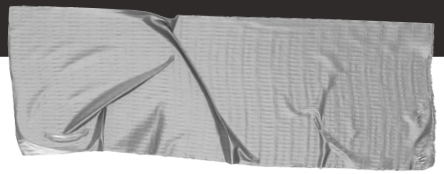
Módulo II - Linguagens de Programação

Tópico 3 - Lógica de Programação e Algoritmos

.StartSe

O que são constantes e variáveis?



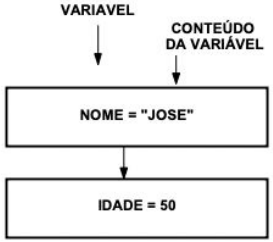
Variáveis e constantes

Variáveis e **constantes** são os elementos básicos que um programa manipula. **Uma variável é um espaço reservado na memória do computador para armazenar um tipo de dado determinado.**

Variáveis devem receber nomes para poderem ser referenciadas e modificadas quando necessário. **Constantes não necessariamente recebem nomes**, mas podem ser nomeadas. Um programa deve conter declarações que **especificam de que tipo são as variáveis** que ele utilizará e às vezes um valor inicial. As expressões combinam variáveis e constantes para calcular novos valores.

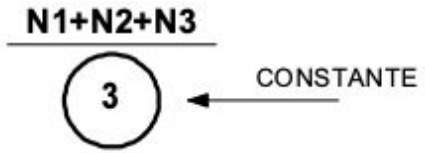
Variáveis

Variável é a representação simbólica dos elementos de um certo conjunto. Cada variável corresponde a uma posição de memória, cujo conteúdo pode se alterado ao longo do tempo durante a execução de um programa. Embora uma variável possa assumir diferentes valores, ela só pode armazenar um valor a cada instante.



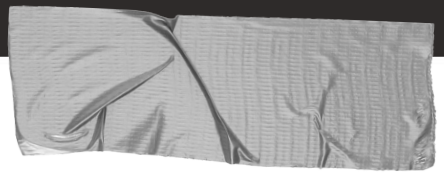
Constantes

Constante é um determinado valor fixo que não se modifica ao longo do tempo, durante a execução de um programa. Conforme o seu tipo, a constante é classificada como sendo numérica, lógica e literal.




```
const city = "Noida";
```

O que são tipos de dados?



Tipos de dados

As variáveis e as constantes podem ser basicamente de quatro tipos: numéricas, caracteres, alfanuméricas ou lógicas.

Numéricas: Específicas para armazenamento de números, que posteriormente poderão ser utilizados para cálculos. Podem ser ainda classificadas como Inteiras ou Reais. As variáveis do tipo inteiro são para armazenamento de números inteiros e as Reais são para o armazenamento de números que possuam casas decimais. (**int, float, double, long**)

Caracteres: Específicas para armazenamento de caracteres que não contenham números (literais). (**char**)

Alfanuméricas: Específicas para dados que contenham letras e/ou números. Pode em determinados momentos conter somente dados numéricos ou somente literais. Se usado somente para armazenamento de números, não poderá ser utilizada para operações matemáticas. (**string**)

Lógicas: Armazenam somente dados lógicos que podem ser Verdadeiro ou Falso. (**bool**)

Podemos definir o tipo de uma variável antes de declarar seu nome como o exemplo em Dart em cima.

E há linguagens como python que inferem o tipo atribuído a variável, automaticamente atribuindo um tipo.

Em algoritmos ou em código em pseudo-linguagens, também devemos definir o tipo de dado de nossa variável ou constante.

```dart
String text = "This is a string 123";

int integer = 22

double precise = 3.1415926535912154548

bool boolean = true;
```

```python
a = 'a'
number = 12
decimal = 12.3
double = 12.123123123123123
string = "now or never"
boolean = True
```

Outra coisa importante a se observar é que o símbolo de igual ali nos exemplos não está na sua função usual de denotar igualdade, mas sim do que chamamos de **atribuição.** Ele está indicando que aquela variável está recebendo aquele valor!

Nos nossos algoritmos e programas em papel, utilizaremos uma seta para a esquerda para indicar atribuição. Exemplo:

text **<-** “This is a string 123”

```
String text = "This is a string 123";

int integer = 22

double precise = 3.1415926535912154548

bool boolean = true;
```

```
a = 'a'
number = 12
decimal = 12.3
double = 12.123123123123123
string = "now or never"
boolean = True
```

O que são pseudo-linguagens?

Portugol

Início

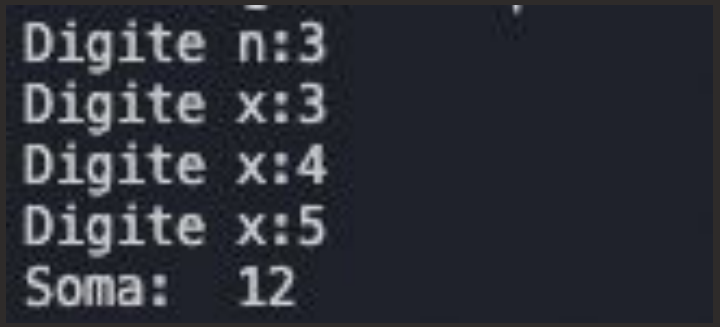
1. **Ler** (n)
2. S **<-** 0
3. i **<-** 1
4. **Enquanto** i <= n **Faça**
5. **Ler**(x)
6. S **<-** S + x
7. i **<-** i + 1
8. **Fim Enquanto**
9. **Exibir**('Soma: ', S)

Fim

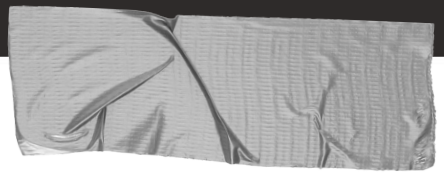
Python

```python
n = int(input("Digite n:"))
s = 0
i = 1
while (i <= n):
    x = int(input('Digite x:'))
    s = s + x
    i = i + 1
print("Soma: ", s)
```

Este seria o output deste programa.



O que são operadores?



Operadores

Os operadores são meios pelo qual incrementamos, decrementamos, comparamos e avaliamos dados dentro do computador.

Temos três tipos de operadores:

- **Operadores Aritméticos**
- **Operadores Relacionais**
- **Operadores Lógicos**

Operadores Aritméticos

Os operadores aritméticos são os utilizados para obter resultados numéricos.

Além da **adição, subtração, multiplicação e divisão,** podem utilizar também o operador para **exponenciação**.

OPERAÇÃO	SÍMBOLO
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Exponenciação	**

Hierarquia das Operações Aritméticas

1 º ( ) Parênteses
2 º Exponenciação
3 º Multiplicação, divisão (o que aparecer primeiro)
4 º + ou – (o que aparecer primeiro)

Exemplo

TOTAL = PRECO * QUANTIDADE

1 + 7 * 2 ** 2 –1 = 28

3 * (1 – 2) + 4 * 2 = 5

Operadores Relacionais

Os operadores relacionais são utilizados para comparar String de caracteres e números. Os valores a serem comparados podem ser caracteres ou variáveis.

Estes operadores sempre retornam valores lógicos (verdadeiro ou falso / True ou False).

Para estabelecer prioridades no que diz respeito a qual operação executar primeiro, utilize os parênteses.

Descrição	Símbolo
Igual a	=
Diferente de	<> ou # ou !=
Maior que	>
Menor que	<
Maior ou igual a	>=
Menor ou igual a	<=

Tendo duas variáveis A = 5 e B = 3

Os resultados das expressões seriam:

Expressão	Resultado
A = B	Falso
A <> B	Verdadeiro
A > B	Verdadeiro
A < B	Falso
A >= B	Verdadeiro
A <= B	Falso

Operadores Lógicos

Os operadores lógicos servem para combinar resultados de expressões, retornando se o resultado final é verdadeiro ou falso.

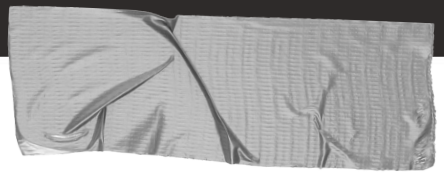
E / AND: Uma expressão AND (E) é verdadeira se todas as condições forem verdadeiras

OR/OU: Uma expressão OR (OU) é verdadeira se pelo menos uma condição for verdadeira

NOT/NÃO: Uma expressão NOT (NÃO) inverte o valor da expressão ou condição, se verdadeira inverte para falsa e vice-versa.

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

Estrutura de decisão e repetição



Comandos de Decisão

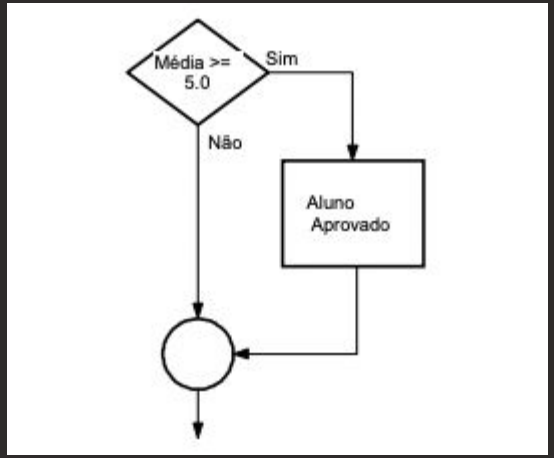
Os comandos de decisão ou desvio fazem parte das técnicas de programação que conduzem a estruturas de programas que não são totalmente sequenciais.

Com as instruções de SALTO ou DESVIO pode-se fazer com que o programa proceda de uma ou outra maneira, de acordo com as decisões lógicas tomadas em função dos dados ou resultados anteriores. As principais estruturas de decisão são: “**Se Então**”, “**Se Então Senão**” e “**Caso Selecione**”.

SE - ENTÃO

A estrutura de decisão “SE/IF” normalmente vem acompanhada de um comando, ou seja, se determinada condição for satisfeita pelo comando SE/IF então execute determinado comando.
Imagine um algoritmo que determinado aluno somente estará aprovado se sua média for maior ou igual a 5.0, veja no exemplo de algoritmo como ficaria.

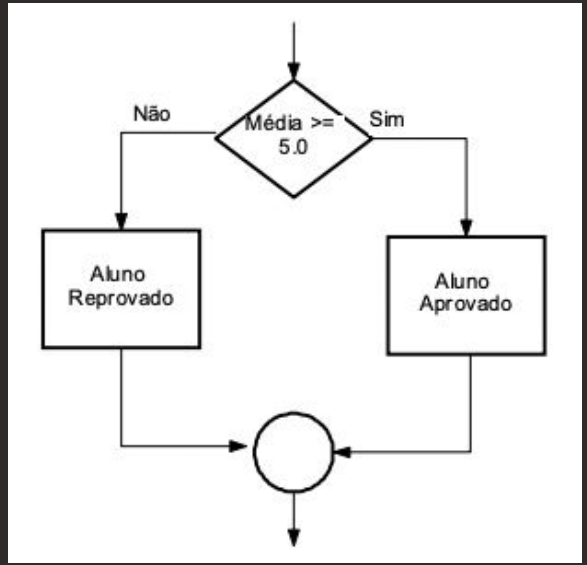
SE MEDIA >= 5.0 **ENTÃO** ALUNO APROVADO



SE - ENTÃO - SENÃO

A estrutura de decisão “SE/ENTÃO/SENÃO”, funciona exatamente como a estrutura “SE”, com apenas uma diferença, em “SE” somente podemos executar comandos caso a condição seja verdadeira, diferente de “SE/SENÃO” pois sempre um comando será executado independente da condição, ou seja, caso a condição seja “verdadeira” o comando da condição será executado, caso contrário o comando da condição “falsa” será executado.

SE MEDIA >= 5.0 **ENTÃO**
    ALUNO APROVADO
SENÃO
    ALUNO REPROVADO

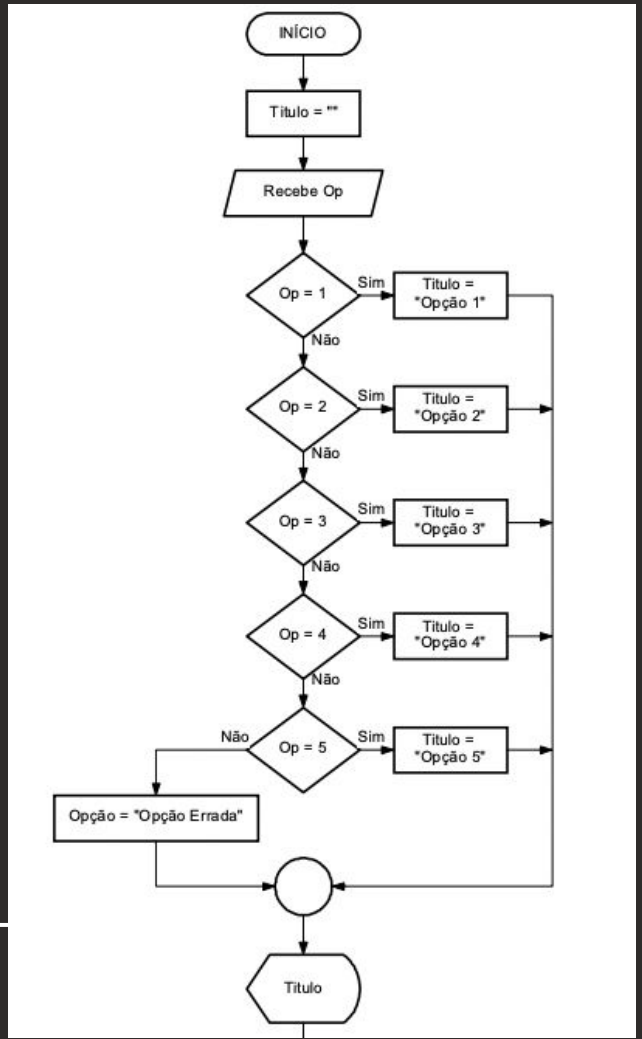


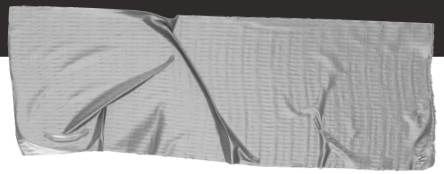
CASO - SELECIONE

A estrutura de decisão CASO/SELECIONE é utilizada para testar, na condição, uma única expressão, que produz um resultado, ou, então, o valor de uma variável, em que está armazenado um determinado conteúdo. Compara-se, então, o resultado obtido no teste com os valores fornecidos em cada cláusula "Caso".

```
TITULO = ""
OP = INPUTBOX("DIGITE A OPÇÃO")
SELECT CASE OP
        CASE 1
            TITULO = "OPÇÃO 1"
        CASE 2
            TITULO = "OPÇÃO 2"
        CASE 3
            TITULO = "OPÇÃO 3"
        CASE 4
            TITULO = "OPÇÃO 4"
        CASE 5
             TITULO = "OPÇÃO 5"
        CASE ELSE
             TITULO = "OPÇÃO ERRADA"
END SELECT

LABEL1.CAPTION = TITULO
```





Comandos de Repetição

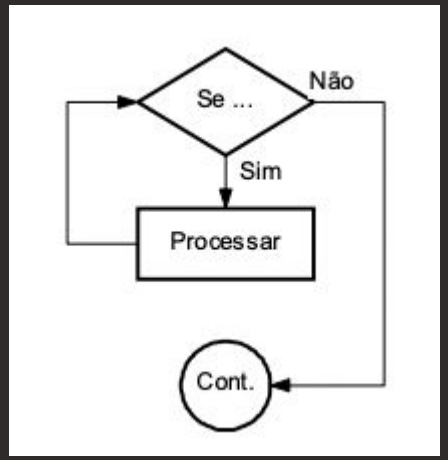
Utilizamos os comandos de repetição quando desejamos que um determinado conjunto de instruções ou comandos sejam executados um número definido ou indefinido de vezes, ou enquanto um determinado estado de coisas prevalecer ou até que seja alcançado. Veremos alguns modelos de comandos de repetição:

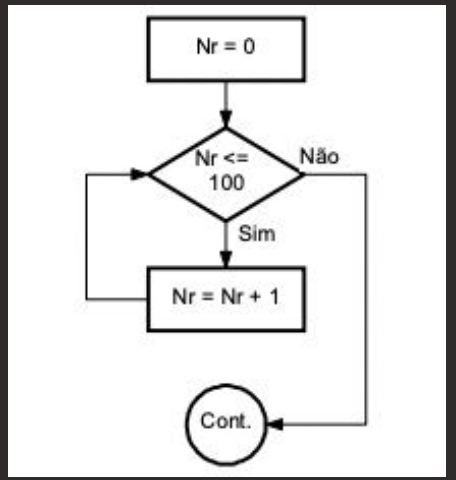
- **Enquanto** ..., processar (While ... Loop);
- **Até que** ..., processar ... (Do Until ... Loop);
- **Processar** ..., **Enquanto** x (Do ... Loop While);
- **Processar** ..., **Até que** x (Do ... Loop Until)
- **Para** ... **Até** ... **Seguinte** (For ... To ... Next)

Enquanto x, Processar (While ... Loop)

Neste caso, o bloco de operações será executado enquanto a condição x for verdadeira. O teste da condição será sempre realizado antes de qualquer operação. Enquanto a condição for verdadeira o processo se repete. Podemos utilizar essa estrutura para trabalharmos com contadores.

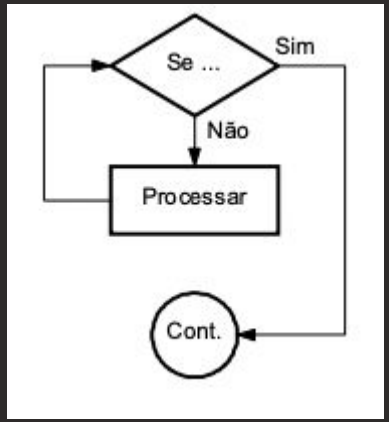
```
Nr = 0
Do While Nr <= 100
    Nr = Nr + 1
Loop
```

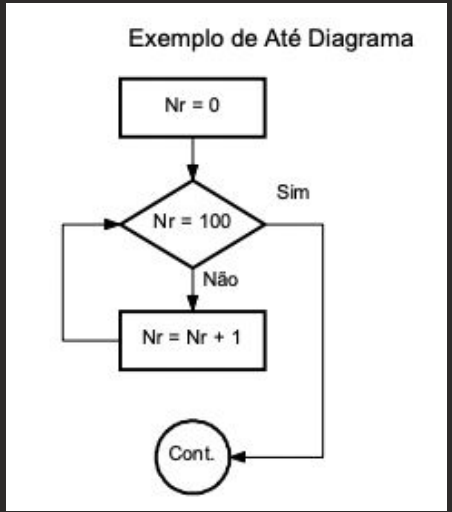




Até que x, processar ... (Do Until ... Loop)

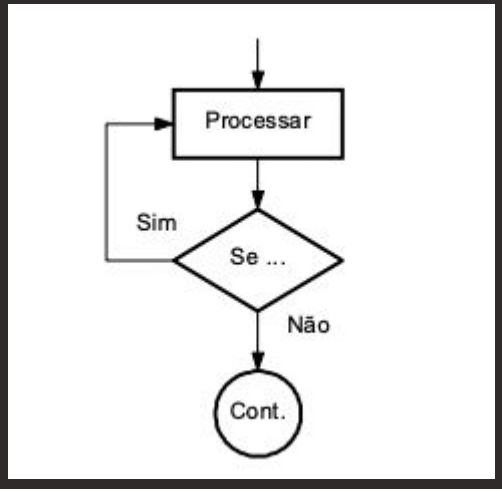
Neste caso, o bloco de operações será executado até que a condição seja satisfeita, ou seja, somente executará os comandos enquanto a condição for falsa.

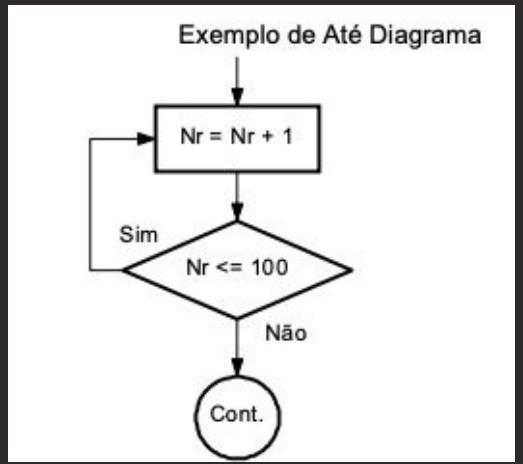




Processar ..., Enquanto x (Do ... Loop While)

Neste caso primeiro são executados os comandos, e somente depois é realizado o teste da condição. Se a condição for verdadeira, os comandos são executados novamente, caso seja falso é encerrado o comando DO.





Processar ..., Até que x (Do ... Loop Until)

Neste caso, executa-se primeiro o bloco de operações e somente depois é realizado o teste de condição. Se a condição for verdadeira, o fluxo do programa continua normalmente. Caso contrário é processado novamente os comandos antes do teste da condição.