CURSO TÉCNICO DE
GESTÃO E PROGRAMAÇÃO DE SISTEMAS INFORMÁTICOS

Disciplina de Programação de Sistemas Informáticos

Módulo 1 – Introdução à Programação e Algoritmia

João Duarte
jduarte@esrbp.pt

ALGORITMO

Algoritmo - Uma sequência de ações finitas encadeadas e lógicas que descrevem como um determinado problema deve ser resolvido.

Um algoritmo é formalmente uma sequência finita de passos que levam a execução de uma tarefa. Podemos pensar em algoritmo como uma receita, uma sequência de instruções que executam uma meta específica. Estas tarefas não podem ser redundantes nem subjetivas na sua definição, devem ser claras e precisas.

Como exemplos de algoritmos podemos citar os algoritmos das operações básicas (adição, multiplicação, divisão e subtração) de números reais decimais. Outros exemplos seriam os manuais de aparelhos eletrônicos, como um aparelho de som, que explicam passo-a-passo como, por exemplo, gravar um evento.

Apesar do nome pouco habitual, os algoritmos são comuns em nosso cotidiano, como por exemplo, a receita de um bolo. Nela está descrita uma série de ingredientes necessários e uma sequência de diversos passos (ações) que devem ser fielmente cumpridos para que se consiga fazer o alimento desejado, conforme se esperava, antes do início das atividades (objetivo bem definido).

Observa-se, porém que uma ordem isolada não permite realizar o processo completo, para isso é necessário um conjunto de instruções colocadas em ordem sequencial lógica. No exemplo do parágrafo anterior para fazermos um bolo não podemos começar por colocar os ingredientes no forno. É necessário todo um processo passo a passo para se chegar a este fim.

Formas de Representação de Algoritmos

Existem diversas formas de representação de algoritmos, mas não há um consenso em relação à melhor forma delas.

O critério usado para classificar hierarquicamente estas formas está diretamente ligado ao nível de detalhe, inversamente ao grau de abstração oferecido.

Algumas formas de representação de algoritmos tratam os problemas apenas em nível lógico, abstraindo-se de detalhes de implementação muitas vezes relacionados com alguma linguagem de programação específica. Por outro lado, existem formas de representação de algoritmos que possuem uma maior riqueza de detalhes e muitas vezes acabam por obscurecer a ideia principal, o algoritmo, dificultando seu entendimento.

Dentre as formas de representação de algoritmo mais conhecidas, destacam-se:

- A descrição narrativa;
- O fluxograma convencional;
- O pseudocodigo, também conhecido como linguagem estruturada ou Português Estruturado.

Descrição Narrativa

Nesta forma de representação, os algoritmos são expressos em linguagem natural.

Exemplo:

"Receita de um bolo"

1. Separar os ingredientes
2. Bater os ovos em neve na batedeira
3. Acrescentar açúcar e farinha de trigo
4. Colocar extrato de baunilha
5. Acrescentar uma colher de manteiga
6. Acrescentar uma colher de Fermento em pó
7. Verificar se esta doce o suficiente
8. Colocar na forma
9. Colocar no forno e assar
10. Retirar do forno
11. Tirar da forma e servir
12. Fim do processo

Quando elaboramos um algoritmo devemos especificar ações claras e precisas, que a partir de um estado inicial, após um período de tempo finito, produzem um estado final previsível e bem

definido. Isto significa que o algoritmo fixa um padrão de comportamento a ser seguido, uma norma de execução a ser trilhada, com vistas a alcançar, como resultado final, a solução de um problema, garantindo que sempre que executado, sob as mesmas condições, produza o mesmo resultado.

A importância de se construir um algoritmo: conseguimos visualizar e testar ainda no papel, a solução criada com lógica de programação sem nos preocupar com detalhes computacionais e uma vez concebida uma solução algorítmica para um problema, esta pode ser traduzida facilmente para qualquer linguagem de programação e ser agregada das funcionalidades disponíveis nos diversos ambientes, ou seja, a codificação.

Pseudocódigo

Como foi visto até agora, o fluxograma convencional é a primeira forma de notação gráfica, mas existe outra, que é uma técnica narrativa denominada pseudocódigo, também conhecida como português estruturado ou chamada por alguns de **portugol**.

Esta técnica de algoritmização é baseada em uma PDL – Program Design Language (Linguagem de Projeto de Programação). Aqui vamos apresentá-la em português. A forma originas de escrita é conhecida como inglês estruturado, muito parecida com a notação da linguagem PASCAL. A PDL (neste caso, o pseudocódigo) é usada como referência genérica para uma linguagem de projeto de programação, tendo como finalidade mostrar uma notação para elaboração de algoritmos, os quais serão utilizados na definição, criação e desenvolvimento de uma linguagem computacional (Clipper, C, Fortran, Pascal, Delphi, Visual-Objects) e sua documentação. Abaixo é apresentado um exemplo deste tipo de algoritmo.

Os algoritmos são independentes das linguagens de programação. Ao contrário de uma linguagem de programação não existe um formalismo rígido de como deve ser escrito o algoritmo.

O algoritmo deve ser fácil de se interpretar e fácil de codificar. Ou seja, ele deve ser o intermediário entre a linguagem falada e a linguagem de programação.

Portugol

A maioria esmagadora das linguagens de programação de computadores é em língua inglesa. Para facilitar o aprendizado de lógica de programação foram criadas algumas pseudolinguagens.

O Portugol é uma pseudolinguagem de programação, uma simbiose de Português, Algol e Pascal, criada originalmente em inglês, com a proposta de ser independente da linguagem nativa (ou seja, existe em japonês, javanês, russa...).

Basicamente, é uma notação para algoritmos, a ser utilizada na ***definição***, ***criação***, ***desenvolvimento*** e ***documentação*** dos programas.

Algumas Palavras Chave

Inicio, Fim, Algoritmo, Enquanto, Se, então, Para, Até, Até que, Leia, Escreva, Faça, Repita, FimSe, FimEnquanto, FimSelecione, entre outras que veremos adiante.

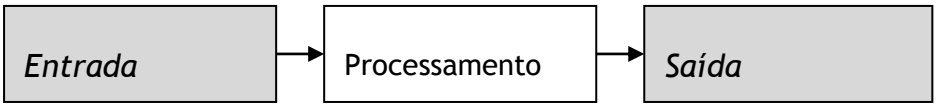
Regras para construção do Algoritmo

Para escrever um algoritmo precisamos descrever a sequência de instruções, de maneira simples e objetiva. Para isso utilizaremos algumas técnicas:

1. Usar somente um verbo por frase
2. Imaginar que você está desenvolvendo um algoritmo para pessoas que não trabalham com informática
3. Usar frases curtas e simples
4. Ser objetivo
5. Evite usar palavras que tenham sentido dúbio
6. Procure dividir o problema em etapas menores.

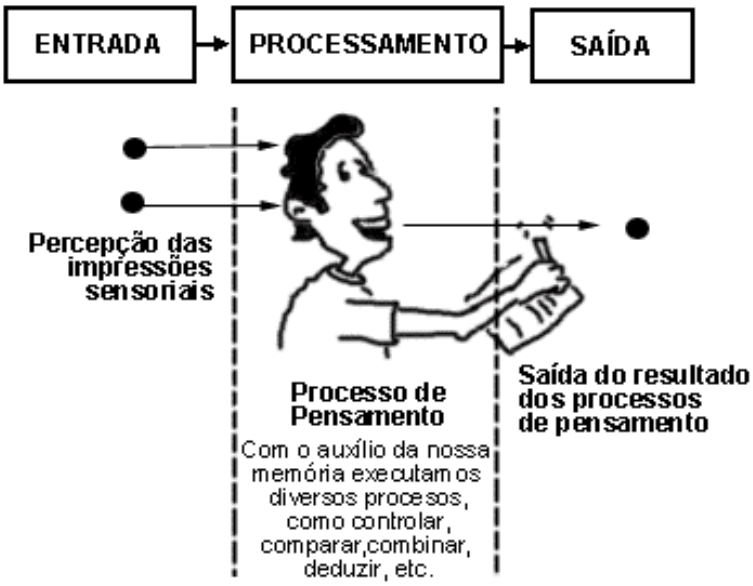
Fases Fundamentais

Vimos que ALGORITMO é uma sequência lógica de instruções que podem ser executadas.
É importante ressaltar que qualquer tarefa que siga comportamento padrão pode ser descrita por um algoritmo.
Entretanto ao montar um algoritmo, precisamos primeiro dividir o problema apresentado em três fases fundamentais...



***ENTRADA*:** São os dados de entrada do algoritmo
***PROCESSAMENTO*:** Procedimentos utilizados para chegar ao resultado final
***SAÍDA*:** São os dados já processados

Analogia com o homem



Exemplo de Algoritmo

Imagine o seguinte problema: Calcular a média final dos alunos. Pelas provas os alunos receberão 2 notas: N1, N2. Onde:

$$\text{Média Final} = \frac{N1 + N2}{2}$$

Para montar o algoritmo proposto, faremos três perguntas:

a) Quais são os dados de entrada?	**R:** Os dados de entrada são N1, N2
b) Qual será o processamento a ser utilizado?	**R:** O procedimento será somar todos os dados de entrada e dividi-los por 2 (dois)
c) Quais serão os dados de saída?	**R:** O dado de saída será a média final

INICIO do algoritmo
- Receba a nota da prova1
- Receba a nota de prova2
- Some todas as notas e divida o resultado por 2
- Mostre o resultado da divisão

FIM do algoritmo

Teste de Mesa

Após desenvolver um algoritmo ele deverá sempre ser testado. Este teste é chamado de **TESTE DE MESA**, que significa, seguir as instruções do algoritmo de maneira precisa para verificar se o procedimento utilizado está correto ou não, escrevendo todas as variáveis e resultados numa tabela.

```
Programa média
Var
        Nome: caractere
        N1, N2: real
        Soma, média: real
Início
        Imprima "Informe nome e notas do aluno:"
        Leia nome
        Leia n1
        Leia n2
        Soma <- n1 + n2
        Média ← soma / 2
        Imprima nome
        Imprima Média
        Se média >=5  Então
            Imprima "Aprovado"
        Caso contrário
            Imprima "Reprovado"
        Fim – Se
Fim
```

A diferença entre uma linguagem de programação de alto nível utilizada em computação e uma PDL é que esta (seja escrita em português, inglês, ou qualquer outro idioma) não pode ser compilada em um computador (por enquanto). Porém, existem "Processadores de PDL" que possibilitam traduzir essa linguagem numa representação gráfica de projeto, fornecendo uma variedade de informações, como: tabelas de referência cruzada, mapas de aninhamento, índice operacional do projeto, entre tantas outras.

ATENÇÃO: Lembramos que o fluxograma e o pseudocódigo são as duas técnicas importantes para a documentação da solução de um problema computacional.

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. Os símbolos para os operadores aritméticos podem ser vistos ao lado:

Operação	Símbolo
Adição	**+**
Subtração	**-**
Multiplicação	*****
Divisão	**/**
Exponenciação	** ou ^

Hierarquia das Operações Aritméticas

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

Exemplos:

- TOTAL = PREÇO * 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.

Igual a	**=**
Diferente de	**<>** ou ≠
Maior que	**>**
Menor que	**<**
Maior ou igual a	**>=**
Menor ou igual a	**<=**

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. Os operadores relacionais são:

Exemplos

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.

Os operadores lógicos são:

E (And) A expressão é verdadeira se todas as condições forem verdadeiras

OU (Or) A expressão é verdadeira se pelo menos uma condição for verdadeira

NÃO (Not) Inverte o valor da expressão ou condição, se verdadeira inverte para falsa e vice-versa.

A tabela mostra todos os valores possíveis criados pelos três operadores lógicos.

Exemplos

Suponha que temos três variáveis A = 5, B = 8 e C =1. Os resultados das expressões seriam:

Expressões			*Resultado*
A = B	AND	B > C	Falso
A <> B	OR	B < C	Verdadeiro
A > B	NOT		Verdadeiro
A < B	AND	B > C	Verdadeiro
A >= B	OR	B = C	Falso
A <= B	NOT		Falso.

Constantes, Variáveis e Tipos de Variáveis

Variáveis e **constantes** são os elementos básicos que um programa manipula. Uma variável é um espaço reservado na memória (endereço de memória) do computador para armazenar um conteúdo de determinado tipo, variável durante a execução do algoritmo.

Variáveis devem receber nomes para poderem ser referenciadas e modificadas quando necessário. Um programa deve conter declarações que especificam de que tipo são as variáveis que ele utilizará e as vezes um valor inicial. Tipos podem ser por exemplo: inteiros, reais, caracteres, etc. 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.

Exemplos de variáveis:

Nome
Media = Soma / 20

→ Nome é o nome dado a uma posição de memória para armazenar um conteúdo com letras
→ Media é o nome dado a uma posição de memória para armazenar um valor real
→ Soma e o nome dado a uma posição de memória para armazenar um valor real

Constantes

Constante é uma determinada variável que possui um valor fixo (que não se modifica) durante a execução de um algoritmo.

Conforme o seu tipo, a constante é classificada como sendo numérica, lógica e literal.

Exemplos de constante:
Nome = "Jose da Silva"
Peso= 85

Tipos de Variáveis

As variáveis e as constantes podem ser basicamente de seis tipos:

Numérica	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 (números fracionários).
Caractere	Específicas para armazenamento de conjunto de caracteres que não contenham números (literais). Ex: nomes.
Alfanumérica	(Expressões) 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 expressões (operações matemáticas).
Lógica	Armazenam somente dados lógicos que podem ser Verdadeiro ou Falso.
Indexada	Armazenam variáveis uni e multidimensionais
Utilizador	Utilizada para variáveis não básicas. São definidas através da instrução TIPO

Declaração e Atribuição de Variáveis

As variáveis para utilizadas no programa devem ser **declaradas** (criadas) logo no início do algoritmo, na seção **DECLARAÇÕES**, quando será reservado espaço na memória. Só podem armazenar valores de um mesmo tipo.

Todas as variáveis quando são criadas tem um conteúdo **nulo** (ou seja, nenhum conteúdo).

Utilizamos o comando **ATRIBUIÇÃO** (←), para atribuir valor a uma variável, bem como inicializá-la com um determinado valor.

O sentido a seta é sempre da direita para a esquerda, e deve ser entendida como : ... **um valor** ATRIBUIDO a uma **variável**.

Estrutura de Um Algoritmo

```
ALGORITMO ("Nome do algoritmo" )
{Declarações}
    Nome : caracter  { a variável Nome é criada como caracter }
    Idade : inteiro     { a variável Idade é número inteiro }
{Atribuições}
    Nome ← "José da Silva"
    Peso   ← 85
  INICIO (Nome do bloco)       { se for o caso, de refinamento }
{Instruções}
    < comandos >
  FIM
```

Nome ← "José da Silva" (A variável "nome" recebe valor "José da silva")
Peso ← 85 (A variável "Peso" recebe valor igual a 85)

Obs.: As seções Declarações e Atribuições são PÚBLICAS dentro do programa. ou seja são "vistas" em qualquer parte do programa. Dentro do Bloco, as seções Declarações e Atribuições são LOCAIS ao Bloco, ou seja são privadas (particulares, internas) ao Bloco, conforme veremos em Refinamentos Sucessivos.

Fluxograma Convencional

Sabemos que uma figura fala por mil palavras. No processo de aprendizagem fixamos com mais facilidade imagens do que conceitos escritos.

O **diagrama de blocos ou fluxograma** é uma forma padronizada eficaz para **representar** os passos lógicos de um determinado processamento.

Com o diagrama podemos definir uma sequência de símbolos, com significado bem definido. Portanto, a sua principal função é facilitar a visualização dos passos de um processamento.

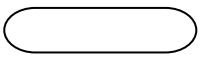
O fluxograma é uma ferramenta usada e desenvolvida pelos profissionais de análise de sistemas, bem como, por alguns profissionais de Organização, Sistemas e Métodos. Tem como finalidade descrever o fluxo seja manual ou mecânico, especificando os suportes usados para os dados e informações. Usa símbolos convencionais, permitindo poucas variações. Representado por alguns desenhos geométricos básicos, os quais indicarão os símbolos de entrada de dados, do processamento de dados e da saída de dados, acompanhados dos procedimentos requeridos pelo analista de sistemas e a serem realizados pelo programador por meio do desenvolvimento do raciocínio lógico, o qual deverá solucionar o problema do programa a ser processado pelo computador.

É uma ferramenta de uso em diversas áreas do conhecimento humano, por traduzir em formato gráfico algum procedimento ou norma.

Simbologia

Existem diversos símbolos em um diagrama de bloco. No decorrer do curso apresentaremos os mais utilizados. Veja no quadro abaixo alguns dos símbolos que iremos utilizar:

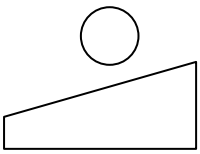
FLUXO DE DADOS
Indica o sentido do fluxo de dados. Conecta os demais símbolos



TERMINAL
Indica o INÍCIO ou FIM de um processamento
Exemplo: Início do algoritmo

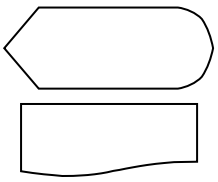
PROCESSAMENTO
Processamento em geral
Exemplo: Calculo de dois números

ENTRADA/SAÍDA (Genérica)
Operação de entrada e saída de dados
Exemplo: Leitura e Gravação de Arquivos

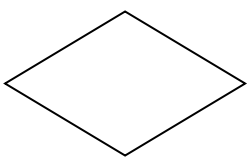


DESVIO (conector)
Permite o desvio para um ponto qualquer do programa

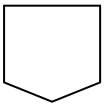
ENTRADA MANUAL
Indica entrada de dados via Teclado
Exemplo: Digite a nota da prova 1



EXIBIR/SAÍDA
Mostra informações ou resultados
Exemplo: Mostre o resultado do cálculo



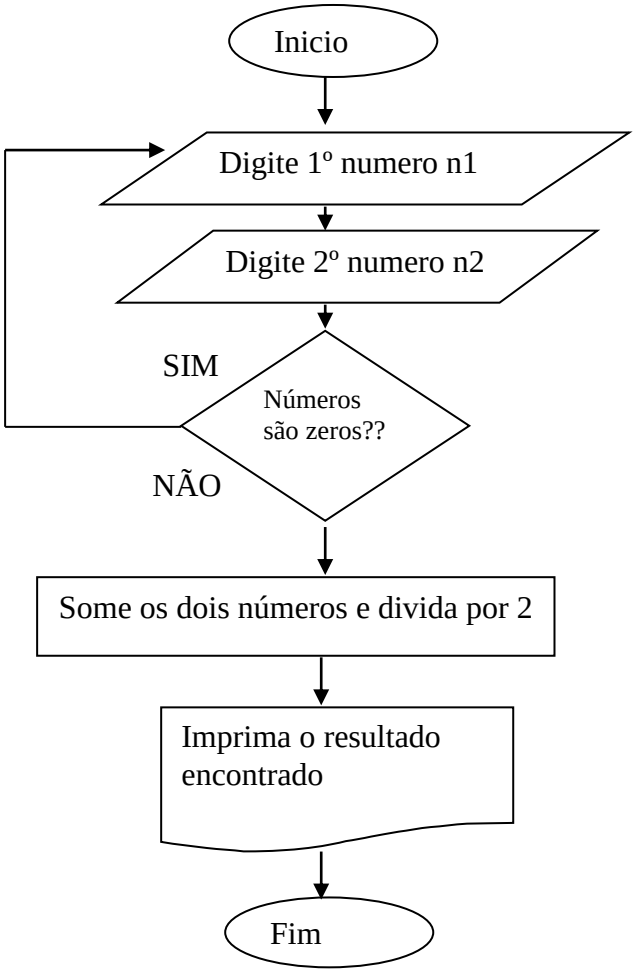
DECISÃO
Permite elaborar processos de decisão



CONECTOR DE PAGINA
Permite informar de qual pagina vem o fluxograma

Dentro do símbolo terá sempre algo escrito, pois somente os símbolos não nos dizem nada. Veja no exemplo a seguir:.

Exemplo: Fluxograma de um programa para ler dois números aleatórios diferentes de zero, calcular a média dos mesmos e mostrar o resultado encontrado.



Estruturas Básicas de Controle

Na maioria das vezes necessitamos tomar decisões no andamento do algoritmo. Essas decisões interferem diretamente no andamento do programa. Trabalharemos com dois tipos de estrutura. A estrutura de **Decisão** e a estrutura de **Repetição**

Decisão

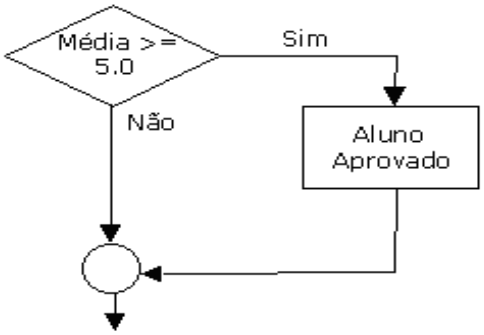
Os comandos de decisão ou desvio fazem parte das técnicas de programação que conduzem a estruturas de programas que não são totalmente sequenciais. Com as instruções de SALTO ou DESVIO pode-se fazer com que o programa proceda de uma ou outra maneira, de acordo com as decisões lógicas tomadas em função dos dados ou resultados anteriores. As principais estruturas de decisão são: ***"Se Então"***, ***"Se então Senão"*** e ***"Selecion*e *Caso"***

SE ENTÃO / IF ... THEN

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 o exemplo abaixo, um algoritmo que determinado aluno somente estará aprovado se sua média for maior ou igual a 5.0, o diagrama de blocos ficaria conforme ao lado.

SE Media **>=** 5.0 **ENTÃO "**aluno Aprovado**"**

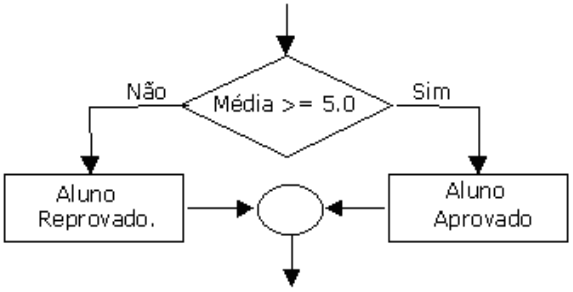


SE ENTÃO SENÃO / IF ... THEN ... ELSE

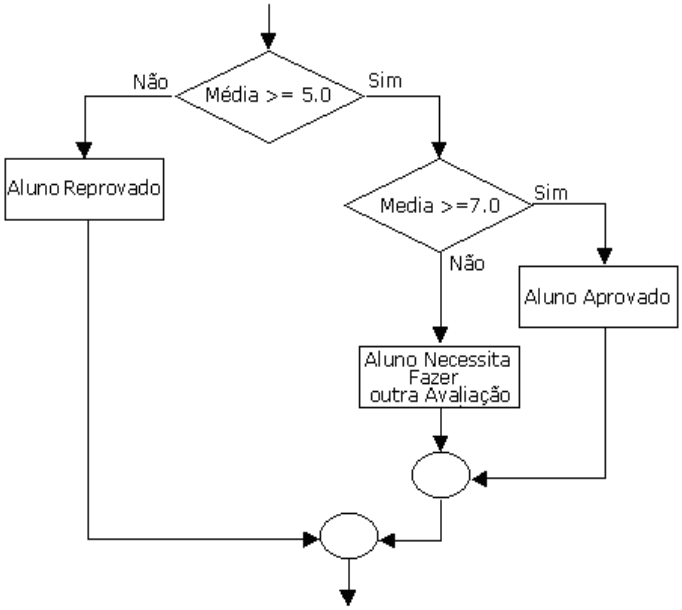
A estrutura de decisão "SE/ENTÃO/SENÃO", funciona exatamente como a estrutura "SE", com apenas uma diferença, em "SE" somente podemos executar comandos caso a condição seja verdadeira, diferente de "SE/SENÃO" pois sempre um comando será executado independente da condição, ou seja, caso a condição seja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

Em algoritmo ficaria assim:

SE Média >= 5.0 **ENTÃO**
    "aluno Aprovado**"**
SENÃO
    "aluno Reprovado**"**
FimSe



No exemplo acima foi executada uma condição que, se for verdadeira, executa o comando "APROVADO", caso contrário executa o segundo comando "REPROVADO". Podemos também dentro de uma mesma condição testar outras condições. Como por exemplo:

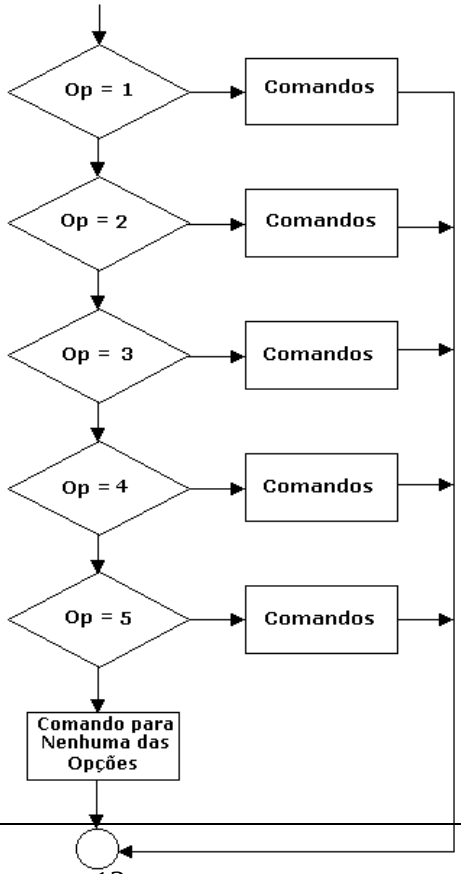


SELECIONE CASO / SELECT ... CASE

A estrutura de decisão SELECIONE/CASO 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".

No exemplo do diagrama de blocos abaixo, é recebido uma variável **"Op"** e testado seu conteúdo, caso uma das condições seja satisfeita, é executado o bloco de comandos apropriado, caso contrário é executado a opção de comandos de **caso senão**.

SELECIONE CASO Op
CASO 1
<comandos>
CASO 2
<comandos>
CASO 3
<comandos>
CASO 4
<comandos>
CASO 5
<comandos>
CASO SENÃO
<comandos>
FimSelecione



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.

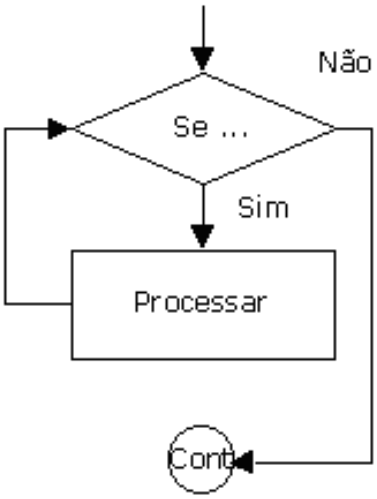
Trabalharemos com modelos de comandos de repetição:

- Enquanto <condição>, faça (Do While ...Loop);
- Até que <condição>, faça (Do Until ... Loop);
- Repita, ...Enquanto <condição> (Do ... Loop While);
- Repita, ...Até que <condição> (Do ... Loop Until);
- Para ...de..., até...passo...faça...seguinte (For ... To ... Next)

Enquanto <condição>, Faça (Do 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. Em diagrama de bloco:

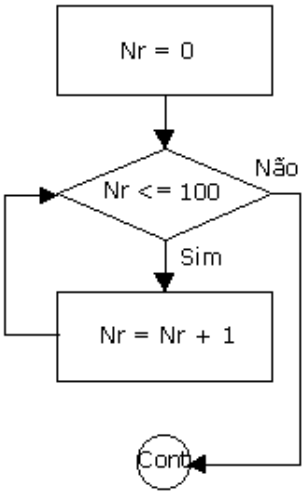


Exemplo de Contador

Enquanto <condição> **Faça**

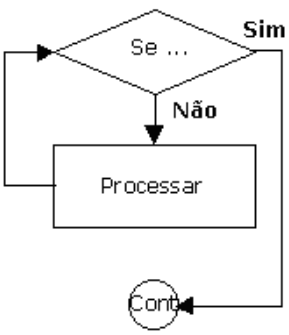
<Comandos>

FimEnquanto



Até que <condição>, Faça ... (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. Em diagrama de blocos:

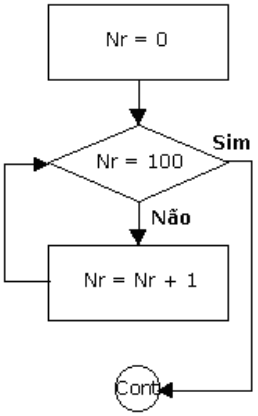


Exemplo de Até Diagrama

Até <condição> **Faça**

<Comandos>

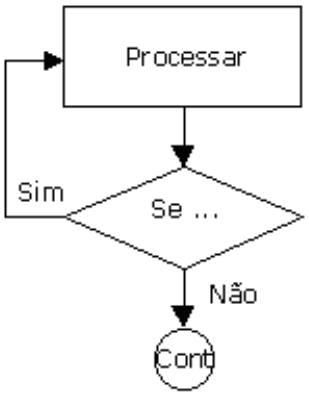
FimAté



Repita ..., Enquanto <condição> (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. Em diagrama de bloco

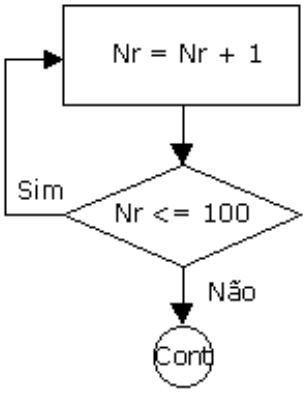
Exemplo de Enquanto Diagrama



Repita

<Comandos>

Enquanto <condição>

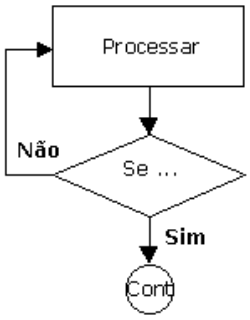


Repita ..., Até que <condição> (Do ... Loop Until)

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.

Em diagrama de Bloco

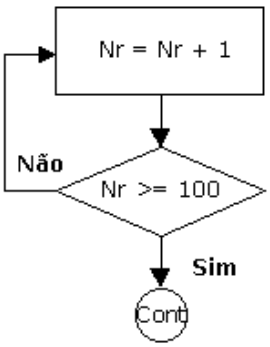
Exemplo de Do .... Loop - Until



Repita

<Comandos>

Até que <condição>



Para ...de..., até...passo...faça...seguinte (For...To...Next)

Tem seu funcionamento controlado por uma variável chamada contador. Sendo assim, executará um determinado conjunto de instruções, um determinado numero de vezes, a passos (intervalos) controlados (de 2 em 2, de 3 em 3 ...)

PARA <contador> **DE** <inicio> **ATÉ** <fim> **PASSO** <incremento> **FAÇA**
<comandos>

SEGUINTE