPONDE A FINANCIAMENTOS

6) Informe o numero correspondente a um dia da semana:

- 1 – Domingo
- 2 – Segunda-Feira
- 3 – Terça-Feira
- 4 – Quarta-Feira
- 5 – Quinta-Feira
- 6 – Sexta-Feira
- 7 – Sábado

OBS: Caso o usuário selecione a opção 3 apresentar na tela a seguinte mensagem:

VOCÊ INFORMOU O DIA CORRESPONDE À TERÇA-FEIRA

REVISÃO AV3 → Pseudocódigo, Fluxograma e Linguagem C

Elaborar os seguintes enunciados:

(1) Usando Instrução ENQUANTO – Comando WHILE

- (a) Apresentar na tela valores entre 30 e 50 – apenas pares em Ordem Crescente
- (b) Apresentar na tela a soma dos ímpares entre 30 e 50 - Ordem Decrescente

(2) Usando Instrução REPITA ATÉ QUE – Comando DO WHILE

- (a) Apresentar na tela valores entre 0 e 1000 – múltiplos de 3 em Ordem Crescente
- (b) Apresentar na tela soma dos múltiplos de 5 entre 0 e 1000 – Ordem Decrescente

(3) Usando Instrução SE – Comando IF

Ler o Nome do usuário e a sua Nacionalidade (1 - Brasileiro ou 2 – Estrangeiro).
Apresentar na tela: “Sr(a). xxx” e a opção correspondente.

(4) Usando Instrução DESVIO – Comando CASE

Ler o Nome do usuário e a sua Nacionalidade (1 - Brasileiro ou 2 – Estrangeiro).
Apresentar na tela: “Sr(a). xxx” e a opção correspondente.

(5) Usando Instrução PARA – Comando FOR

- (a) Apresentar na tela valores entre 3000 e 5000 – apenas pares em Ordem Crescente
- (b) Apresentar na tela a soma dos ímpares entre 3000 e 5000 – Ordem Decrescente

(6) Usando Comando FOR – Vetores

- (a) Preenche um vetor com os múltiplos de 10 de 0 a 100 e, em seguida, os apresenta em ordem crescente e decrescente.
- (b) Preenche um vetor com os pares de 0 a 10 e, em seguida, os apresenta em ordem crescente e decrescente.
- (c) Preenche um vetor com os ímpares de 0 a 10 e, em seguida, os apresenta em ordem crescente e decrescente.

