# UNIDADE 2 – ESTRUTURAS DE DECISÃO, ESCOLHA E REPETIÇÃO
## MÓDULO 1 – ESTRUTURA DE CONDIÇÃO

01

## 1 - ESTRUTURA DE CONDIÇÃO

Nos módulos anteriores foram apresentados alguns conceitos básicos sobre as estruturas e comandos que são utilizados para construir um algoritmo simples. Entretanto, as possibilidades de construção de algoritmos que temos até o presente momento são bastante limitadas, pois ainda não estamos aptos a tomar decisões durante o tempo de execução do algoritmo, ou até mesmo de classi?car determinados valores de variáveis. Evoluiremos o módulo apresentando algumas estruturas de condição.

• Estrutura de condição se-entao

É utilizada da seguinte forma:

```
se<expressão> então
<comandos>
fim-se
```

A estrutura de condição, que inicia e finaliza com as palavras se e fim-se, contém uma expressão que deverá retornar um valor verdadeiro (V) ou de falso (F), e caso o resultado dessa expressão seja verdadeiro, serão executados os comandos que estão dentro da estrutura. Caso seja falso, a execução do programa ignora os comandos contidos dentro da expressão e continua na linha seguinte da estrutura de condição.

Os comandos são sequência de códigos que serão executados até o fim-se, caso seja verdadeira a expressão.

02

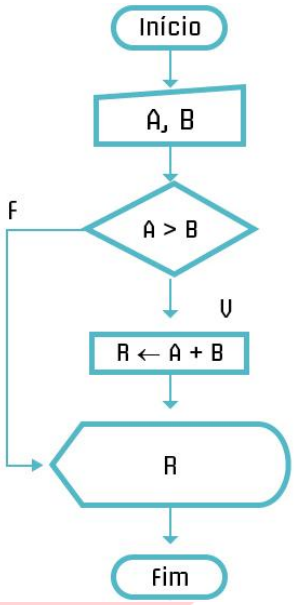
Alguns exemplos de expressões lógicas já foram vistos anteriormente, a seguir temos mais alguns exemplos:

```
Se <18 > 20> então
    Imprima na tela a palavra “maior”
Fim-se
```

O resultado então será falso e não acontecerá nada.

Se <45 = 45> então
    Imprima na tela a palavra “igual”
Fim-se

O resultado será verdadeiro e será impresso a palavra “igual”



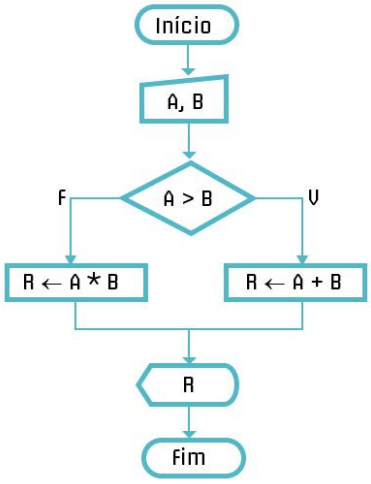

03

• **Estrutura de condição se-entao-senão**

Essa estrutura de condição oferece a possibilidade de executarmos uma determinada ação ou comando se o resultado da expressão for verdadeiro ou falso. Para essas situações é utilizado o comando **senão**, como mostrado nos exemplos abaixo.

```
se<expressão> então
    <comandos>
Senão
    <comandos>
fim-se
```

**Exemplo no fluxograma**




04

• **Estrutura de condição encadeada**

Dentro de uma estrutura **se-então-senão** é perfeitamente possível utilizarmos mais de uma linha de comando, ou até mesmo outras estruturas **se-então-senão**. Existem situações em que os caminhos para a tomada de uma decisão acabam formando uma espécie de árvore com ramificações diversas, onde cada caminho é um conjunto de ações. Nesses casos podemos recorrer à utilização de várias estruturas se-então-senão embutidas umas dentro das outras, comumente chamadas de **ninhos**.

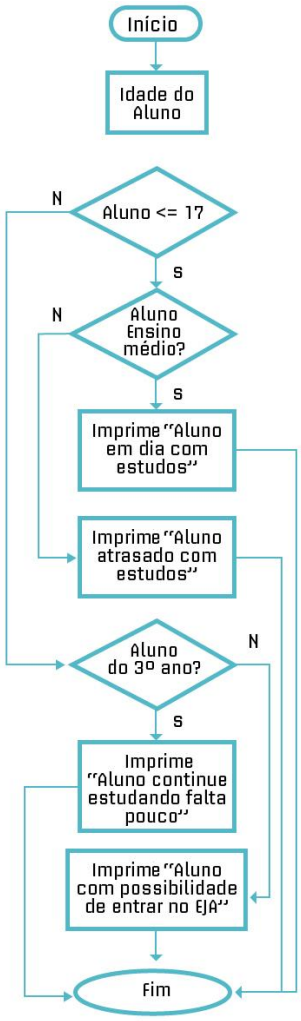
Exemplo:

```
Se Aluno <= 17 então
    seAluno no Ensino Médio então
    imprima “Aluno em dia com estudos”
    senão
    imprima “Aluno atrasado com estudos”
    fim-se
senão
    seAluno no 3º ano do Ensino Médio então
    imprima “Aluno continue estudando falta pouco”
    senão
    imprima “Aluno com possibilidade de entrar do EJA”
    fim-se
fim-se
```

05

Nas estruturas de decisão encadeadas, uma estrutura de condição é aninhada dentro de outra. Para que as condições internas sejam executadas é necessário que seja “Verdadeira” a condição de fora.

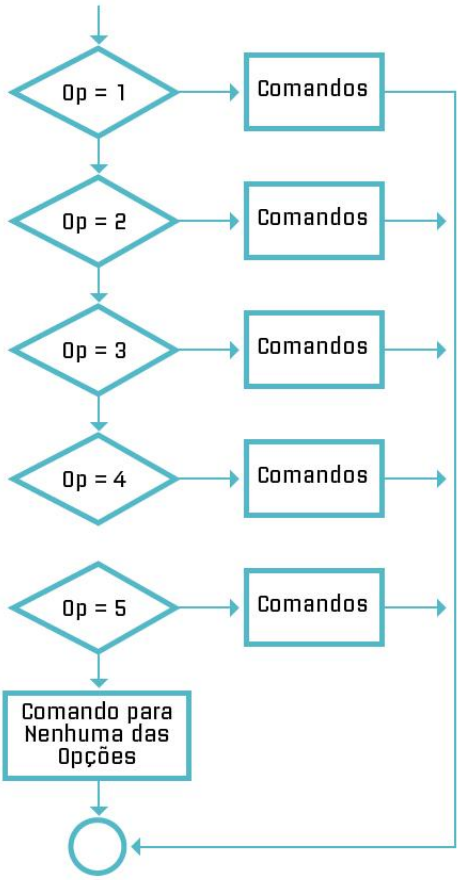
**Exemplo no fluxograma**




Estrutura de condição caso seja

Outra forma para trabalhar com comandos condicionados a um determinado valor é a estrutura caso seja. Nessa estrutura o valor de uma determinada variável é avaliado e caso esse valor coincida com determinado valor pré-estabelecido um determinado comando é executado. A estrutura de condição caso é utilizada da forma mostrada a seguir:

```
caso variável seja
<bloco de comandos>
fim-se
```




07

## 2 - TESTE DE MESA

O teste de mesa ou chinês é um teste feito manualmente, analisando o algoritmo, com valores que possam testar as suas diversas condições para que depois os resultados sejam comparados com a execução do programa no computador. Havendo qualquer divergência entre os resultados, o algoritmo deve ser analisado.

Para fazer um teste de mesa, siga passos descritos abaixo:

- Identifique as variáveis envolvidas em seu algoritmo;
- Crie uma tabela com linhas e colunas correspondentes ao número de instruções observadas pelo teste de mesa e o número de variáveis envolvidas. Utilize a primeira coluna para identificar os números das linhas correspondentes

às instruções observadas e identifique as demais colunas com o nome de uma variável;
• De cima para baixo, preencha cada uma das linhas da tabela com o número da linha que identifica cada instrução, seguida dos valores assumidos pelas variáveis do programa após a execução daquela instrução.
o Para indicar que o valor de uma variável foi lido, envolva-o entre parênteses;
o Se o valor foi escrito pela instrução, envolva-o entre chaves;
o Para valores indefinidos, ou seja, aqueles que ainda não foram determinados até uma dada instrução, utilize a interrogação.

08

Observe como um teste de mesa funciona por meio de um algoritmo simples:

*Dadas duas variáveis **a** e **b**, inicialmente com valores **v1** e **v2**, respectivamente, crie um algoritmo que atribua o valor **v2** à variável **a** e o valor **v1** à variável **b**.*

Suponhamos que sua primeira proposta de solução se pareça com o algoritmo a seguir:

**Algoritmo 1**

```
01 programa
02 var a, b: inteiro
03 //inicialização das variáveis
04 a ← v1
05 b ← v2
06 // troca dos valores das variáveis
07 a ← b
08 b ← a
09 fim
```

Analisando esse algoritmo, vejamos se ele produz o efeito desejado: perceba que a instrução da linha 7 altera o valor de *a* para *v2*, de forma que a atribuição da linha 8 não muda o valor armazenado em *b*, pois depois da linha 7 o valor de **a** ficou igual ao valor de **b**. Isso pode ser visto em detalhes no teste de mesa, onde omitimos as linhas de comentário, pois essas não afetam a execução:

**Teste de mesa do Algoritmo 1.**

| Linha | a | b |
|---|---|---|
| 02 | ? | ? |
| 04 | v1 | ? |
| 05 | v1 | v2 |
| 07 | v2 | v2 |
| 08 | v2 | v2 |

Uma possível solução para esse problema é dada pelo algoritmo seguinte:

```
Algoritmo 2
01 programa
02 var a, b, aux: inteiro
03 //inicialização das variáveis
04 a ← v1
05 b ← v2
06 //troca dos valores das variáveis
07 aux ← a
08 a ← b
09 b ← aux
10 fim
```

Neste algoritmo, introduzimos a variável aux, responsável por armazenar o valor original de a, permitindo que o acessemos mesmo depois da troca do valor de a. Esse é o procedimento padrão utilizado para troca dos valores de duas variáveis. Veja, a seguir, o teste de mesa para esse novo algoritmo:

Teste de mesa do Algoritmo 2.

| Linha | a | b | b |
|---|---|---|---|
| 02 | ? | ? | ? |
| 04 | v1 | ? | ? |
| 05 | v1 | v2 | ? |
| 07 | v1 | v2 | v1 |
| 08 | v2 | v2 | v1 |
| 09 | v2 | v1 | v1 |

Dessa forma, ao final do algoritmo a variável a tem valor v2 e a variável b tem valor v1, conforme desejado.

09

Perceba como o teste de mesa torna a análise do algoritmo mais clara, ajudando o programador a identificar falhas. O teste de mesa pode ser adaptado de acordo com o problema tratado, portanto, utilize o esquema proposto como um guia geral e ajuste-o quando necessário.

Veja agora outro exemplo da validação do algoritmo com o teste de mesa.

**Exemplo 2**

```
#incluir <biblioteca>
principal()
início
    real idade, R;
    escreva("Digite a idade");
    leia(idade);
    se (idade >= 18)
    escreva("Maior de idade");
fim
```

A tabela abaixo representa o resultado de um teste de mesa do algoritmo do exemplo anterior.

| Valor de idade | Condição (idade >= 18) | Valor da Saída |
|---|---|---|
| 10 | Falso | Vazio |
| 17 | Falso | Vazio |
| 18 | Verdadeiro | Maior de Idade |
| 20 | Verdadeiro | Maior de Idade |
| 50 | Verdadeiro | Maior de Idade |

10

Para fixar bem o teste de mesa, clique para assistir ao vídeo abaixo.

https://www.youtube.com/watch?v=CUKuXzka_Xs

11

Agora faça o teste de mesa do algoritmo a seguir e clique no link para ver o resultado.

**Exemplo 3**


Elaborar um algoritmo que receba um número e mostre o quíntuplo somente quando o número digitado for maior que trinta.

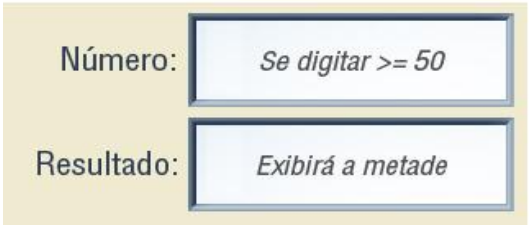
```
#incluir <biblioteca>
principal()
início
      real N, R;
      escreva("Digite um número");
      leia(N);
      R ← N;
      se (N > 30)
            R ← 5*N;
      escreva(R);
fim
```

Teste de mesa

| Valor de N | Condição (N > 30) | Valor de R |
|---|---|---|
| 1 | Falso | 1 |
| 10 | Falso | 10 |
| 30 | Falso | 30 |
| 31 | Verdadeiro | 155 |
| 100 | Verdadeiro | 500 |

12

**Exemplo 4**




Elaborar um algoritmo que receba um número e mostre a metade somente quando o número digitado for maior ou igual a cinquenta.

Vamos exercitar! Faça o algoritmo e o teste de mesa. Somente depois, clique para ver o resultado.

```
Algoritmo
Solução:
#incluir <biblioteca>
principal()
início
    real N, R;
    escreva(“Digite um número”);
    leia(N);
    R ← N;
    se (N >= 50)
        R ← N/2;
    escreva(R);
fim
```

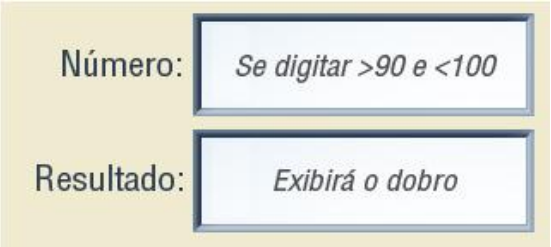
Teste de mesa

| Valor de N | Condição (N >= 50) | Valor de R |
|---|---|---|
| 1 | Falso | 1 |
| 10 | Falso | 10 |
| 50 | Verdadeiro | 25 |
| 51 | Verdadeiro | 25,5 |
| 100 | Verdadeiro | 50 |

13

**Exemplo 5**

Elaborar um algoritmo que receba um número e mostre o dobro somente quando o número digitado for maior que noventa e menor que cem.




```
Algoritmo:
#incluir <biblioteca>
principal()
início
    real N, R;
    escreva(“Digite um número”);
    leia(N);
```

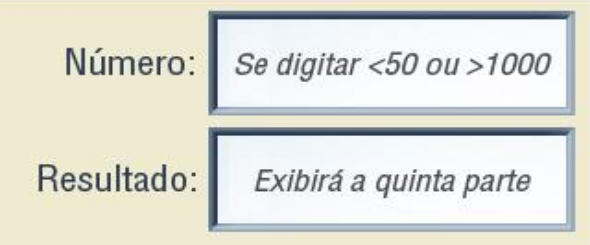
```
    R &larr N;
    se (N > 90 && N < 100)
        R ← 2*N;
    escreva(R);
fim
```

Teste de mesa

| Valor de N | Condição (N > 90 && N < 100) | Valor de R |
|---|---|---|
| 1 | Falso | 1 |
| 10 | Falso | 10 |
| 90 | Falso | 90 |
| 91 | Verdadeiro | 182 |
| 99 | Verdadeiro | 198 |
| 100 | Falso | 100 |
| 101 | Falso | 101 |
| 200 | Falso | 200 |

14

**Exemplo 6**




Elaborar um algoritmo que receba um número e mostre a quinta parte somente quando o número digitado for menor que 50 ou maior que 1000.

```
Algoritmo
#incluir <biblioteca>
principal()
início
    real N, R;
    escreva(“Digite um número”);
    leia(N);
    R ← N;
    se (N < 50 || N > 1000)
        R ← N/5;
    escreva(R);
fim
```

Teste de mesa

| Valor de N | Valor de R (N < 50 \|\| N > 1000) | Valor de R |
|---|---|---|
| -10 | Verdadeiro | -2 |
| 10 | Verdadeiro | 2 |
| 50 | Falso | 50 |
| 51 | Falso | 51 |
| 999 | Falso | 999 |
| 1000 | Falso | 1000 |
| 1001 | Verdadeiro | 200,2 |
| 2002 | Verdadeiro | 400,4 |

15

## 3 - INTERPRETAÇÃO

Os comandos 1, 2, ...e N só serão executados se a condição for verdadeira. As palavras início e fim serão necessárias apenas quando dois ou mais comandos forem executados. Se a condição for falsa, o programa executará o primeiro comando que vier após a estrutura condicional.

Observe os exemplos a seguir.

Observe os exemplos a seguir.

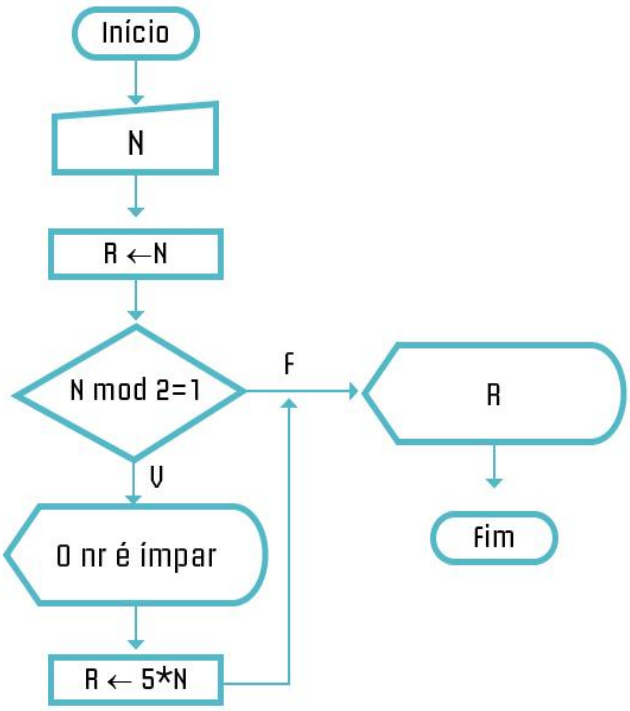
**Exemplo 7**


Elaborar um algoritmo que verifique e informe se um número é ímpar, se for calcular o seu quíntuplo.

Algoritmo

```
#incluir <biblioteca>
principal()
início
    real N, R;
    escreva("Digite um
número");
    leia(N);
    R ← N;
    se (N mod 2 = 1)
    inicio
        escreva("O
número é ímpar");
        R ← 5*N;
    fim
    escreva(R);
fim
```

Fluxograma






**Exemplo 8**

Analise o exemplo abaixo e, com os valores de entrada, verifique o resultado final do algoritmo.

Algoritmo

```
#incluir <biblioteca>
principal()
início
    inteiro A, B, C, D;
    leia(A);
    leia(B);
    leia(C);
    leia(D);
    se (A < B || B < D)
    inicio
        A ← B * C – D;
```

Teste de mesa

```
        B ← A – C * D;
        C ← B – A * D;
        D ← A + B * C – D;
        A ← A * (D – C + B);
    fim
    escreva(A, B, C, D);
fim
```

| Valores de Entrada | | | |
|---|---|---|---|
| A | B | C | D |
| 4 | 1 | 2 | 3 |

| Condição |
|---|
| Se (A < B \|\| B < D) |
| 4 < 1 ou 1 < 3 |
| F ou V |
| Resultado: V |

| Expressão | Resultado |
|---|---|
| A ← B * C – D;<br>A ← 1 * 2 – 3 | A ← -1 |
| B ← A – C * D;<br>B ← -1– 2 * 3 | B ← -7 |
| C ← B – A * D;<br>C ← -7 – (–1) * 3<br>C ← -7 + 3 | C ← -4 |
| D ← A + B * C – D;<br>D ← –1 + (–7) * (–4) – 3<br>D ← – 1 + 28 – 3 | D ← 24 |
| A ←A * (D – C + B);<br>A ← -1 *(24 – (-4) + (-7))<br>A ← -1 * (21) | A ← -21 |

| Valores de Saída | | | |
|---|---|---|---|
| A | B | C | D |
| -21 | -7 | -4 | 24 |

17

Agora é sua vez. Tente resolver o que se pede e, em seguida, clique para verificar se você acertou.

**Exemplo 9**

Verifique o resultado final do algoritmo do exemplo anterior se a linha “se (A < B || B < D)” for substituída por “se (A < B && B < D)”.

Teste de mesa

**Exemplo 10**


Elaborar um algoritmo que dado 2 números apresente-os ordenados utilizando uma Variável Auxiliar.

Algoritmo

Teste de Mesa

Teste de mesa

| Valores de Entrada | | | |
|---|---|---|---|
| A | B | C | D |
| 4 | 1 | 2 | 3 |

| Condição |
|---|
| se (A < B && B < D) |
| 4 < 1 e 1 < 3 |
| F e V |
| Resultado: F |

| Valores de Saída | | | |
|---|---|---|---|
| A | B | C | D |
| 4 | 1 | 2 | 3 |

```
Algoritmo
#incluir <biblioteca>
principal()
inicio
inteiro a, b, aux;
limpatela();
escreva(“Digite os Números: “);
leia(a,b);
se (a < b)
inicio
    aux ← a;
    a ← b;
    b ← aux;
fim
escreva(“Ordenados: “, b, a);
fim
```

Teste de mesa

| a | b | Condição (a < b) | Resultado |
|---|---|---|---|
| 1 | 2 | Verdadeira | aux← 1;<br>a← 2;<br>b ← 1;<br>Ordenados: 1, 2 |
| 10 | 5 | Falsa | Ordenados: 5, 10 |
| 2 | 2 | Falsa | Ordenados: 2, 2 |

18

## RESUMO

Neste módulo aprofundamos mais nas estruturas de condição. Esses comandos fazem parte das técnicas de programação que conduzem a estruturas de programas que não são totalmente sequenciais. Com as instruções de DESVIO pode-se fazer com que o algoritmo proceda de uma ou outra maneira, de acordo com as decisões lógicas tomadas e nos resultados anteriores. As principais estruturas de condição apresentadas são: **“se então”, “se então senão”** e **“caso seja”.**

Outro aprendizado que tivemos foi o Teste de Mesa, muito chamado de chinês. No qual o usuário pode executar manualmente o algoritmo proposto, seguindo as instruções do algoritmo de maneira precisa para verificar se o procedimento utilizado está correto ou não, escrevendo todas as variáveis e resultados em uma tabela.

## UNIDADE 2 – ESTRUTURAS DE DECISÃO, ESCOLHA E REPETIÇÃO

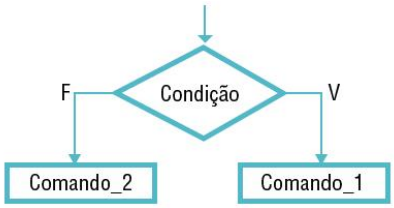
## MÓDULO 2 – ESTRUTURA DE CONDIÇÃO

01

## 1 - ESTRUTURAS DE DECISÃO

Vimos nos módulos anteriores o conceito de estruturas de decisão (simples e composto). Neste módulo aprofundaremos mais apresentando os tipos de decisões que fazem parte do dia a dia dos algoritmos e dos programas.

Exemplo de estrutura de decisão simples utilizando Fluxograma




**Sintaxe**

```
se (condição)
        comando_1;
senão
        comando_2;
```

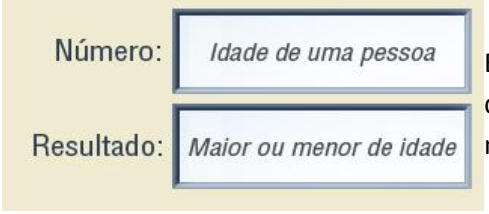
**Interpretação**

Se a condição for verdadeira, será executado o comando_1; caso contrário, se a condição for falsa, será executado o comando_ 2.

02

Observe os seguintes exemplos.

**Exemplo 1**




Elaborar um algoritmo que receba a idade de uma pessoa e informe se ela é maior ou menor de idade.

Linguagem algorítmica

```
#incluir <biblioteca>
principal()
início
    real idade, R;
    escreva("Digite a idade");
    leia(idade);
    se (idade >= 18)
```

```
        escreva(“Maior de idade”);
    senão
        escreva(“Menor de idade”);
fim
```

| Valor de idade | Condição (idade >= 18) | Valor da Saída |
|---|---|---|
| 10 | Falso | Menor de Idade |
| 17 | Falso | Menor de Idade |
| 18 | Verdadeiro | Maior de Idade |
| 20 | Verdadeiro | Maior de Idade |
| 50 | Verdadeiro | Maior de Idade |



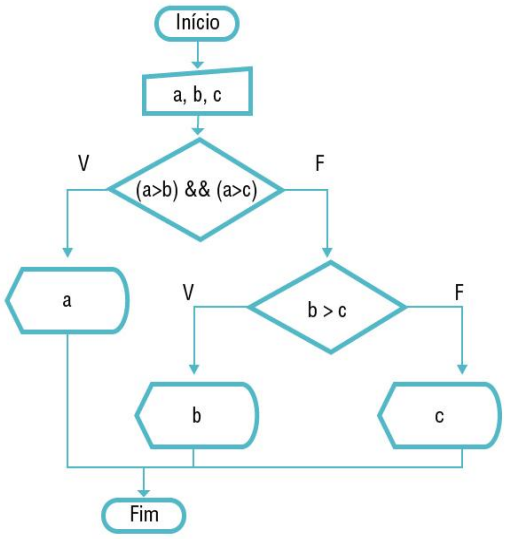
Exemplo 2

Elaborar um algoritmo que decida qual a maior entre três pessoas após informar suas estaturas.

Linguagem algorítmica

```
#incluir <biblioteca>
principal( )
inicio
real a,b,c;
escreva(“Digite as 3 alturas: “);
leia(a,b,c);
se ((a > b)&&(a > c))
      escreva("O Maior mede: ",a);
senão
      se (b > c)
            escreva("O Maior mede: ",b);
      senão
            escreva("O Maior mede: ",c);
fim
```

Fluxograma

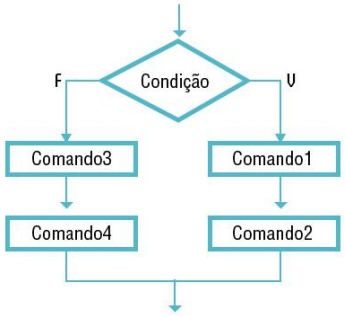



**Teste de Mesa**

| A | b | c | Condições | | Resultado |
|---|---|---|---|---|---|
| | | | (a>b)&&(a>c) | b>c | |
| 1,50 | 1,75 | 1,92 | Falso | Falso | c |
| 1,75 | 1,92 | 1,50 | Falso | Verdadeiro | b |
| 1,92 | 1,50 | 1,75 | Verdadeiro | | a |

04

Exemplo de estrutura de decisão composta utilizando fluxograma




Sintaxe

```
se (condição)
início
      comando1;
      comando2;
fim
senão
início
      comando3;
      comando4;
fim
```

05

## 2 - INTERPRETAÇÃO

Se a condição for verdadeira, o comando 1 e o comando 2 serão executados; caso contrário, se a condição for falsa, o comando 3 e o comando 4 serão executados.

Observações:

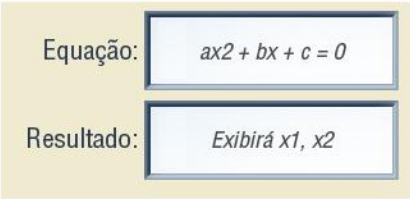
• Os recuos devem ser usados sempre que se escrever um bloco de comandos, ou seja, após o símbolo início, as instruções seguintes devem estar recuadas (à direita) de um ou mais espaços. O símbolo fim, que termina o conjunto de instruções, deve estar alinhado ao início correspondente.

• O compilador sempre associa um senão ao se anterior mais próximo que ainda não possui um senão.

06

A estrutura de decisão **se-senão** pode ser utilizada na implementação da solução das equações polinomiais do 2º e 3º grau. Veja a seguir.

**Exemplo 3**

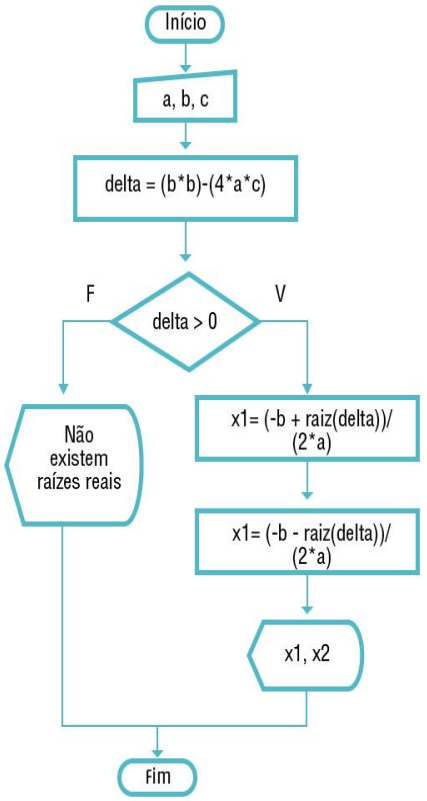



Elaborar um algoritmo que resolva uma equação do 2o grau ($ax^2 + bx + c = 0$).

Algoritmo

```
#incluir<biblioteca>
principal()
inicio
inteiro a,b,c,x1,x2,delta;
escreva("Digite a, b, c:");
leia(a,b,c);
delta ← (b*b)-(4*a*c);
se (delta >= 0)
inicio
x1 = (-b - raiz(delta))/(2*a);
x2 = (-b + raiz(delta))/(2*a);
escreva("As raízes sao: “,x1,x2);
fim
senão
escreva("Não existe raízes reais”);
fim
```

Fluxograma

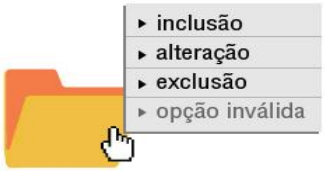



07

A estrutura de decisão **se-senão** também pode ser utilizada na elaboração de um menu de opções. No entanto, no próximo tópico veremos um comando melhor para a realização desta ação.

**Exemplo 4**

Elaborar um algoritmo que apresente um menu com as opções de inclusão, alteração, exclusão e opção inválida.




**Linguagem algorítmica**

```
#incluir <biblioteca>
principal()
inicio
    inteiro x;
    escreva("1. Inclusao");
    escreva("2. Alteracao");
    escreva("3. Exclusao");
    escreva("Digite sua opção: ");
    leia(x);
    se (x = 1) escreva("Escolheu inclusao\n");
    se (x = 2) escreva("Escolheu alteracao\n");
    se (x = 3) escreva("Escolheu exclusao\n");
    se (x != 1 && x != 2 && x != 3) escreva("Opção invalida\n");
fim
```



3 - Comando Escolha

Diversas vezes é necessário escolher uma opção que se encontra numa lista de um menu. O comando escolha() aparece como a melhor opção para esta necessidade.

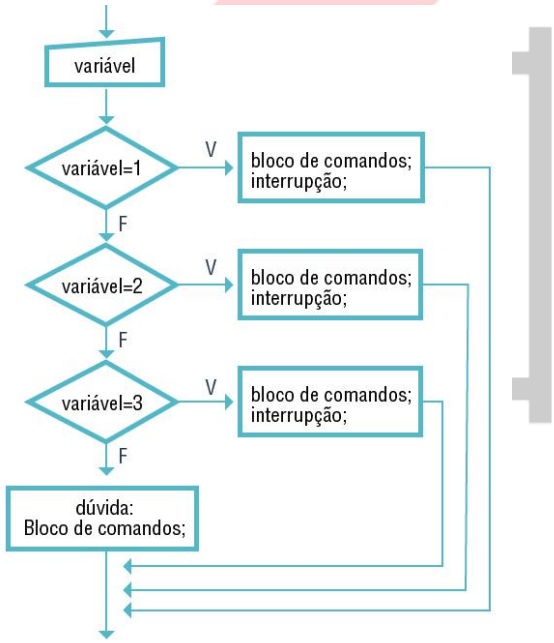
Sintaxe

```
escolha (variável)
inicio
    caso 1:
        bloco de
comandos;
        interrupção;
    caso 2:
        bloco de
comandos;
```

```
                interrupção;
        caso 3:
                bloco de
comandos;
                interrupção;
        dúvida:
                bloco de
comandos;
fim
```

Uma variável é testada sucessivamente contra uma lista de variáveis inteiras ou de caracteres. Depois de encontrar uma coincidência, o bloco de comandos correspondente é executado. Se nenhuma coincidência for encontrada, o bloco de comandos do caso dúvida será executado. O comando interrupção serve para terminar o bloco de comandos em execução.

Exemplo do comando escolha utilizando fluxograma




O comando **interrupção** é opcional e por isto, se for suprimido, permite que o próximo "caso" seja executado, sem haver qualquer quebra na sequência do processamento. Observa-se que não é necessário utilizar o comando **início** e o comando **fim** para delimitar o bloco de comandos.

09

Agora é sua vez. Faça o que se pede nos exemplos a seguir e depois verifique se você acertou.

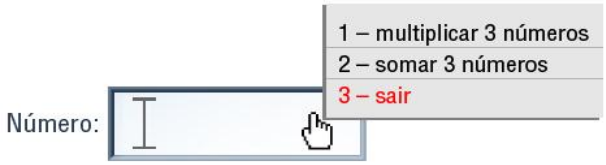
**Exemplo 5**


Elaborar um algoritmo que apresente um menu com as opções de inclusão, alteração, exclusão e sair.

Linguagem algorítmica

**Exemplo 6**

Elaborar um algoritmo que apresente um menu com as opções: 1 – multiplicar 3 números; 2 – somar 3 números e 3 – sair.

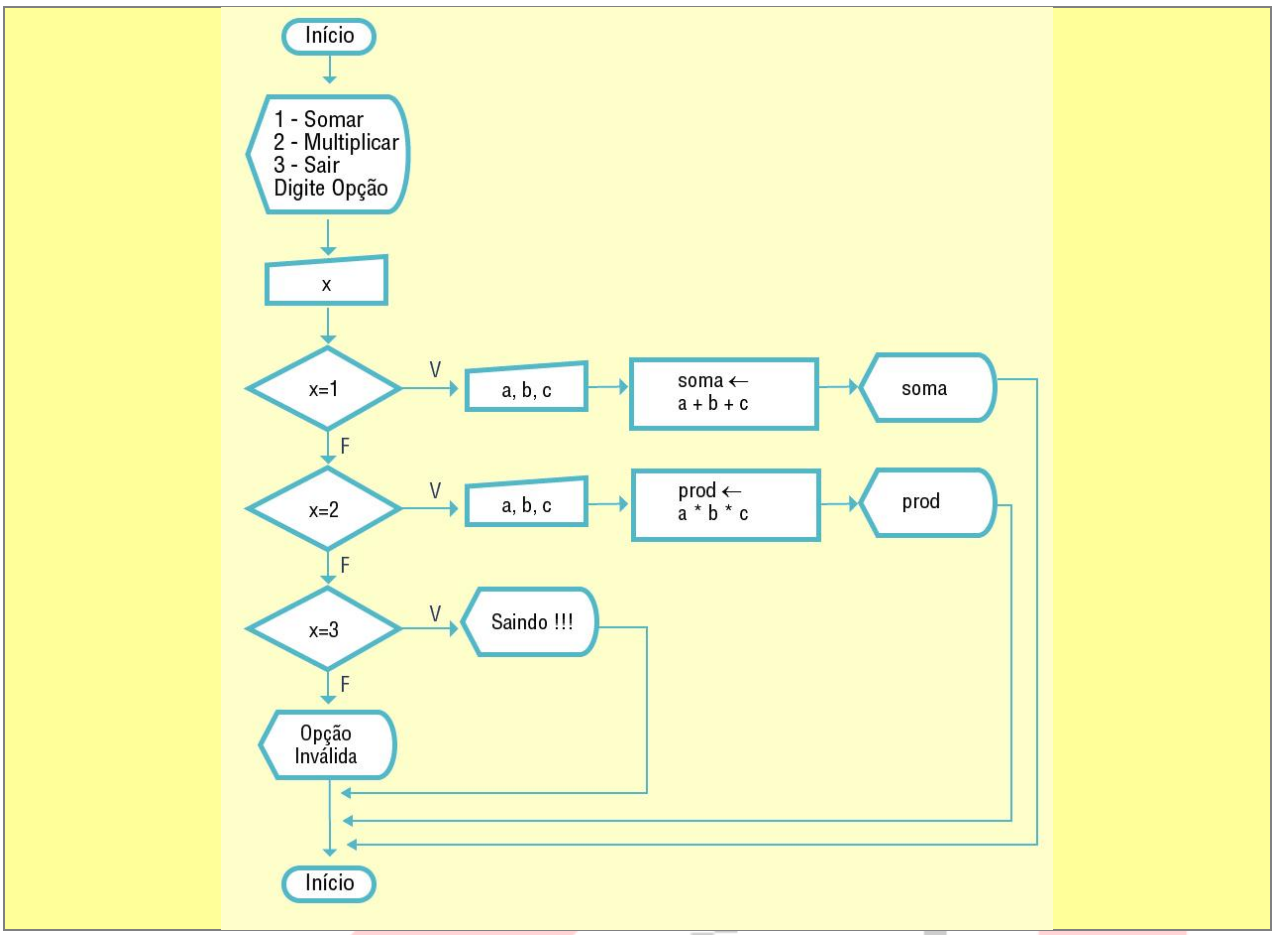



Linguagem algorítmica

Fluxograma

```
Linguagem algorítmica:
#incluir <biblioteca>
principal()
inicio
    inteiro x;
    escreva("1. Inclusao");
    escreva("2. Alteracao");
    escreva("3. Exclusao");
    escreva("4. Sair");
    escreva("Digite sua opção: ");
    leia(x);
    escolha (x)
    inicio
    caso 1: escreva(“Escolheu inclusao”);     interrupção;
    caso 2: escreva(“Escolheu alteracao”);     interrupção;
    caso 3: escreva(“Escolheu exclusao”);     interrupção;
    caso 4: escreva(“Atualizando     alterações”); interrupção;
    dúvida: escreva(“Opção invalida”);
    fim
fim
```

```
Linguagem algorítmica
#incluir <biblioteca>
principal()
inicio
inteiro x;
real a, b, c, soma, prod;
escreva(“1. Somar 3 números”);
escreva(“2. Multiplicar 3 números”);
escreva(“3. Sair”);
escreva(“Digite sua opção: “);
leia(x);
escolha(x)
    inicio
    caso 1:
        escreva(“Digite 3 números”);
        leia(a,b,c);
        soma = a + b + c;
        escreva(“Soma = “, soma);
        interrupção;
    caso 2:
        escreva(“Digite 3 números”);
        leia(a,b,c);
        prod = a * b * c;
        escreva(“Produto = “, prod);
        interrupção;
    caso 3:
        escreva(“Saindo !!!!”);
        interrupção;
    dúvida: escreva(“Opção invalida\n”);
    fim
fim
```




10

## RESUMO

Nos módulos anteriores foram apresentados alguns conceitos básicos sobre as estruturas e comandos que são utilizados para construir um algoritmo. Neste módulo foi apresentado o comando da estrutura de decisão ***escolha***, que possibilita ao usuário do sistema configurar uma lista em um menu de um sistema qualquer.

Vimos também que no algoritmo você poderá realizar uma interrupção dos comandos que estão sendo executados, usando o comando **interrupção**, que para a execução do algoritmo até a próxima ação do usuário.

As possibilidades de construção de algoritmos que temos até o presente momento são bastante boas para construirmos quase todo o tipo de algoritmo.

# UNIDADE 2 – ESTRUTURAS DE DECISÃO, ESCOLHA E REPETIÇÃO
## MÓDULO 3 - ESTRUTURA PARA
01

## 1- ESTRUTURAS DE REPETIÇÃO - PARA

Aprendemos no início do curso o conceito de Estrutura de Repetição. Nesse módulo e no próximo módulo avançaremos mais sobre o assunto. Aprenderemos nesse módulo a estrutura de repetição "Para".

Vale lembrar que comandos de repetição são estruturas utilizadas no corpo de um algoritmo que possibilitam ao computador repetir a mesma tarefa.

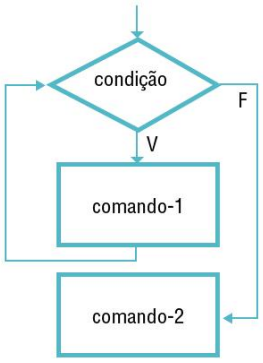
Os comandos de repetição são recursos muito importantes nas linguagens de programação devido ao grande poder de processamento dos computadores, que permitem a execução de tarefas repetidas com extrema rapidez.

> A estrutura de repetição PARA é utilizada quando se sabe o número de vezes que um trecho do algoritmo deve ser repetido.

A estrutura de repetição PARA, implementa um contador implicitamente. Veja como é o seu esquema:

**PARA** <variável contadora> **DE** <valor inicial> **ATE** <valor final> [**PASSO** <valor de incremento>] **FAÇA** <instruções a serem executadas repetidamente até a <variável contadora> atingir o valor final>

Exemplo de estrutura de repetição Para utilizando fluxograma com um comando




Sintaxe

```
para(i ← 1; i <= cond_lim; i ← i + 1)
```

```
comando_1;
```

Interpretação

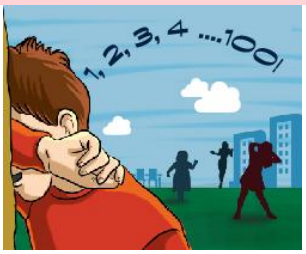
O comando será executado utilizando a variável i como controle, cujo conteúdo vai variar do valor inicial 1 até o valor final cond_lim, de 1 em 1, incrementando automaticamente.

02

Verifique como ocorre esse comando, observando os exemplos a seguir.

**Exemplo 1**

Elaborar um algoritmo que realize a contagem de 1 a 100, apresentando-a na tela.




**Linguagem algorítmica**

```
#incluir <biblioteca>
principal()
inicio
  inteiro i;
  para (i ← 1; i <= 100; i ← i + 1)
    escreva(i, " – ");
  escreva(i);
fim
```

Observação: a penúltima linha **escreva(i)**; irá escrever o último número (100), pois quando i for igual a 100, a condição (i<100) será falsa, portanto, a instrução dentro do laço não será executada.

03

Agora é sua vez. Faça o que é pedido nos exemplos a seguir e depois clique para verificar o resultado.

**Exemplo 2**


A tabuada é uma tabela contendo todos os produtos dos números inteiros de 0 a 10. E laborar um algoritmo que imprima a tabuada de um número fornecido pelo usuário.

Linguagem algorítmica

Teste de Mesa

**Exemplo 3**


Na matemática, o fatorial de um número natural n é o produto de todos os inteiros positivos menores ou iguais a n. Isso é escrito como n! e lido como "fatorial de n". A notação n! foi introduzida por Christian Kramp em 1808. Elaborar um algoritmo que calcule o fatorial de um número inteiro informado pelo usuário.

0! = 1
1! = 1
n!=n.(n-1)...3.2.1

Linguagem algorítmica

Teste de Mesa

```
Linguagem algorítmica
#incluir <biblioteca>
principal()
inicio
  inteiro i,num;
  escreva("digite um numero: ");
  leia(num);
```

```
  para(i ← 0; i <= 10; i ← i + 1)
    escreva(num, "*", i, "=", num * i);
fim
```

Teste de Mesa

| Valor de Entrada |
|---|
| num=7 |

| i | Saída |
|---|---|
| 0 | 7*0 = 0 |
| 1 | 7*1 = 7 |
| 2 | 7*2 = 14 |
| 3 | 7*3 = 21 |
| 4 | 7*4 = 28 |
| 5 | 7*5 = 35 |
| 6 | 7*6 = 42 |
| 7 | 7*7= 49 |
| 8 | 7*8 = 56 |
| 9 | 7*9= 63 |
| 10 | 7*10 = 70 |

Linguagem algorítmica

```
#incluir <biblioteca>
principal()
inicio
   inteiro i, num, fat;
   fat ← 1;
   escreva("digite um numero: ");
   leia(num);
   se (num = 0)
     escreva("o fatorial de", num, "e'", fat);
   senão
   início
     para(i=1; i<=num ; i← ! i+1)
        fat = fat*i;
     escreva("o fatorial de", num, "e'", fat);
   fim
fim
```

Teste de Mesa

| Valor de Entrada |
|---|
| num=8 |

| i | fat |
|---|---|
| 1 | fat = 1*1 |
| 2 | fat = 1*2 |
| 3 | fat = 2*3 |
| 4 | fat = 6*4 |
| 5 | fat = 24*5 |
| 6 | fat = 120*6 |
| 7 | fat = 720*7 |
| 8 | fat = 5040*8 |

| Valor de Saída |
|---|
| fat=40320 |

04

Exemplo 4

Elaborar um algoritmo que calcule quantos múltiplos de um determinado número fornecido pelo usuário existem no intervalo de 1 a 1000 e informe quais são.


**LINGUAGEM ALGORÍTMICA**

```
#incluir <biblioteca>
principal()
inicio
 inteiroi,num,cont;
 escreva("Digite um numero: ");
 leia(num);
 cont← 0;
 para(i=1; i<=1000 ; i←i+1)
  se(i mod num = 0)
  inicio
   cont←cont + 1;
   escreva(i);
  fim
 escreva("Existem ", cont, "múltiplos");
fim
```

**TESTE DE MESA**

| Valor de Entrada |
|---|
| num=175 |

| Valores de Saída |
|---|
| 175 |
| 350 |
| 525 |
| 700 |
| 875 |
| Existem 5 múltiplos |

05

Exemplo 5


Criptografia é a codificação e descaracterização de mensagens para impedir o acesso não autorizado ou a compreensão dos dados transmitidos. Uma mensagem pode ser criptografada aplicando-se a ela um código numérico secreto, denominado chave criptográfica. Em geral, muitos destes códigos utilizam as propriedades da teoria dos números primos. Elaborar um algoritmo que descubra se um determinado número inteiro informado pelo usuário é primo ou composto.

Um número é primo se possui apenas 2 divisores naturais, o número 1 e ele mesmo.
P = {2, 3, 5, 7, 11, 13, 17, 19, ...}

Linguagem algorítmica

Teste de mesa 1

Teste de mesa 2

Linguagem algorítmica:

```
#incluir <biblioteca>
principal()
inicio
 inteiro i,num,cont;
 escreva("Digite um numero: ");
 leia(num);
 cont ← 0;
 para(i=1; i <= num ; i ← !i + 1)
    se(num mod i = 0)
     cont ← cont + 1;
     se(cont = 2)
   escreva("O número", num, "é primo");
senão
   escreva("O número", num, "não é
primo");
fim
```

Teste de Mesa

| Valor de Entrada |
|---|
| num=5 |

| i | condição | cont |
|---|---|---|
| 1 | 5 mod 1 = 0 | 1 |
| 2 | 5 mod 2 = 1 | 1 |
| 3 | 5 mod 3 = 2 | 1 |
| 4 | 5 mod 4 = 1 | 1 |
| 5 | 5 mod 5 = 0 | 2 |

| Valor de Saída |
|---|
| O número 5 é primo |

Teste de Mesa

| Valor de Entrada |
|---|
| num=6 |

| i | condição | cont |
|---|---|---|
| 1 | 6 mod 1 = 0 | 1 |
| 2 | 6 mod 2 = 0 | 2 |
| 3 | 6 mod 3 = 0 | 3 |
| 4 | 6 mod 4 = 2 | 3 |
| 5 | 6 mod 5 = 1 | 3 |
| 6 | 6 mod 6 = 0 | 4 |

| Valor de Saída |
|---|
| O número 6 não é primo |

06

Exemplo 6

Elaborar um algoritmo que some os números naturais até um número informado pelo usuário.

$$\sum_{i=1}^{n} i = 1+2+3+...+ n$$

Linguagem algorítmica

```
#incluir <biblioteca>
principal()
inicio
    inteiro i,num,soma;
    escreva("Digite um numero: ");
    leia(num);
    soma ← 0;
    para(i=1; i<=num ; i←i+1)
    soma ← soma + i;
    escreva("Soma =", soma);
fim
```

07

Exemplo 7


Número par é todo número inteiro que ao ser dividido pelo número dois resulta em um número inteiro. Elaborar um algoritmo que apresente a soma dos N primeiros números pares.

$$\sum_{i=1}^{n} 2.i = 2+4+6+...+ 2.n$$

| LINGUAGEM ALGORÍTMICA | TESTE DE MESA |
|---|---|

```
#incluir <biblioteca>
principal()
inicio
 inteiroi,num,soma;
 escreva("Digite um numero: ");
 leia(num);
 soma← 0;
 para(i=1; i<=num ; i←i+1)
  soma←soma + 2*i;
 escreva("Soma =", soma);
fim
```

| Valor de Entrada |
|---|
| num=8 |

| i | soma |
|---|---|
| 1 | 0 + 2*1=2 |
| 2 | 2 + 2*2=6 |
| 3 | 6 + 2*3=12 |
| 4 | 12 + 2*4=20 |
| 5 | 20 + 2*5=30 |
| 6 | 30 + 2*6=42 |
| 7 | 42 + 2*7=56 |
| 8 | 56 + 2*8=72 |

| Valor de Saída |
|---|
| Soma = 72 |

Observação: no teste de mesa, o algoritmo soma os oito primeiros números pares, ou seja, 2+4+6+8+10+12+14+16.

08

Exemplo 8


Número ímpar é todo número inteiro que ao ser dividido pelo número dois resulta em um número racional não inteiro. Elaborar um algoritmo que some todos os números ímpares até 2005.

$$\sum_{i=1}^{1003} 2.i-1 = 1+3+5+...+ (2.1003-1)$$

```
#incluir <biblioteca>
principal()
inicio
      inteiro i,num,soma;
      soma←0;
      para(i=1; i<=2005 ; i←i+2)
```

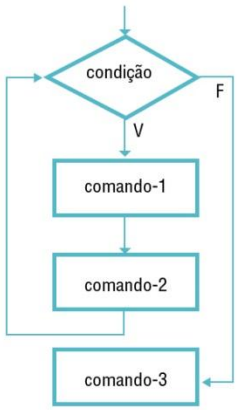
```
        soma←soma + i;
    escreva(“Soma =”, soma);
fim
```

Observação: no exemplo, o incremento da variável de controle i é 2 e não 1 (i←i+2), como nos demais exemplos, o que significa que o i inicia em 1 e a cada passo pula os números pares: 1, 3, 5, ..., 2005.

09

## 2- Estrutura Para – Bloco de Comandos

Exemplo de estrutura de repetição **Para** utilizando Fluxograma com mais de um comando




Sintaxe:

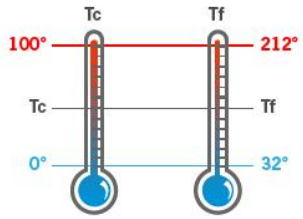
```
para(i ← 1; i <= cond_lim; i ← i + 1)
inicio
    comando-1;
    comando-2;
fim
comando-3;
```

Interpretação:

Os comando1 e comando2 serão executados utilizando a variável i como controle, cujo conteúdo vai variar do valor inicial 1 até o valor final cond_lim, de 1 em 1, incrementando automaticamente.

10

Exemplo 9




Exemplo: Elaborar um algoritmo que converta temperaturas dadas na Escala Fahrenheit (1 a 300 ºF) para a Escala Celsius.

```
#incluir <biblioteca>
principal()
inicio
    inteirofahr;
    realcelsius;
    para (fahr = 1; fahr<= 300; fahr = fahr + 1)
    inicio
        escreva(fahr);
        celsius ← (5.0/9.0)*(fahr-32);
        escreva(celsius);
    fim
fim
```

11

Agora é sua vez. Tente resolver o que se pede e clique para ver o resultado.

**Exemplo 10**


Elaborar um algoritmo que receba quatro notas e calcule a média.

**LINGUAGEM ALGORÍTMICA**

```
#incluir <biblioteca>
principal()
inicio
 inteiro i;
 real nota, media;
 media← 0;
 para (i = 1; i <=4; i = i + 1)
 inicio
  leia(nota);
  media ←media + nota;
 fim
 escreva(media/4);
fim
```

**TESTE DE MESA**

| i | condição | Média |
|---|---|---|
| 1 | nota = 5.5 | 5.5 |
| 2 | nota = 7.3 | 12.8 |
| 3 | nota = 6.4 | 19.2 |
| 4 | nota = 5.1 | 24.3 |

| Valor de Saída |
|---|
| media = 6.075 |

12

## RESUMO

Caro aluno, estamos aos poucos evoluindo nas estruturas de repetição. Estruturas essas que compõem os algoritmos e os programas que são construídos pelos programadores. Os comandos de repetição são estruturas utilizadas no corpo de um algoritmo que possibilitam ao computador repetir a mesma tarefa quantas vezes forem necessária para satisfazer a condição preestabelecida.

Nesse módulo evoluímos na estrutura de repetição PARA. O comando PARA é executado até a condição do PARA for verdadeira, depois ela vai para o próximo comando.

A estrutura de repetição PARA pode ser utilizada para diversas necessidades de algoritmos como tabuada, conversão de escala de temperatura dentre outros.

## UNIDADE 2 – ESTRUTURAS DE DECISÃO, ESCOLHA E REPETIÇÃO
## MÓDULO 4 - ESTRUTURA ENQUANTO E FAÇA-ENQUANTO

01

## 1- ESTRUTURA DE REPETIÇÃO - ENQUANTO

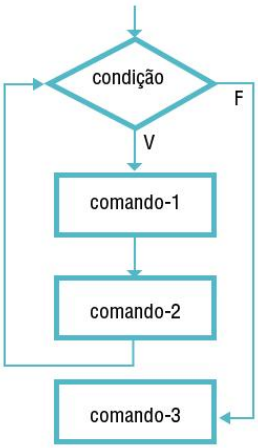
> Essa estrutura de repetição é utilizada quando não se sabe o número de vezes em que um trecho do algoritmo deve ser repetido, embora também possa ser utilizada quando se sabe esse número.

Existem situações em que o teste condicional da estrutura de repetição, que fica no início, resulta em um valor falso logo na primeira comparação. Nestes casos, os comandos de dentro da estrutura de repetição não serão executados.

Sintaxe:

```
enquanto (condição)
inicio
comando_1;
comando_2;
fim
```

Fluxograma:




**Interpretação:**

Enquanto a condição for verdadeira, o comando-1 e comando-2 serão executados.



Observe o uso da estrutura *enquanto* nos exemplos a seguir.

**Exemplo 1**

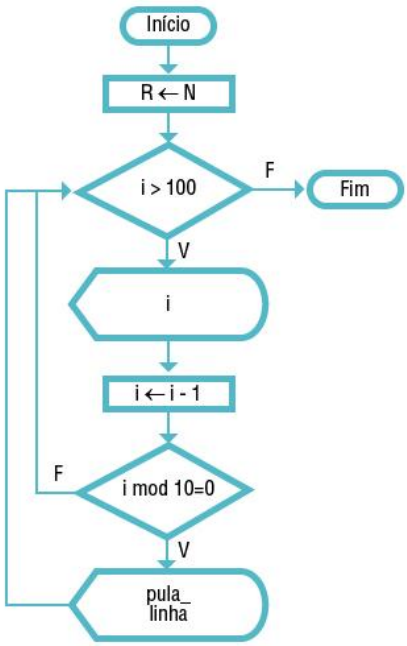

Elaborar um algoritmo que apresente em ordem decrescente os números de 100 até 11, ordenados de 10 em 10 por linha.

**Linguagem algorítmica**

```
#incluir <biblioteca>
principal()
inicio
    inteiro i;
    i ← 100;
    enquanto(i > 10)
    inicio
        escreva(i);
        i ← i - 1;
        se(i mod 10 = 0)
```

```
            escreva(pula_linha);
    fim
fim
```

Exemplo de estrutura de repetição **ENQUANTO** utilizando Fluxograma



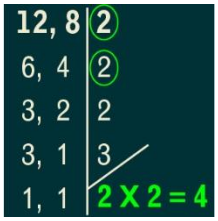

**Teste de Mesa**

| Valor de Saída |
| --- |
| 100<br>99 98 97 96 95 94 93 92 91 90<br>89 88 87 86 85 84 83 82 81 80<br>⋮<br>19 18 17 16 15 14 13 12 11 |

03

Agora, tente resolver os algoritmos a seguir e depois clique para verificar o resultado.

**Exemplo 2**




O máximo divisor comum entre dois números a e b (vulgarmente abreviada como mdc(a,b)) é o maior número inteiro encontrado, que seja fator dos outros dois. Elaborar um algoritmo para o cálculo do máximo divisor comum entre 2 números.

| LINGUAGEM ALGORÍTMICA | TESTE DE MESA |
|---|---|
| #incluir <biblioteca><br>principal()<br>inicio<br>inteiromaior,menor,res;<br>escreva("digite o maior número: ");<br>leia(maior);<br>escreva("digite o menor número: ");<br>leia(menor);<br>enquanto (menor != 0)<br>inicio<br>res← maior mod menor;<br>maior←menor;<br>menor← res;<br>fim<br>escreva("o mdc entre os numeros é igual a", maior);<br>fim | **Valor de Entrada**<br>Maior=12<br>menor=8<br>**Passos**<br>res← 12 mod 8 = 4<br>maior← 8<br>menor← 4<br>res← 8 mod 4 = 0<br>maior← 4<br>menor← 0<br>**Valor de Saída**<br>O mdc é 4 |

04

Exemplo 3

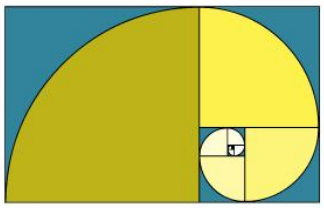
Elaborar um algoritmo que calcule a média de 500 números informados pelo usuário.

**LINGUAGEM ALGORÍTMICA**

```
#incluir <biblioteca>
principal()
inicio
 inteiro i;
 real num, soma;
 i ← 1;
 soma← 0;
 enquanto (i <= 500)
 inicio
  leia(num);
  soma←soma + num;
  i ←i + 1
 fim
 escreva("A média é", soma/500);
fim
```

05

Exemplo 4



Sequência de Fibonacci é uma sequência de números inteiros normalmente iniciada por 0 e 1 onde cada número seguinte (número de Fibonacci) corresponde à soma dos dois elementos anteriores. Elabore um algoritmo que liste os n primeiros elementos da Sequência de Fibonacci, onde n deverá ser informado pelo usuário.

Linguagem algorítmica

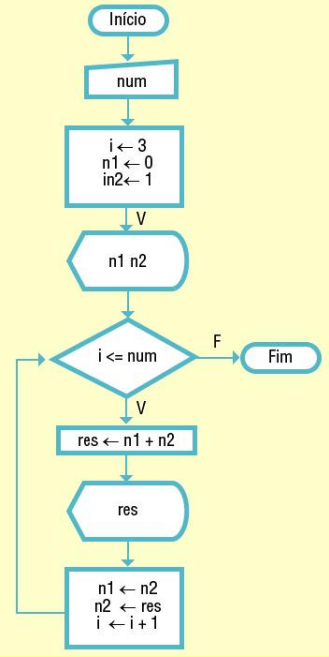
Fluxograma

Teste de mesa

Linguagem algorítmica:

```
#incluir <biblioteca>
principal()
inicio
    inteiro i, num, res, n1, n2;
    escreva("Digite o número de elementos: ");
    leia(num);
    i ← 3;
    n1←0;
    n2←1;
    escreva(n1);
    escreva(n2);
    enquanto (i <= num)
    inicio
        res← n1 + n2;
        escreva(res);
        n1 ← n2;
        n2 ← res;
        i ← i + 1
    fim
fim
```

Fluxograma



Teste de mesa

| Valor de Entrada |
|---|
| num=10<br>n1=0<br>n2=1 |

| i | Entrada |
|---|---|
| 3 | res←0+1<br>n1←1<br>n2←1 |
| 4 | res←1+1<br>n1←1<br>n2←2 |
| 5 | res←1+2<br>n1←2<br>n2←3 |
| 6 | res←2+3<br>n1←3<br>n2←5 |
| 7 | res←3+5<br>n1←5<br>n2←8 |
| 8 | res←5+8<br>n1←8<br>n2←13 |
| 9 | res←8+13<br>n1←13<br>n2←21 |
| 10 | res←13+21<br>n1←13<br>n2←34 |

| Saída |
|---|
| 0 1 1 2 3 5 8 13 21 34 |



06

## 2 - Estrutura de Repetição - FAÇA-ENQUANTO

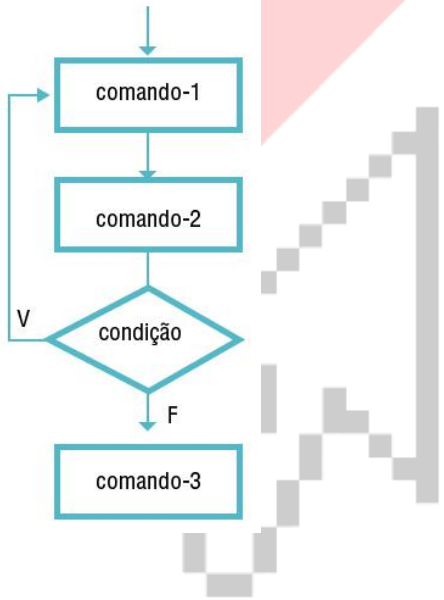
Essa estrutura de repetição é semelhante à estrutura Enquanto, ou seja, é utilizada quando não se sabe o número de vezes em que um trecho do algoritmo deve ser repetido, embora também possa ser utilizada quando se sabe esse número.

No entanto, o teste condicional da estrutura Faça-Enquanto é feito no fim do bloco de comando, portanto, sempre ocorre a execução dos comandos.

Sintaxe:

```
faça
inicio
    comando_1;
    comando_2;
fim
enquanto (condição);
```

Exemplo de estrutura de repetição FAÇA ENQUANTO utilizando Fluxograma




**Interpretação:**

Enquanto a condição for verdadeira, os comando1 e comando2 serão executados.

07

Veja, no exemplo a seguir, como é usada a Estrutura Faça-Enquanto.

**Exemplo 5**


Elaborar um algoritmo que apresente um menu com as opções de inclusão, alteração, exclusão e opção sair.

**Linguagem algorítmica**

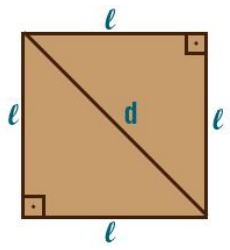
```
#incluir <biblioteca>
principal()
inicio
    inteiro op;
    faça
    inicio
        escreva("1. Inclusao");
        escreva("2. Alteracao");
        escreva("3. Exclusao");
        escreva("4. Sair");
        escreva("Digite sua opção: ");
        leia(op);
        escolha(op)
        inicio
            caso 1: escreva("Escolheu inclusao"); interrupção;
            caso 2: escreva("Escolheu alteracao"); interrupção;
            caso 3: escreva("Escolheu exclusao"); interrupção;
            caso 4: escreva("Sair"); interrupção;
            dúvida: escreva("Opcao Inválida");
        fim
    fim
    enquanto(op != 4);
fim
```

08

Agora é a sua vez. Faça o que se pede nos exemplos a seguir e depois clique para verificar o resultado.

**Exemplo 6**

Elaborar um algoritmo que leia o lado de um quadrado e calcule a sua área evitando que o usuário informe valores negativos ou zero.

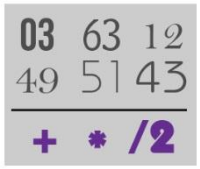


**LINGUAGEM ALGORÍTMICA**

```
#incluir <biblioteca>
principal()
inicio
realarea, lado;
faça
leia(lado)
Enquanto (lado <=0);
area = lado*lado;
escreva(área);
fim
```

09

Exemplo 7




Elaborar um algoritmo que receba dois números positivos, e o usuário escolha se quer calcular a soma, o produto ou a média desses números.

**LINGUAGEM ALGORÍTMICA**

```
#incluir <biblioteca>
principal()
inicio
 real n1, n2, res;
 inteiroop;
 faça
  leia(n1)
enquanto (n1 <=0);
 faça
  leia(n2)
enquanto (n2<=0);
 faça
 inicio
  escreva("1. Soma");
  escreva("2. Produto");
  escreva("3. Média");
  escreva("4. Sair");
  escreva("Digite sua opção: ");
  leia(op);
  escolha(op)
  inicio
   caso 1: escreva(n1 + n2); interrupção;
   caso 2: escreva(n1 * n2); interrupção;
   caso 3: escreva ((n1+n2)/2); interrupção;
   caso 4: escreva("Sair"); interrupção;
   dúvida: escreva("Opcao Inválida");
  fim
 fim
 enquanto(op != 4);
fim
```

10

Exemplo 8


Elaborar um algoritmo que receba o tipo de gasto (Cartão ou Dinheiro) e o valor, no fim calcule o total para cada tipo de gasto.

**LINGUAGEM ALGORÍTMICA**

```
#incluir <biblioteca>
principal()
inicio
 real gasto, s_cartao, s_din;
 inteiroop;
 literal tipo;
 s_cartao← 0;
 s_din← 0;
 faça
 inicio
  escreva("C. Cartão");
  escreva("D. Dinheiro");
  escreva("T. Total");
  escreva("Digite sua opção: ");
  leia(op);
  leia(gasto);
  seop = "C" s_cartao←s_cartao + gasto;
  seop = "D" s_din←s_din + gasto;
 fim
 enquanto(op != "T");
 escreva("Total de gasto com cartão",s_cartao);
 escreva("Total de gasto com dinheiro",s_din);
fim
```

11

Exemplo 9

Elabore um algoritmo que some os votos de uma eleição para 100 eleitores com 4 candidatos possibilitando os votos branco e nulos.

11

**LINGUAGEM ALGORÍTMICA**

```
#incluir <biblioteca>
principal()
inicio
 inteiro voto, tot_c1, tot_c2, tot_c3, tot_c4, tot_b, tot_n;
 tot_c1 ← 0;
 tot_c2 ← 0;
 tot_c3 ← 0;
 tot_c4 ← 0;
 tot_b← 0;
 tot_n← 0;
 i ← 0;
 faça
 inicio
  escreva("1. Candidato 1");
  escreva("2. Candidato 2");
  escreva("3. Candidato 3");
  escreva("4. Candidato 4");
  escreva("5. Branco");
  escreva("Outro. Nulo");
  escreva("Digite seu voto: ");
  leia(voto);
  escolha(voto)
  inicio
   caso 1: tot_c1 ←tot_c1 + 1; interrupção;
   caso 2: tot_c2 ←tot_c2 + 1; interrupção;
   caso 3: tot_c3 ←tot_c3 + 1; interrupção;
   caso 4: tot_c4 ←tot_c4 + 1; interrupção;
   caso 5: tot_b←tot_b + 1; interrupção;
   dúvida: tot_n←tot_n + 1; interrupção;
  fim
  i ←i + 1;
 fim
 enquanto (i<=100);
```

12

## RESUMO

Caro aluno, continuamos nesse módulo estudando as Estruturas de Repetição. Aprendemos duas estruturas: ENQUANTO e FAÇA ENQUANTO. A grande diferença das duas é a execução dos comandos que estão dentro dessa estrutura. No caso do FAÇA ENQUANTO, os comandos são executados e só após é verificado se a condição ENQUANTO atende.

Aprendemos também que a utilização do ENQUANTO ocorre quando não sabemos o momento da parada da repetição dos comandos.

Todas essas estruturas estudadas serão muito úteis no dia a dia da programação, então aproveite o máximo possível o seu estudo.