Lógica

de

Programação

Autor: Paulo Sérgio de Moraes

Ultima Atualização: 04 de Abril de 2000

INDICE

1 Introdução à Lógica de Programação

1.1 Lógica

A lógica de programação é necessária para pessoas que desejam trabalhar com desenvolvimento de sistemas e programas, ela permite definir a seqüência lógica para o desenvolvimento.

Então o que é lógica?

Lógica de programação é a técnica de encadear pensamentos para atingir determinado objetivo.

1.2 Seqüência Lógica

Estes pensamentos, podem ser descritos como uma seqüência de instruções, que devem ser seguidas para se cumprir uma determinada tarefa.

Seqüência Lógica são passos executados até atingir um objetivo ou solução de um problema.

1.3 Instruções

Na linguagem comum, entende-se por instruções "**um conjunto de regras ou normas definidas para a realização ou emprego de algo**".

Em informática, porém, instrução é a informação que indica a um computador uma ação elementar a executar.

Convém ressaltar que uma ordem isolada não permite realizar o processo completo, para isso é necessário um conjunto de instruções colocadas em ordem seqüencial lógica.

Por exemplo, se quisermos fazer uma omelete de batatas, precisaremos colocar em prática uma série de instruções: descascar as batatas, bater os ovos, fritar as batatas, etc...

É evidente que essas instruções tem que ser executadas em uma ordem adequada – não se pode descascar as batatas depois de fritá-las.

Dessa maneira, uma instrução tomada em separado não tem muito sentido; para obtermos o resultado, precisamos colocar em prática o conjunto de todas as instruções, na ordem correta.

Instruções são um conjunto de regras ou normas definidas para a realização ou emprego de algo. Em informática, é o que indica a um computador uma ação elementar a executar.

1.4 Algoritmo

Um algoritmo é formalmente uma seqüência finita de passos que levam a execução de uma tarefa. Podemos pensar em algoritmo como uma receita, uma seqüência de instruções que dão cabo de 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 videocassete, que explicam passo-a-passo como, por exemplo, gravar um evento.

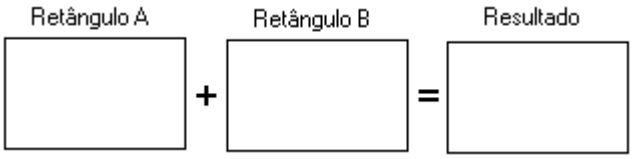
Até mesmo as coisas mais simples, podem ser descritas por seqüências lógicas. Por exemplo:

"Chupar uma bala".

- Pegar a bala
- Retirar o papel
- Chupar a bala
- Jogar o papel no lixo

"Somar dois números quaisquer".

- Escreva o primeiro número no retângulo A
- Escreva o segundo número no retângulo B
- Some o número do retângulo A com número do retângulo B e coloque o resultado no retângulo C



1.5 Programas

Os programas de computadores nada mais são do que algoritmos escritos numa linguagem de computador (Pascal, C, Cobol, Fortran, Visual Basic entre outras) e que são interpretados e executados por uma máquina, no caso um computador. Notem que dada esta interpretação rigorosa, um programa é por natureza muito específico e rígido em relação aos algoritmos da vida real.

1.6 EXERCÍCIOS

1) Crie uma seqüência lógica para tomar banho:

2) Faça um algoritmo para somar dois números e multiplicar o resultado pelo primeiro número

3) Descreva com detalhes a seqüência lógica para Trocar um pneu de um carro.

4) Faça um algoritmo para trocar uma lâmpada. Descreva com detalhes:

2 Desenvolvendo algoritmos

2.1 Pseudocódigo

Os algoritmos são descritos em uma linguagem chamada **pseudocódigo**. Este nome é uma alusão à posterior implementação em uma linguagem de programação, ou seja, quando formos programar em uma linguagem, por exemplo Visual Basic, estaremos gerando código em Visual Basic. Por isso 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.

2.2 Regras para construção do Algoritmo

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

- Usar somente um verbo por frase
- Imaginar que você está desenvolvendo um algoritmo para pessoas que não trabalham com informática
- Usar frases curtas e simples
- Ser objetivo
- Procurar usar palavras que não tenham sentido dúbio

2.3 Fases

No capítulo anterior vimos que ALGORITMO é uma seqüência lógica de instruções que podem ser executadas.

É importante ressaltar que qualquer tarefa que siga determinado padrão pode ser descrita por um algoritmo, como por exemplo:

COMO FAZER ARROZ DOCE

ou então

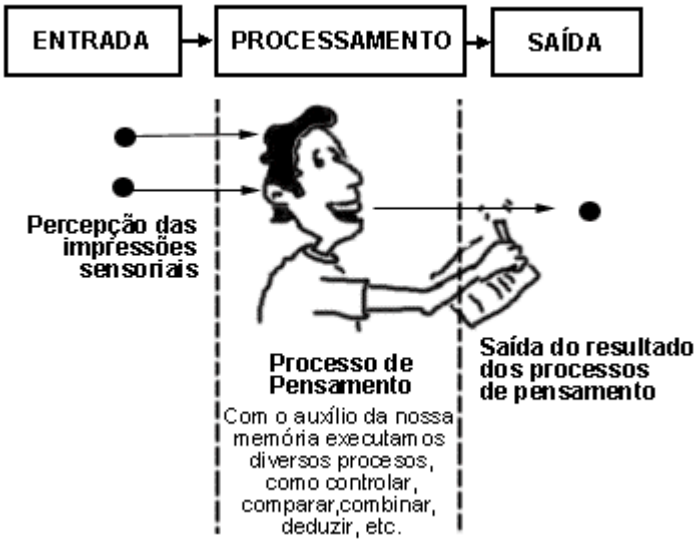
CALCULAR O SALDO FINANCEIRO DE UM ESTOQUE

Entretanto ao montar um algoritmo, precisamos primeiro dividir o problema apresentado em três fases fundamentais.

Onde temos:

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

Analogia com o homem



2.4 Exemplo de Algoritmo

Imagine o seguinte problema: Calcular a média final dos alunos da 3ª Série. Os alunos realizarão quatro provas: P1, P2, P3 e P4.

Onde:

$$\text{Média Final} = \frac{P1 + P2 + P3 + P4}{4}$$

Para montar o algoritmo proposto, faremos três perguntas:

a) Quais são os dados de entrada?

R: Os dados de entrada são P1, P2, P3 e P4

b) Qual será o processamento a ser utilizado?

R: O procedimento será somar todos os dados de entrada e dividi-los por 4 (quatro)

$$\frac{P1 + P2 + P3 + P4}{4}$$

c) Quais serão os dados de saída?

R: O dado de saída será a média final

Algoritmo

Receba a nota da prova1
Receba a nota de prova2
Receba a nota de prova3
Receba a nota da prova4
Some todas as notas e divida o resultado por 4
Mostre o resultado da divisão

2.5 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.

Veja o exemplo:

Nota da Prova 1
Nota da Prova 2
Nota da Prova 3
Nota da Prova 4

Utilize a tabela abaixo:

P1	P2	P3	P4	Média

2.6 EXERCÍCIOS

1) Identifique os dados de entrada, processamento e saída no algoritmo abaixo

   - Receba código da peça
   - Receba valor da peça
   - Receba Quantidade de peças
   - Calcule o valor total da peça (Quantidade * Valor da peça)
   - Mostre o código da peça e seu valor total

2) Faça um algoritmo para "Calcular o estoque médio de uma peça", sendo que ESTOQUEMÉDIO = (QUANTIDADE MÍNIMA + QUANTIDADE MÁXIMA) /2

3) Teste o algoritmo anterior com dados definidos por você.

3 Diagrama de Bloco

3.1 O que é um diagrama de bloco?

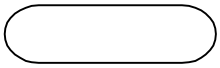
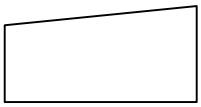
O diagrama de blocos é uma forma padronizada e eficaz para representar os passos lógicos de um determinado processamento.

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

3.2 Simbologia

Existem diversos símbolos em um diagrama de bloco. No decorrer do curso apresentaremos os mais utilizados.

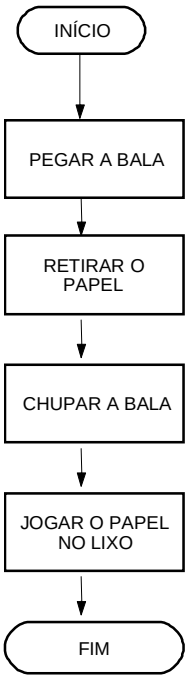
Veja no quadro abaixo alguns dos símbolos que iremos utilizar:

Símbolo	**Função**
 TERMINAL	Indica o INÍCIO ou FIM de um processamento Exemplo: Início do algoritmo
 PROCESSAMENTO	Processamento em geral Exemplo: Calculo de dois números
 ENTRADA DE DADO MANUAL	Indica entrada de dados através do Teclado Exemplo: Digite a nota da prova 1
 EXIBIR	Mostra informações ou resultados Exemplo: Mostre o resultado do calculo

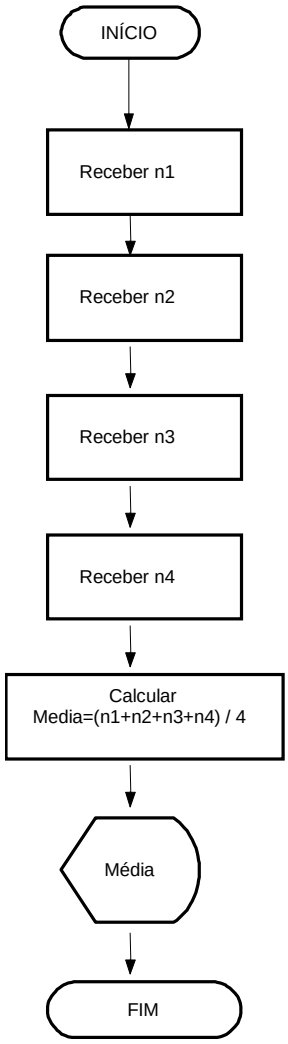
Dentro do símbolo sempre terá algo escrito, pois somente os símbolos não nos dizem nada. Veja no exemplo a seguir:

Exemplos de Diagrama de Bloco

"CHUPAR UMA BALA"



"CALCULAR A MÉDIA DE 4 NOTAS



Veja que no exemplo da bala seguimos uma seqüência lógica somente com informações diretas, já no segundo exemplo da média utilizamos cálculo e exibimos o resultado do mesmo.

3.3 EXERCÍCIOS

1) Construa um diagrama de blocos que :

   - Leia a cotação do dólar
   - Leia um valor em dólares
   - Converta esse valor para Real
   - Mostre o resultado

2) Desenvolva um diagrama que:

   - Leia 4 (quatro) números
   - Calcule o quadrado para cada um
   - Somem todos e
   - Mostre o resultado

3) Construa um algoritmo para pagamento de comissão de vendedores de peças, levando-se em consideração que sua comissão será de 5% do total da venda e que você tem os seguintes dados:

   - Identificação do vendedor
   - Código da peça
   - Preço unitário da peça
   - Quantidade vendida

   E depois construa o diagrama de blocos do algoritmo desenvolvido, e por fim faça um teste de mesa.

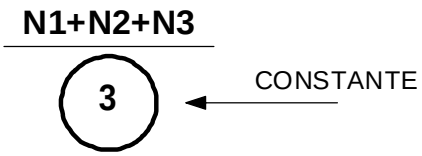
4 Constantes, Variáveis e Tipos de Dados

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. 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.

4.1 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.

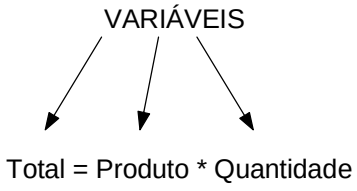
Exemplo de constantes:

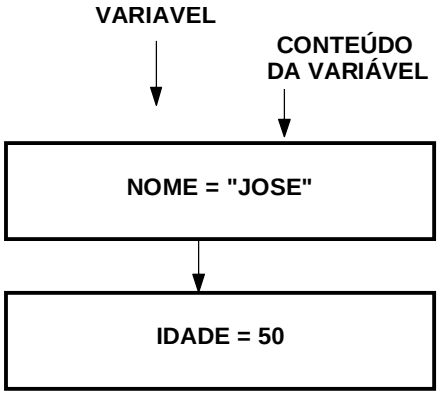


4.2 Variáveis

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





4.3 Tipos de Variáveis

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.

Caracteres Específicas para armazenamento de conjunto de caracteres que não contenham números (literais). Ex: nomes.

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.

Lógicas Armazenam somente dados lógicos que podem ser Verdadeiro ou Falso.

4.4 Declaração de Variáveis

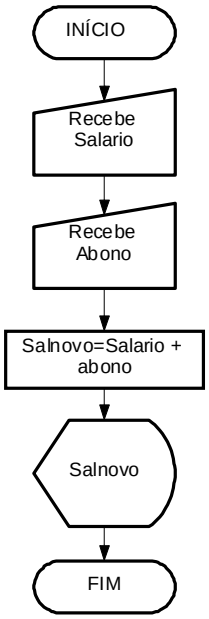
As variáveis só podem armazenar valores de um mesmo tipo, de maneira que também são classificadas como sendo numéricas, lógicas e literais.

4.5 EXERCÍCIOS

1) O que é uma constante? Dê dois exemplos.

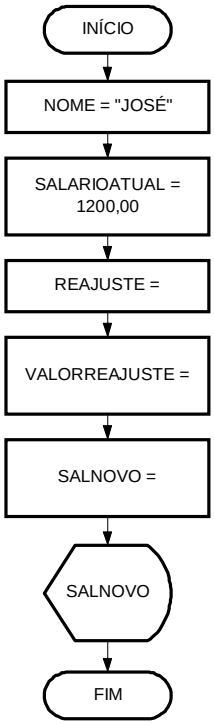
2) O que é uma variável? Dê dois exemplos.

3) Faça um teste de mesa no diagrama de bloco abaixo e preencha a tabela ao lado com os dados do teste:



Salário	Abono	Salnovo
600,00	60,00	
350,00		

4) Sabendo-se que José tem direito a 15% de reajuste de salário, complete o diagrama abaixo:



5 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

5.1 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 são:

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
```

5.2 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.

Os operadores relacionais são:

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

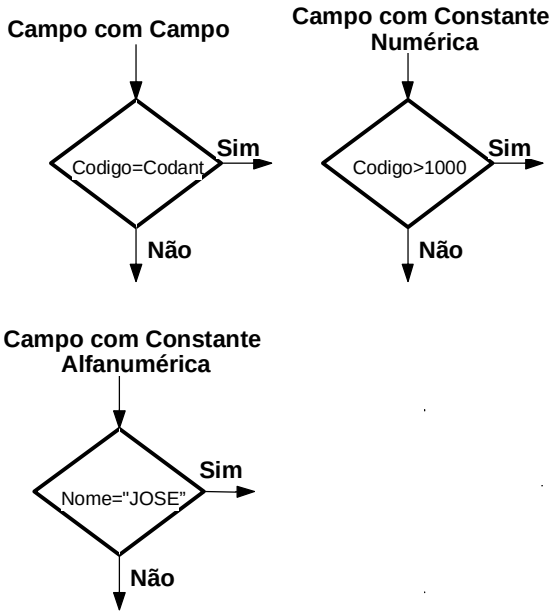
Exemplo:

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

Símbolo Utilizado para comparação entre expressões



5.3 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
OU	OR
NÃO	NOT

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

A tabela abaixo mostra todos os valores possíveis criados pelos três operadores lógicos (AND, OR e NOT)

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**

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

5.4 EXERCÍCIOS

1) Tendo as variáveis SALARIO, IR e SALLIQ, e considerando os valores abaixo. Informe se as expressões são verdadeiras ou falsas.

SALARIO	IR	SALLIQ	EXPRESSÃO	V ou F
100,00	0,00	100	(SALLIQ >= 100,00)	
200,00	10,00	190,00	(SALLIQ < 190,00)	
300,00	15,00	285,00	SALLIQ = SALARIO - IR	

2) Sabendo que A=3, B=7 e C=4, informe se as expressões abaixo são verdadeiras ou falsas.

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

3) Sabendo que A=5, B=4 e C=3 e D=6, informe se as expressões abaixo são verdadeiras ou falsas.

a) (A > C) **AND** (C <= D) ( )
b) (A+B) > 10 **OR** (A+B) = (C+D) ( )
c) (A>=C) **AND** (D >= C) ( )

6 Operações Lógicas

Operações Lógicas são utilizadas quando se torna necessário tomar decisões em um diagrama de bloco.

Num diagrama de bloco, toda decisão terá sempre como resposta o resultado VERDADEIRO ou FALSO.

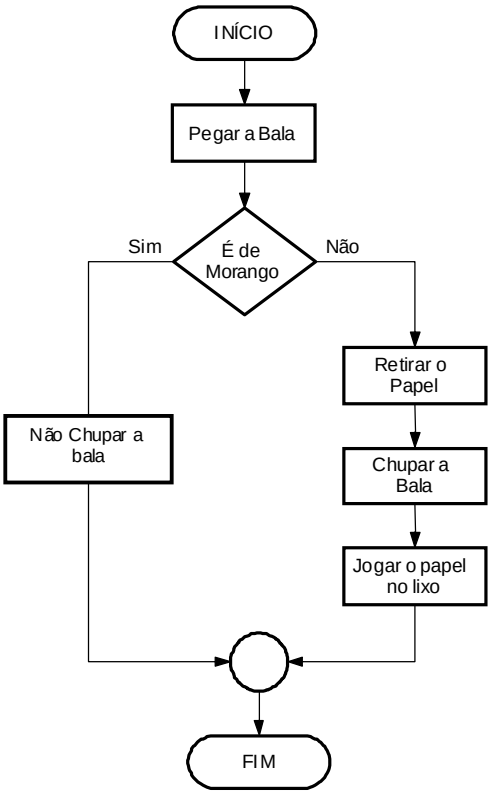
Como no exemplo do algoritmo “CHUPAR UMA BALA”. Imaginemos que algumas pessoas não gostem de chupar bala de Morango, neste caso teremos que modificar o algoritmo para:

“Chupar uma bala”.

- Pegar a bala
- A bala é de morango?
  - Se **sim**, não chupe a bala
  - Se **não**, continue com o algoritmo

- Retirar o papel
- Chupar a bala
- Jogar o papel no lixo

Exemplo: Algoritmo “Chupar Bala” utilizando diagrama de Blocos



6.1 EXERCÍCIOS

1) Elabore um diagrama de blocos que leia um número. Se positivo armazene-o em A, se for negativo, em B. No final mostrar o resultado

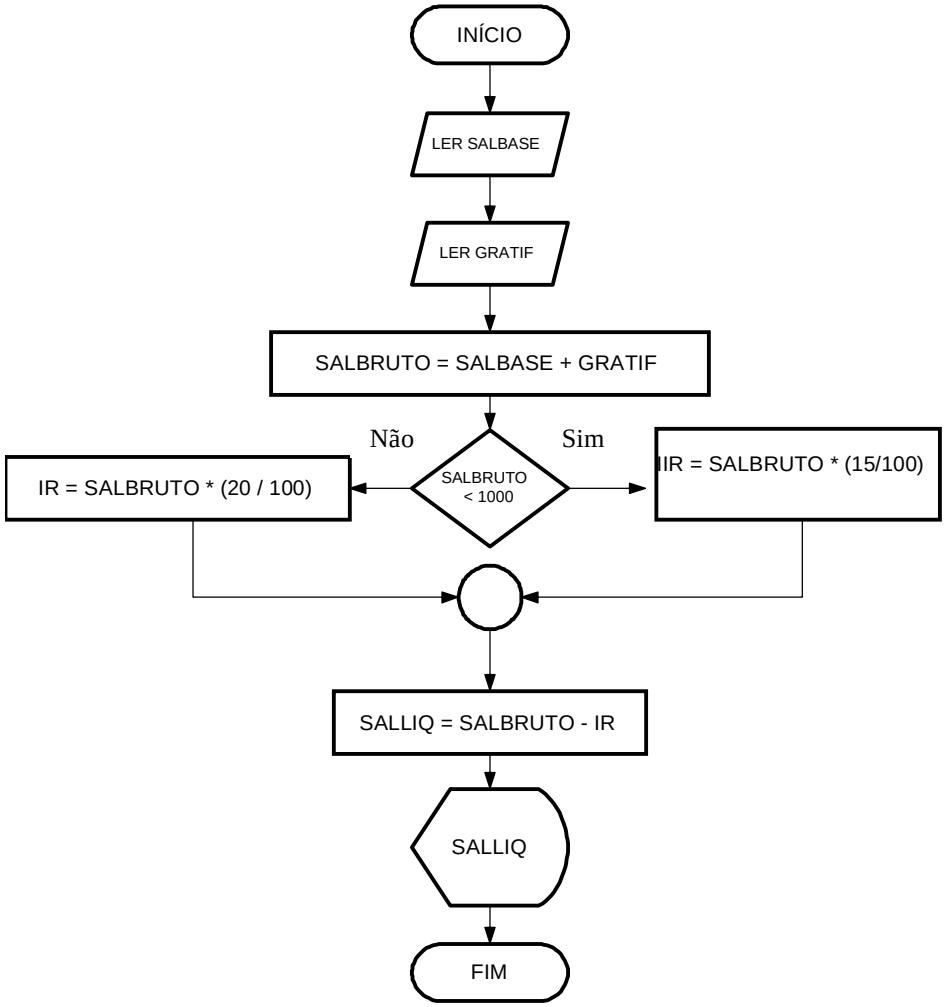
2) Ler um número e verificar se ele é par ou ímpar. Quando for par armazenar esse valor em P e quando for ímpar armazená-lo em I. Exibir P e I no final do processamento.

3) Construa um diagrama de blocos para ler uma variável numérica N e imprimi-la somente se a mesma for maior que 100, caso contrário imprimi-la com o valor zero

4) Tendo como dados de entrada a altura e o sexo de uma pessoa, construa um algoritmo que calcule seu peso ideal, utilizando as seguintes fórmulas:

   Para homens: (72.7*h) - 58
   Para mulheres: (62.1*h) - 44.7 (h = altura)

5) Faça um teste de mesa do diagrama apresentado abaixo, de acordo com os dados fornecidos:



Teste o diagrama com os dados abaixo

SALBASE	GRATIF
3.000,00	1.200,00
1.200,00	400,00
500,00	100,00

Memória

SALBASE	GRATIF	SALBRUTO	IR	SALLIQ

Dados de Saída

SALLIQ

Elabore um algoritmo levando-se em conta o diagrama apresentado:

7 Estrutura de Decisão e Repetição

Como vimos no capítulo anterior em “Operações Lógicas”, verificamos que 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

7.1 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 seqüenciais. 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***”

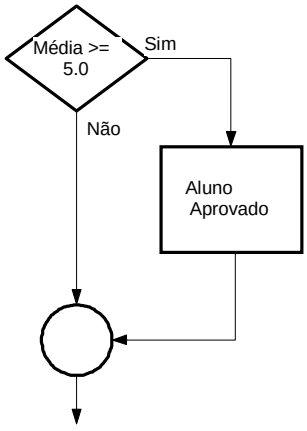
7.1.1 SE ENTÃO / IF ... THEN

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 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

Em diagrama de blocos ficaria assim:



Em Visual Basic

```
IF MEDIA >= 5 Then
   Text1 = “APROVADO”
ENDIF
```

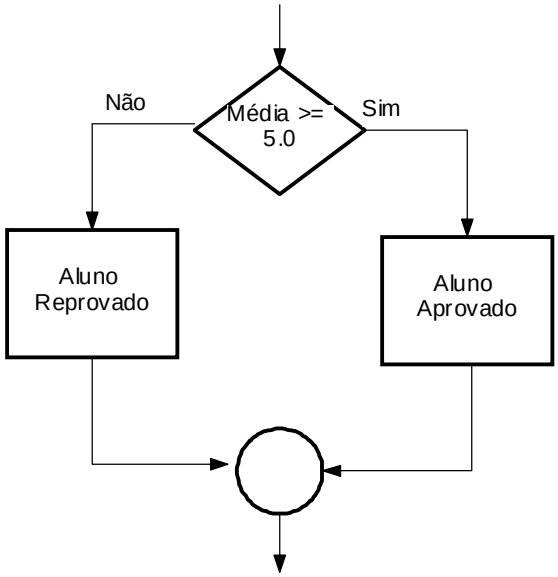
7.1.2 SE ENTÃO SENÃO / IF ... THEN ... ELSE

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

Em algoritmo ficaria assim:

```
SE MÉDIA >= 5.0 ENTÃO
    ALUNO APROVADO
SENÃO
    ALUNO REPROVADO
```

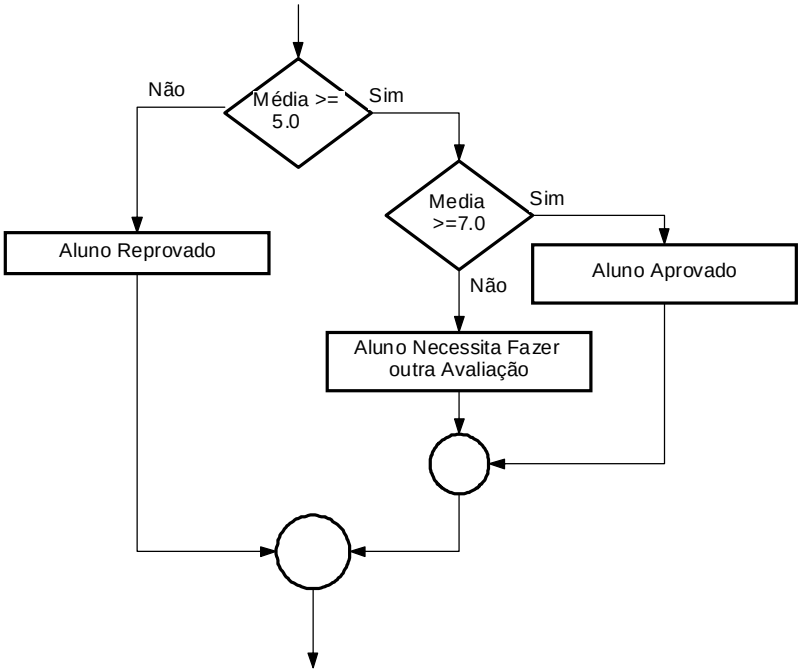
Em diagrama



Em Visual Basic

```
IF MEDIA >= 5 Then
   Text1 = “APROVADO”
ELSE
   Text1 = “REPROVADO”
ENDIF
```

No exemplo acima está sendo 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 no exemplo abaixo:



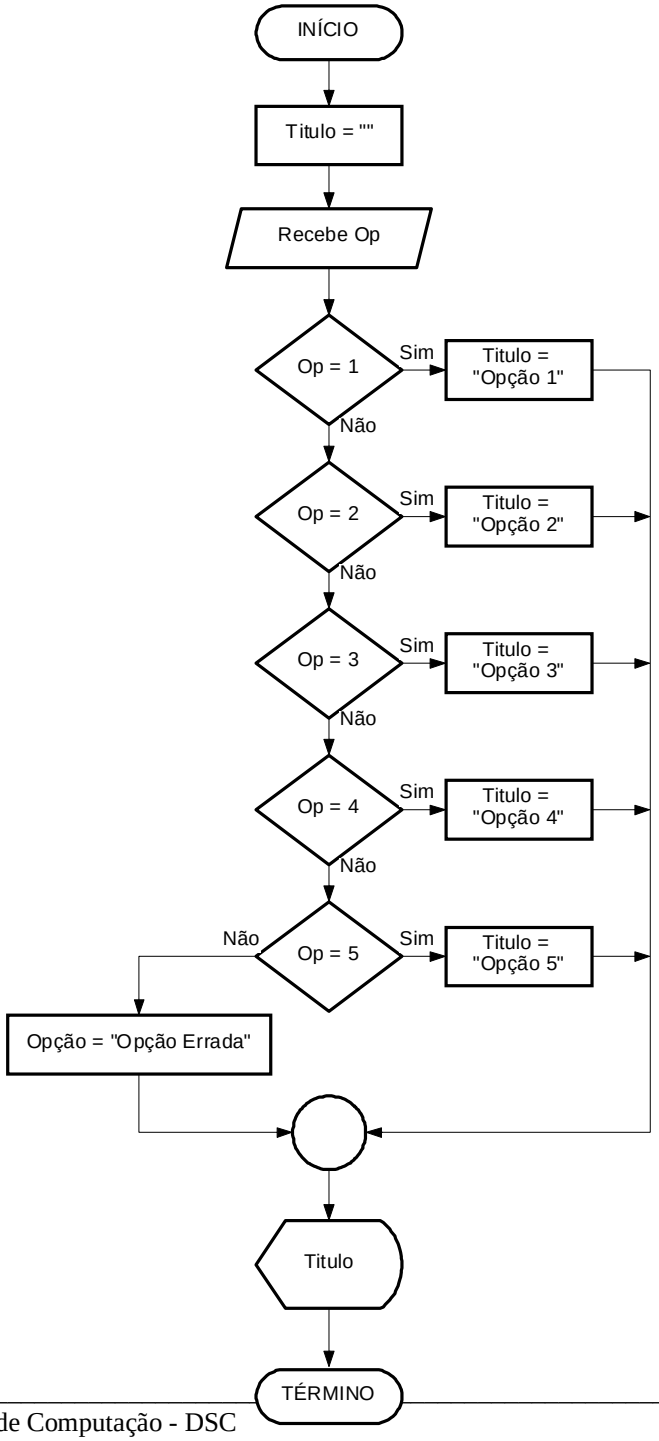
Em Visual Basic

```
IF MEDIA >= 5 Then
   IF MEDIA >= 7.0 then
      Text1 = “Aluno APROVADO”
   ELSE
       Text1 = “Aluno Necessita fazer outra Avaliação”
   ENDIF
ELSE
   Text1 = “Aluno REPROVADO”
ENDIF
```

7.1.3 CASO SELECIONE / SELECT ... CASE

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”.

No exemplo do diagrama de blocos abaixo, é recebido uma variável “**Op**” e testado seu conteúdo, caso uma das condições seja satisfeita, é atribuído para a variável Titulo a String “Opção X”, caso contrário é atribuído a string “Opção Errada”.



Em Visual Basic utilizamos a seguinte seqüência de comandos para representar o diagrama anterior.

```
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
```

7.1.4 EXERCÍCIOS

1) João Papo-de-Pescador, homem de bem, comprou um microcomputador para controlar o rendimento diário de seu trabalho. Toda vez que ele traz um peso de peixes maior que o estabelecido pelo regulamento de pesca do estado de São Paulo (50 quilos) deve pagar um multa de R$ 4,00 por quilo excedente. João precisa que você faça um diagrama de blocos que leia a variável P (peso de peixes) e verifique se há excesso. Se houver, gravar na variável E (Excesso) e na variável M o valor da multa que João deverá pagar. Caso contrário mostrar tais variáveis com o conteúdo ZERO.

2) Elabore um diagrama de bloco que leia as variáveis C e N, respectivamente código e número de horas trabalhadas de um operário. E calcule o salário sabendo-se que ele ganha R$ 10,00 por hora. Quando o número de horas exceder a 50 calcule o excesso de pagamento armazenando-o na variável E, caso contrário zerar tal variável. A hora excedente de trabalho vale R$ 20,00. No final do processamento imprimir o salário total e o salário excedente.

3) Desenvolva um diagrama que:

   - Leia 4 (quatro) números;
   - Calcule o quadrado de cada um;
   - Se o valor resultante do quadrado do terceiro for >= 1000, imprima-o e finalize;
   - Caso contrário, imprima os valores lidos e seus respectivos quadrados.

4) Faça um diagrama de bloco que leia um número inteiro e mostre uma mensagem indicando se este número é par ou ímpar, e se é positivo ou negativo.

5) A Secretaria de Meio Ambiente que controla o índice de poluição mantém 3 grupos de indústrias que são altamente poluentes do meio ambiente. O índice de poluição aceitável varia de 0,05 até 0,25. Se o índice sobe para 0,3 as indústrias do 1º grupo são intimadas a suspenderem suas atividades, se o índice crescer para 0,4 as industrias do 1º e 2º grupo são intimadas a suspenderem suas atividades, se o índice atingir 0,5 todos os grupos devem ser notificados a paralisarem suas atividades. Faça um diagrama de bloco que leia o índice de poluição medido e emita a notificação adequada aos diferentes grupos de empresas.

6) Elabore um algoritmo que dada a idade de um nadador classifique-o em uma das seguintes categorias:

   Infantil A = 5 a 7 anos
   Infantil B = 8 a 11 anos
   Juvenil A = 12 a 13 anos
   Juvenil B = 14 a 17 anos
   Adultos = Maiores de 18 anos

7) Elabore um algoritmo que gera e escreve os números ímpares dos números lidos entre 100 e 200.

8) Construa um algoritmo que leia 500 valores inteiros e positivos e:

   - Encontre o maior valor
   - Encontre o menor valor
   - Calcule a média dos números lidos

7.2 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.

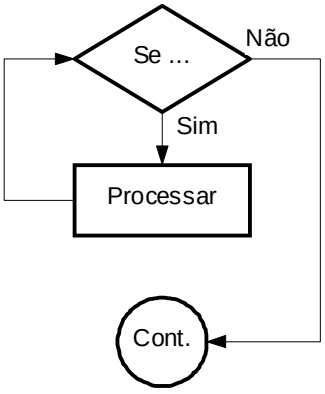
Trabalharemos com modelos de comandos de repetição:

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

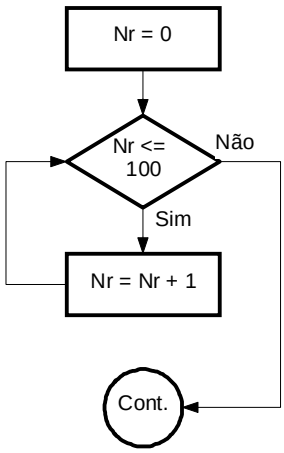
7.2.1 Enquanto x, Processar (Do While ... Loop)

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 a estrutura é a seguinte:



Exemplo de Contador



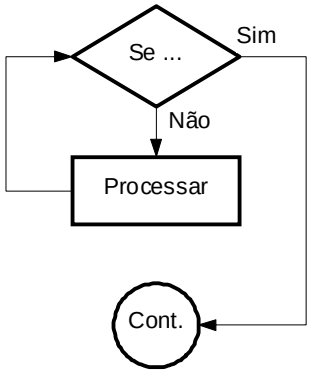
Em Visual Basic:

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

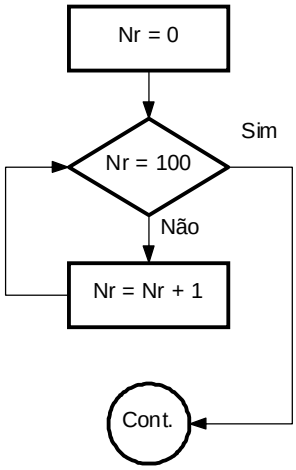
7.2.2 Até que x, processar ... (Do Until ... Loop)

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.

Em diagrama de bloco



Exemplo de Até Diagrama



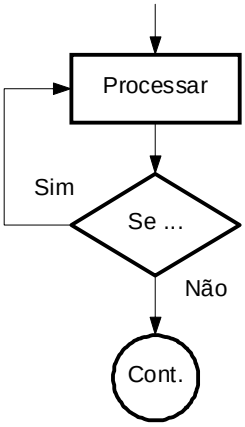
Em Visual Basic

```
Nr = 0
Do Until Nr = 100
    Nr = Nr + 1
Loop

Label1.Caption = Nr
```

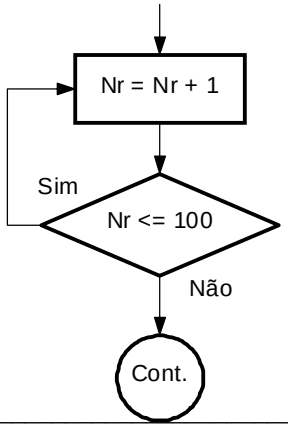
7.2.3 Processar ..., Enquanto x (Do ... Loop While)

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.

Em diagrama de bloco



Exemplo de Até Diagrama



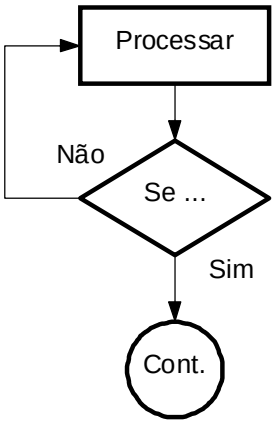
Em Visual Basic

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

Label1.Caption = Nr
```

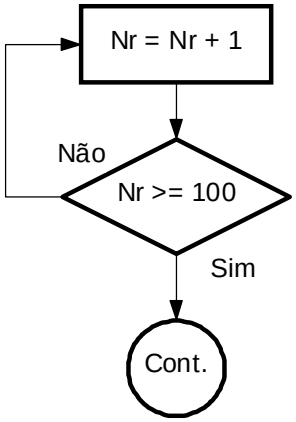
7.2.4 Processar ..., Até que x (Do ... Loop Until)

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.

Em diagrama de Bloco



Exemplo de Do .... Loop - Until



Em Visual Basic

```
nr = 0
Do
   nr = nr + 1
Loop Until nr >= 100
Label1.Caption = nr
```

7.2.5 EXERCÍCIOS

1) Faça um algoritmo que determine o maior entre **N** números. A condição de parada é a entrada de um valor 0, ou seja, o algoritmo deve ficar calculando o maior até que a entrada seja igual a 0 (ZERO).

2) Uma rainha requisitou os serviços de um monge e disse-lhe que pagaria qualquer preço. O monge, necessitando de alimentos , indagou à rainha sobre o pagamento, se poderia ser feito com grãos de trigo dispostos em um tabuleiro de xadrez, de tal forma que o primeiro quadro deveria conter apenas um grão e os quadros subseqüentes , o dobro do quadro anterior. A rainha achou o trabalho barato e pediu que o serviço fosse executado, sem se dar conta de que seria impossível efetuar o pagamento. Faça um algoritmo para calcular o número de grãos que o monge esperava receber.

3) Faça um algoritmo que conte de 1 a 100 e a cada múltiplo de 10 emita uma mensagem: “Múltiplo de 10”.

8 Arquivos de Dados

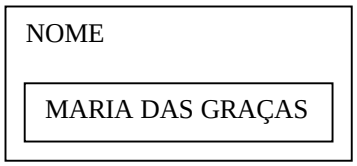
Os dados manipulados até o momento, estavam em memória, ou seja, após a execução do diagrama os dados se perdiam. Para resolver esse problema começaremos a trabalhar com arquivos, onde poderemos guardar os dados e também manipula-los. Para isso necessitamos rever alguns conceitos como: campos, registros e arquivos.

8.1 Conceitos Básicos

CAMPO é um espaço reservado em memória para receber informações (dados).

Exemplo: Campo Nome, Campo Endereço

Campo na memória



REGISTRO é um conjunto de campos

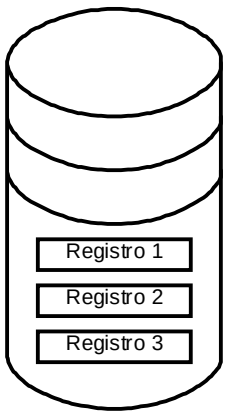
Exemplo: Registro de Clientes

COD-CLI	NOME	ENDEREÇO	FONE
00001	MARIA DAS GRAÇAS	RUA DAS DORES,1400	888-9876

ARQUIVO é um conjunto de registros

Exemplo: O arquivo de Clientes da Empresa, onde estão armazenados os dados de todos os clientes da empresa.

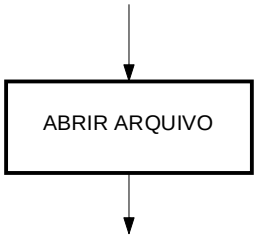
ARQ-CLI



8.2 Abertura de Arquivos

Toda vez que for necessário trabalhar com arquivo, primeiramente precisamos **ABRIR** o arquivo. Abrir o arquivo significa alocar o periférico (disco, disquete) em que o arquivo se encontra, e deixá-lo disponível para leitura/gravação.

O símbolo para abertura de arquivo

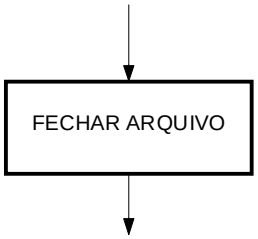


8.3 Fechamento de Arquivos

Da mesma maneira que precisamos abrir um arquivo antes do processamento, também se faz necessário o fechamento do mesmo, para que suas informações não possam ser violadas ou danificadas.

Fechar um arquivo significa liberar o periférico que estava sendo utilizado.

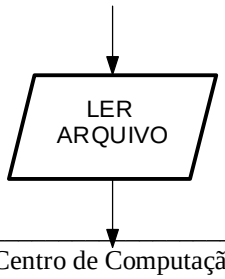
O símbolo para fechamento de arquivo



8.4 Leitura de Arquivos

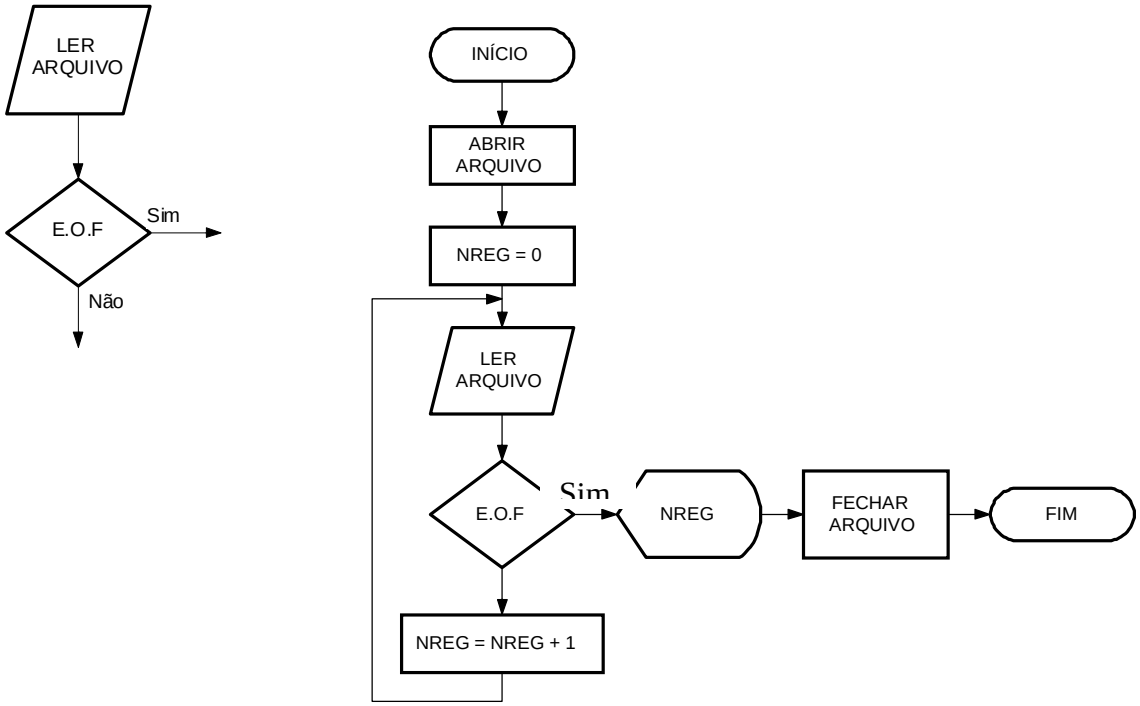
Após abrir um arquivo é necessário **LER** os dados que estão em disco e transferi-los para memória. Essa transferência é feita por registro. Esse procedimento é gerenciado pelo próprio sistema operacional.

O símbolo para leitura de arquivo



Toda vez que abrimos um arquivo ele posiciona o "**ponteiro**" no primeiro registro, ou seja, no início do arquivo. Para que possamos trabalhar com os dados se torna necessário sabermos onde está o ponteiro do registro. Isso poderemos fazer testando se o ponteiro está no início (**BOF – Bottom Of File**) ou no final do arquivo (**EOF – End Of File**). Esse é sempre executado após a leitura do registro (mudança da posição do ponteiro). Simbolicamente podemos representar esse passo da seguinte maneira.

Exemplo de diagrama de bloco



8.5 Movimentação de registros

Como dito no item anterior, quando um arquivo é aberto o ponteiro está no primeiro registro. A cada leitura do Arquivo o ponteiro se movimenta para o próximo registro e assim por diante. Como mostra a figura abaixo:

8.6 Gravação de Arquivos

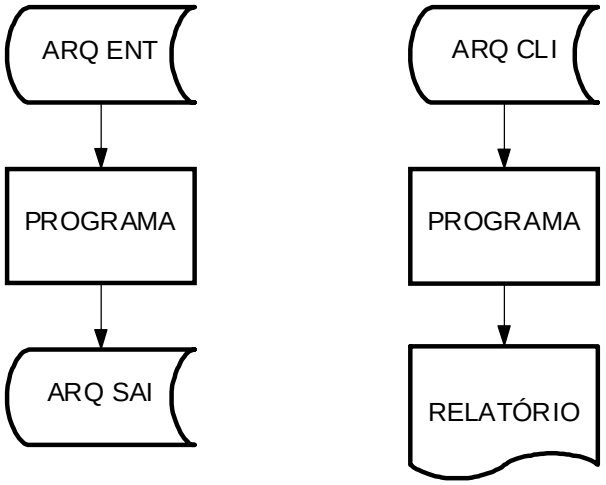
Da mesma maneira que os registros são lidos de um arquivo, também devemos gravar registros em um arquivo.

A gravação consiste na transferência de um registro da memória, para um periférico (disco, disquete).

O símbolo para gravação de arquivos

8.7 Macro Fluxo

O macro fluxo é a representação gráfica dos arquivos que serão processados em um programa.



Estes dois exemplos de Macro-fluxo dão uma visão geral de como devemos proceder com cada um dos programas. O primeiro diz que haverá um arquivo de entrada, um processamento e um arquivo de saída. Já o segundo exemplo diz que haverá um arquivo de entrada, um processamento, e a saída serão um relatório.

8.8 EXERCÍCIOS

1) Foi feita uma pesquisa entre os habitantes de uma região. Foram coletados os dados de idade, sexo (M/F) e salário. Faça um algoritmo que informa:

   a) A média de salário do grupo
   b) Maior e menor idade do grupo
   c) Quantidade de mulheres com salário até R$ 100,00
   d) Quantidade de homens

2) Um arquivo de produtos tem os seguintes campos: Código do produto, Descrição, Quantidade em Estoque, Preço de custo, Margem Custo/Venda. Crie um arquivo com os seguintes campos: Código do Produto e Preço de Venda. Utilize o calculo Preço de Venda = Preço de Custo * Margem CustoVenda.

3) Elabore um diagrama de blocos para verificar que produtos precisam ser comprados e a quantidade a ser adquirida:

   Tendo as seguintes informações

   Código do produto (CODPROD), Quantidade Mínima (QTDMIN), Quantidade Máxima (QTDMAX) e a quantidade em estoque (QTDEST) de cada produto.

   Um produto somente deverá ser comprado quando: a quantidade em estoque for menor ou igual a quantidade mínima:

   QTCOMPRAR = (QTDMAX – QTDEST)

Grave em outro arquivo: Código do Produto e Quantidade a Comprar

9 Relatórios

A impressão de relatórios é o registro de informações processadas pelo computador em um meio de armazenamento de dados chamado de formulário. Para efetuarmos a impressão de relatórios devemos nos preocupar com os seguintes aspectos:

- Características do formulário
- Controle de linhas e salto de página
- Impressão de cabeçalho e estética da página
- Impressão de rodapé
- Numeração de páginas

9.1 Características do Formulário

A maioria dos formulários possui um formato padrão, isto é, a quantidade de linhas por página e de caracteres por linha são constantes.

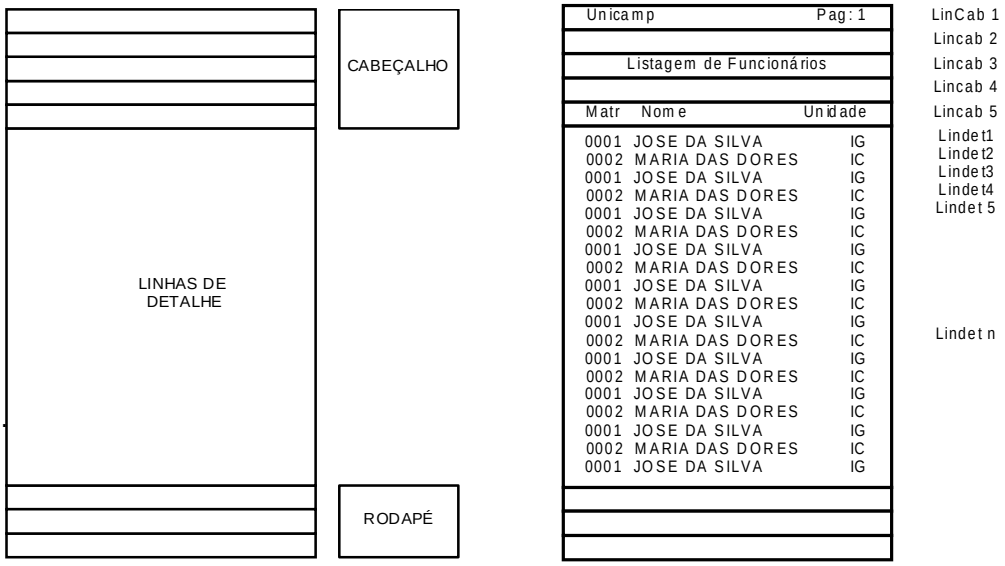
9.2 Controle de linhas e salto de páginas

Uma preocupação com impressão de relatórios é não permitir que a impressora imprima fora do papel, pois além de esteticamente não ficar bom, haveria perda de informações.

Para controlarmos o número de linhas impressas, devemos criar um contador de linha e não deixar o valor desses contadores ultrapassarem o número desejado de linhas por páginas.

9.3 Impressão de Cabeçalho e Estética de Página

Para termos uma idéia melhor da estética do formulário, veja o exemplo abaixo.

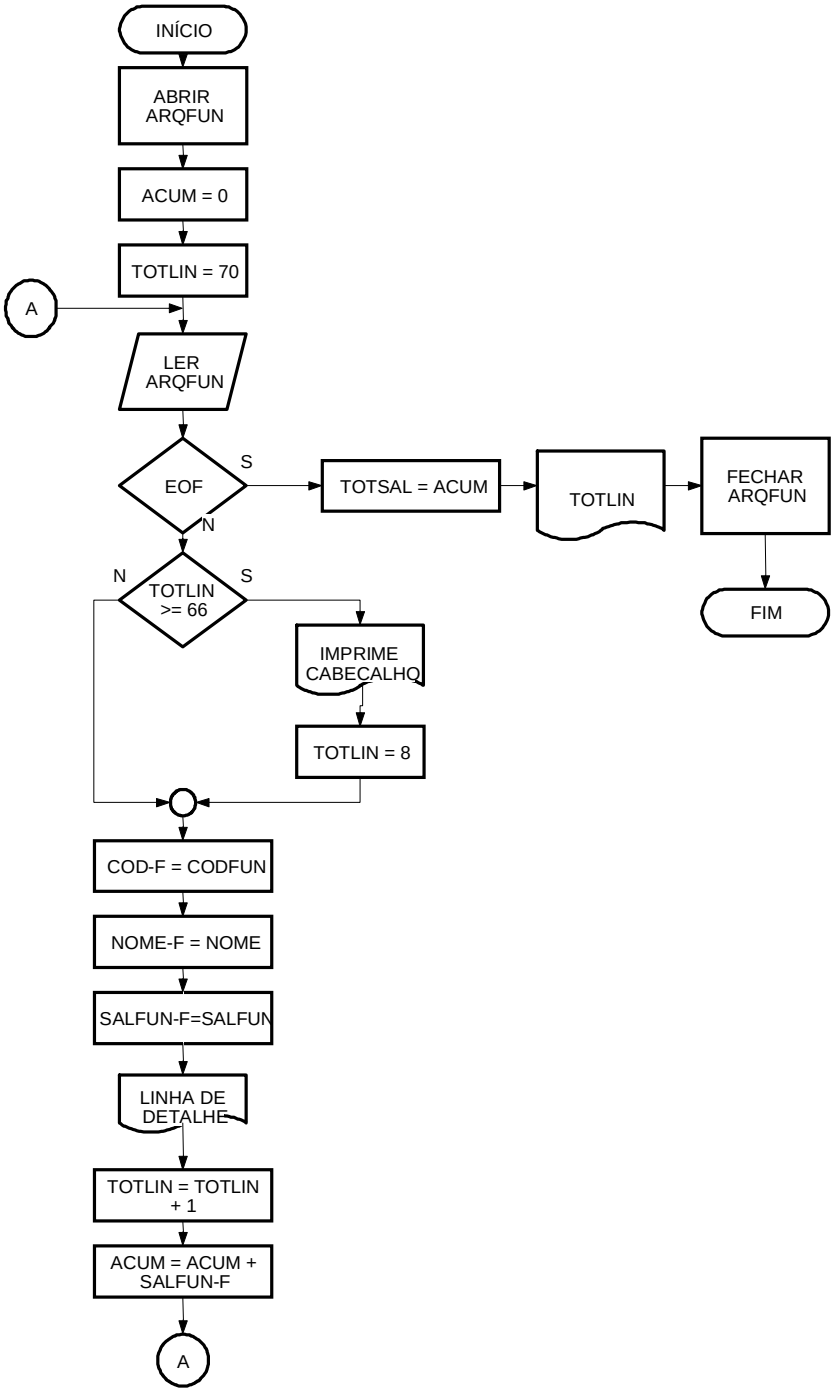


Área de Cabeçalho — Local onde devemos colocar um cabeçalho para identificarmos o assunto a que se refere o conteúdo da página como um todo, e um cabeçalho indicando o significado do conteúdo de cada coluna de informações. Pode haver outras linhas de cabeçalho de acordo com a necessidade.

Linha de Detalhe — São as linhas geradas a partir de dados lidos de um arquivo.

Área de Rodapé — Pode haver linhas contendo valores de totalizações de determinadas colunas e/ou linhas de identificação da empresa, ou outras informações qualquer.

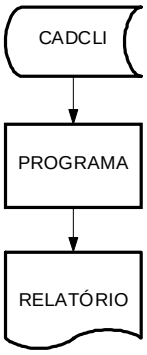
Veja abaixo um exemplo de diagrama de bloco para impressão de relatório



9.4 EXERCÍCIOS

1) Um banco deseja emitir uma listagem de todos os clientes cujos saldos sejam iguais ou superiores a R$ 1.500,00. Faça o diagrama de bloco correspondente, considerando:

Macro-Fluxo



LAY-OUTS (Cadastro de Clientes)

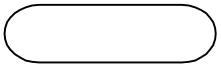
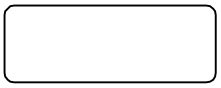
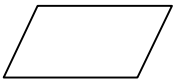
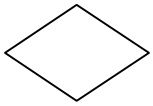
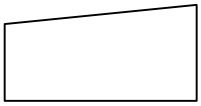
NUMERO-CONTA	NOME-CLIENTE	SALDO-ATUAL

Lay-Outs (Relatório)

```
+          RELAÇÃO DE CLIENTES / SALDOS              PAG: XXX

NÚMERO-CONTA      NOME                          SALDO

XXXXXXXXXXXX      XXXXXXXXXXXXXXXXXXXXX  XXX.XXX,XX
XXXXXXXXXXXX      XXXXXXXXXXXXXXXXXXXXX  XXX.XXX,XX

                  TOTAL EM CONTA CORRENTE  XXX.XXX.XXX,XX
```

Observações: Cabeçalhos em todas as páginas, 66 linhas por página e totalização no final de cada página.

10 Simbologia

Símbolo	Função
 TERMINAL	Indica o INÍCIO ou FIM de um processamento Exemplo: Início do algoritmo
 PROCESSAMENTO	Processamento em geral Exemplo: Calculo de dois números
 ENTRA/SAÍDA	Operação de entrada e saída de dados Exemplo: Leitura e Gravação de Arquivos
 DECISÃO	Indica uma decisão a ser tomada Exemplo: Verificação de Sexo
 DESVIO	Permite o desvio para um ponto qualquer do programa
 ENTRADA MANUAL	Indica entrada de dados através do Teclado Exemplo: Digite a nota da prova 1
 EXIBIR	Mostra informações ou resultados Exemplo: Mostre o resultado do calculo
 RELATÓRIO	Relatórios

11 Referências

Lógica de Programação – A Construção de Algoritmos e Estruturas de Dados – São Paulo: Forbellone, André Luiz Villar - MAKRON, 1993.

Sites na Web: http://www.inf.pucrs.br/~egidio/algo1/