CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE SERGIPE

Unidade de Ensino Descentralizada de Lagarto

Apostila de Algoritmo

Professora: Clênia Melo

I - Lógica 1
1. Lógica de Programação 2
II - Algoritmo 2
1. Definições 2
2. Exemplos 3
3. Estrutura de um algoritmo 3
4. Identificadores 4
5. Tipos Primitivos 4
6. Constantes 4
7. Variáveis 5
7.1 - Declaração de variáveis 5
8. Operações Básicas 5
8.1 - Prioridade de Operadores 6
9. Comandos de Entrada e Saída 7
10. Estruturas de Controle 8
11. VETOR 10
12. MATRIZ 11
13. REGISTRO 11
14. MODULARIZAÇÃO 12
14.1 - Procedimentos 12
14.2 - Funções 13

I - Lógica

O uso corriqueiro está normalmente associado à coerência e racionalidade. É freqüente que se associe lógica apenas à matemática, no entanto, se esquece que também está associada às outras ciências.

Pode-se relacionar a lógica com a “correção de pensamento”, pois uma de suas preocupações é determinar quais operações são válidas e quais não são, fazendo análises das formas e leis do pensamento. A Lógica estuda e ensina a colocar “ordem no pensamento”.

Exemplos:

a. Todo mamífero é um animal.
   Todo cavalo é um mamífero.
   Portanto, todo cavalo é um animal.

b. Num computador, tudo o que é físico é hardware.
   O mouse é um dos componentes físicos de um computador.
   Logo o mouse é hardware.

Estes exemplos ilustram a Lógica Proposicional, pois representam um argumento composto por duas premissas e uma conclusão, que estabelece uma relação que por sua vez pode ser verdadeira ou não.

Outros exemplos:

a. A gaveta está fechada.
   A caneta está dentro da gaveta.
   Precisa-se primeiro abrir a gaveta para depois pegar a caneta.

b. Anacleto é mais velho do que Felisberto.
   Felisberto é mais velho do que Marivaldo.
   Portanto Anacleto é mais velho do que Marivaldo.

1. Lógica de Programação

É o uso correto das leis do pensamento, da “ordem da razão” e de processos de raciocínio e simbolização formais na programação de computadores, ou seja, é uma tentativa de fazer computadores usarem raciocínio lógico.

II - Algoritmo

1. Definições

“O conceito central da programação e da Ciência da Computação é o conceito de algoritmos, isto é, programar é basicamente construir algoritmos”.

É a descrição, de forma lógica, dos passos a serem executados no cumprimento de determinada tarefa.

“O algoritmo pode ser usado como uma ferramenta genérica para representar a solução de tarefas independente do desejo de automatizá-las, mas em geral está associado ao processamento eletrônico de dados, onde representa o rascunho para programas (Software)”.

“Serve como modelo para programas, pois sua linguagem é intermediária à linguagem humana e às linguagens de programação, sendo então, uma boa ferramenta na validação da lógica de tarefas a serem automatizadas”.

“Um algoritmo é uma receita para um processo computacional e consiste de uma série de operações primitivas, interconectadas devidamente, sobre um conjunto de objetos. Os objetos manipulados por essas receitas são as variáveis”.

O algoritmo pode ter vários níveis de abstrações de acordo com a necessidade de representar ou encapsular detalhes inerentes às linguagens de programação. Ex: Certamente um algoritmo feito com o objetivo de servir como modelo para uma linguagem de III geração é diferente daquele para uma linguagem de IV geração. Mas isso não impede que a ferramenta em si possa ser usada em ambos o caso.

Como qualquer modelo, um algoritmo é uma abstração da realidade. A abstração é o processo de identificar as propriedades relevantes do fenômeno que esta sendo modelado. Usando o modelo abstrato, podemos nos centrar unicamente nas propriedades relevantes para nós, dependendo da finalidade da abstração, e ignorar as irrelevantes.

É a forma pela qual se descreve soluções de problemas do **nosso mundo**, afim de, serem implementadas utilizando os recursos do **mundo computacional**. Como este possuí severas limitações em relação ao nosso mundo, exige que, sejam impostas algumas regras básicas na forma de solucionar os problemas, para que, possamos utilizar os recursos de **hardware** e **software** disponíveis. Pois, os algoritmos, apesar de servirem para representar a solução de qualquer problema, no caso do Processamento de Dados, eles devem seguir as regras básicas de programação para que sejam compatíveis com as **linguagens de programação**.

2. Exemplos

a. Trocar a lâmpada
b. Receita de bolo
c. Ouvir uma música no rádio
d. Ir à escola

3. Estrutura de um algoritmo

```
Algoritmo Nome_Do_Algoritmo
```

```
variáveis
Declaração das variáveis
Procedimentos
Declaração dos procedimentos
Funções
Declaração das funções
Início
Corpo do Algoritmo
Fim
```

4. Identificadores

Representam os nomes escolhidos para rotular as variáveis, procedimentos e funções, normalmente, obedecem as seguintes regras:

- O primeiro caracter deve ser uma letra
- Os nomes devem ser formados por caracteres pertencentes ao seguinte conjunto: {a,b,c,..z,A,B,C,...Z,0,1,2,...,9,_}
- Os nomes escolhidos devem explicitar seu conteúdo.

5. Tipos Primitivos

Os tipos de dados e variáveis utilizados dependem da finalidade dos algoritmos, mas, podemos definir alguns, pelo fato de serem largamente utilizados e implementados na maioria das linguagens, sendo estes:

- *INTEIRO*: qualquer número inteiro, negativo, nulo ou positivo.
- *REAL*: qualquer número real, negativo, nulo ou positivo.
- *CARACTER*: qualquer conjunto de caracteres alfanuméricos.
- *LÓGICO*: tipo especial de variável que armazena apenas os valores V e F, onde V representa VERDADE e F FALSO.

6. Constantes

Entende-se que um dado é constante quando não sofre nenhuma variação no decorrer do tempo, ou seja, seu valor é constante desde o início até o fim da execução do algoritmo.

7. Variáveis

Um dado é classificado como variável quando tem a possibilidade de ser alterado em algum instante no decorrer do tempo, ou seja, durante a execução do algoritmo em que é utilizado, o valor sofre alteração ou o dado é dependente da execução em um certo momento ou circunstância.

Variáveis são unidades básicas de armazenamento das informações a nível de linguagens de programação.

7.1 - Declaração de variáveis

Para que os programas manipulem valores, estes devem ser armazenados em variáveis e para isso, devemos declará-las de acordo com a sintaxe:

```
NomeVariável,... : tipo
```

8. Operações Básicas

Na solução da grande maioria dos problemas é necessário que as variáveis tenham seus valores consultados ou alterados e, para isto, devemos definir um conjunto de **OPERADORES**, sendo eles:

- **OPERADOR DE ATRIBUIÇÃO:**

NomeDaVariavel □ Valor ou Expressão Atribuída.

- **OPERADORES ARITMÉTICOS:**

+	Adição
-	Subtração ou inversor do sinal.
*****	Multiplicação
/	Divisão
Quociente	Quociente da divisão de inteiros
Resto	Resto da divisão de inteiros
EXP(a,b)	Exponenciação a^b

- **FUNÇÕES PRIMITIVAS:** SEN(x); COS(x); TG(x); ABS(x); INT(x); Raiz(x); PI( );

- **OPERADORES RELACIONAIS:**

São utilizados para relacionar variáveis ou expressões, resultando num valor lógico (Verdadeiro ou Falso), sendo eles:

=	igual
<	menor
<=	menor ou igual
<>	diferente
>	maior
>=	maior ou igual

- **OPERADORES LÓGICOS:**

São utilizados para avaliar expressões lógicas, sendo eles:

e - e lógico ou conjunção.
ou - ou lógico ou disjunção.
não - negação.

8.1 - Prioridade de Operadores

Durante a execução de uma expressão que envolve vários operadores, é necessário a existência de prioridades, caso contrário poderemos obter valores que não representam o resultado esperado.

A maioria das linguagens de programação utiliza as seguintes prioridades de operadores:

1º - Efetuar operações embutidas em parênteses "mais internos"
2º - Efetuar Funções
3º - Efetuar multiplicação e/ou divisão
4º - Efetuar adição e/ou subtração
5º - Operadores Relacionais
6º - Operadores Lógicos

OBS: O programador tem plena liberdade para incluir novas variáveis, operadores ou funções para adaptar o algoritmo as suas necessidades, lembrando sempre, de que, estes devem ser compatíveis com a linguagem de programação a ser utilizada.

Exercícios

1) Dadas as variáveis inteiras X, Y e Z contendo os valores 2, 5 e 9 respectivamente. A variável NOME do tipo caracter contém “JOSÉ” e a variável lógica SIM contem o valor FALSO, indique os resultados das expressões lógicas abaixo:
    a. (X + Y > Z) e NOME = “JOSÉ”

b. SIM ou $(Y \geq Z)$

c. Não SIM e $(Z \div Y + 1 = X)$

d. (NOME = “MARIA”) e SIM ou $(X^{2} < Z + 10)$

2) Com as declarações

```
var
    A, B: real;
    NOME, PROFISSAO: caracter
```

Completar o quadro a seguir, com o valor das relações indicadas, tendo em vista os valores atribuídos às variáveis.

VARIÁVEIS				**RELAÇÕES**		
A	**B**	**NOME**	**PROFISSAO**	$A + 1 \geq \sqrt{B}$	**NOME ≠ “PEDRO”**	**PROFISSÃO = “MEDICO”**
3,0	16,0	“MIRIAM”	“ADVOGADO”			
5,0	64,0	“PEDRO”	“MEDICO”			
2,5	9,0	“CARLA”	“PROFESSOR”			

9. Comandos de Entrada e Saída

No algoritmo é preciso representar a troca de informações que ocorrerá entre o mundo da máquina e o nosso mundo, para isso, devemos utilizar comandos de entrada e saída, sendo que, a nível de algoritmo esses comandos representam apenas a entrada e a saída da informação, independe do dispositivo utilizado (teclado, discos, impressora, monitor,...), mas, sabemos que nas linguagens de programação essa independência não existe, ou seja, nas linguagens de programação temos comandos específicos para cada tipo de unidade de Entrada/Saída.

Comando de Entrada de Dados

```
Leia(variável_1, variável_2,...)
```

Comando de Saída de Dados

```
Imprima(expressão_1, expressão_2,...)
```

Exercícios

3) Faça um algoritmo para ler os catetos de um triângulo, calcular e imprimir a sua hipotenusa:

4) Faça o algoritmo para ler o preço de compra de um produto e o percentual de lucro desejado por um vendedor, com esses valores, calcule e imprima o preço de venda.

Obs: O percentual de lucro será informado da seguinte maneira: 10 para 10%, 20 para 20%, 30 para 30% e assim por diante.

5) Você foi convidado por um gerente de um banco para desenvolver uma aplicação de apoio a investimento. Dessa forma seu projeto deverá obter um valor de depósito, uma taxa de juros e a quantidade de meses para a aplicação, em seguida, calculem o lucro que seria obtido com essa aplicação.
Obs: Levar em consideração que a forma é de juros compostos

6) Faça o algoritmo para lê a matrícula, o nome, as três notas de um aluno em seguida, calcule e imprima a sua média.

10. Estruturas de Controle

Para representar a solução de um problema devemos escrever o conjunto de passos a serem seguidos, sendo que, a maioria dos problemas exigem uma dinâmica na sua solução, impondo assim que os algoritmos executem conjunto de instruções de acordo com as possíveis situações encontradas no problema original. E de acordo com a **Programação Estruturada** os mecanismos utilizados para esse controle são: **Sequência**, **Seleção** e **Repetição.**

- **SEQUÊNCIA:** usada para executar comandos passo a passo, sabendo que todos eles serão executados na ordem de escrita, sem nenhum desvio. Uma sequência pode possuir um ou vários comandos, os quais devem ser delimitados pelos identificadores **Início** e **Fim**.

Inicio
Comando_1
...
Comando_n
Fim

- **SELEÇÃO:** usada para tomar decisões, ou seja desviar a execução do algoritmo de acordo com uma condição, podendo ser simples ou composta.

Simples	Composta
Se (Expressão Lógica) **Então** Sequência_1	**Se** (Expressão Lógica) **Então** Sequência_1 **Senão** Sequência_2

- **REPETIÇÃO:** Serve para efetuar um conjunto de ações repetidas vezes. Existem três tipos básicos de repetições, sendo elas.

Enquanto (Expressão Lógica) **faça** Seqüência	O comando **Enquanto** analisa a *Expressão Lógica* e enquanto o seu resultado for, o valor lógico, *Verdade* a *Seqüência* é executada.
Para variável valor_inicial **até** valor_final **faça** Seqüência	O comando **Para** incrementa, a *variável*, a partir do *valor_inicial* de uma unidade, até que, esta atinja o *valor_final*. E para cada incremento a *Seqüência* é executada..
Repita Seqüência **Até** (Expressão Lógica)	O comando **Repita** executa a *Seqüência* até que o valor retornado pela *Expressão Lógica* seja *Verdadeiro*

Exercícios

7) Faça um algoritmo para ler um número inteiro e imprimir se este número é “PAR” ou “IMPAR”:

8) Faça um algoritmo para ler três valores reais e caso eles possam formar os lados de um triângulo, informar se o triângulo é eqüilátero, isoceles ou escaleno:

9) Faça um algoritmo para ler dois números e imprimir o maior:

10) Leia um valor real representando um salário e faça este salário lido receber um aumento de acordo com a tabela:

Salário	Percentagem de aumento (%)
Até R$ 300,00	0,5
R$ 300,01 e R$ 500,00	0,4
R$ 500,01 e R$ 700,00	0,3
R$ 700,01 e R$ 800,00	0,2
R$ 800,01 e R$ 1000,00	0,1
Acima de R$ 1000,00	0,05

11) Faça um algoritmo para ler a matrícula, o nome, as três notas e o número de faltas de um aluno, em seguida, imprimir a matrícula, o nome, a média, o número de faltas e a situação final deste aluno.
Obs: a. As situações finais são as seguintes:
AP – aprovado
RM – reprovado por média
RF – reprovado por falta
b. Para ser aprovado o aluno precisa ter média $\geq$ 5,0 e o número de faltas $\leq$ 18
c. A reprovação por falta sobrepõe a reprovação por média.

12) Lê três números e imprimi-los em ordem crescente

13) Faça um algoritmo para imprimir os número inteiros de 1 até 1000:

14) Faça um algoritmo para ler 70 números inteiros e imprimir a média desses números lidos:

15) Faça um algoritmo para lê um conjunto dos números inteiros e imprimir a média dos números lidos
Obs: O número zero (0) irá indicar o final da leitura dos números

16) Faça um algoritmo para lê um valor inteiro N, em seguida, leia N números inteiros e imprima a média desses números lidos:

17) Faça um algoritmo para lê a matrícula, o nome, as três notas e o número de faltas e o sexo dos alunos da turma de Algoritmo. O flag é uma matrícula igual a FIM. Para cada aluno imprima sua matrícula, nome e situação final e para a turma o número de homens aprovados e o número de mulheres reprovadas:

11. VETOR

Estrutura formada por um conjunto unidimensional de dados de mesmo tipo (homogêneo) e possuindo número fixo de elementos (Estático). Na declaração dos vetores devemos informar o seu nome, seu tipo (inteiro, real, caracter, ...), e seu tamanho (número de elementos). Cada elemento do vetor é identificado por um índice (unidimensional), o qual indica a sua posição no vetor.

Declaração:

```
NomeDoVetor : vetor[n° de elementos] de Tipo do Vetor
```

Referência:

```
NomeDoVetor[índice]
```

Exercícios

18) Faça um algoritmo para ler o tamanho e os elementos de dois vetores (o tamanho máximo dos vetores é 100) em seguida gere um terceiro vetor formado pela soma dos dois vetores lidos. Para que dois vetores sejam gerados tem que possuir o mesmo tamanho. Cada elemento do vetor soma é formado para soma dos elementos do mesmo índice dos dois vetores:

19) Faça um algoritmo para lê um conjunto com no máximo 100 nomes de alunos, onde o flag é um nome igual a “FIM”, armazenando-os em um vetor, em seguida inverta este vetor, de forma que o último passe a ser o primeiro, o primeiro passe a ser o último e assim por diante.

20) Lê dois vetores e caso tenham o mesmo tamanho, armazene seus elementos alternadamente em um terceiro vetor:

12. MATRIZ

Estrutura semelhante ao vetor, sendo que, pode possuir **n** dimensões. Desta forma para fazer referência aos elementos de uma matriz, precisaremos de tantos índices quanto for suas dimensões.

Declaração:

```
NomeDaMatriz : matriz[dimensões] de Tipo da Matriz
```

Referência:

```
NomeDaMatriz[índices]
```

13. REGISTRO

Estrutura formada por um conjunto de variáveis, que podem possuir tipos diferentes (Heterogêneo), agrupadas em uma só unidade.

Declaração:

```
Tipo <NomeRegistro> = registro
     campo1: tipo_do_campo;
   campo1: tipo_do_campo;
     campo1: tipo_do_campo;
fimRegistro;
```

Referência:

```
Variáveis
     VariavelRegistro: tipoRegistro
```

Acesso

```
Leia (<NomeRegistro>.<campo1>);
Imprima (<NomeRegistro>.<campo1>);
```

Obs: Podemos ainda definir um vetor formado por registros.

21) Declare um registro que armazene os dados de um Aluno:

22) Declare um conjunto de registros que comporte as informações de estoque de 500 produtos:
23) Declare um registro para os dados de um cheque bancário:

14. MODULARIZAÇÃO

A modularização consiste num método para facilitar a construção de grandes programas, através de sua divisão em pequenas etapas, que são: módulos, rotinas, sub-rotinas ou sub-programas. Permitindo o reaproveitamento de código, já que podemos utilizar um módulo quantas vezes for necessário, eliminando assim a necessidade de escrever o mesmo código em situações repetitivas.

14.1 - Procedimentos

Um procedimento é um bloco de código precedido de um cabeçalho que contém o Nome do procedimento e seus parâmetros. Com isto, podemos fazer referência ao bloco de código de qualquer ponto do algoritmo através do seu *nome* e passando os seus *parâmetros*.

Declaração:

```
Procedimento NomeDoProcedimento [(parâmetros)]
Variáveis
Inicio
     Comandos;
Fim;
```

Onde, *parâmetros* representam as variáveis que devem ser passadas ao procedimento. Os parâmetros podem ser de: ENTRADA (passado por valor) ou de ENTRADA/SAÍDA (passado por referência). Os parâmetros de ENTRADA não podem ser alterados pelo procedimento, para que isso seja possível o parâmetro deve ser de ENTRADA/SAÍDA Para indicar que um parâmetro é de ENTRADA/SAÍDA devemos colocar a palavra *VAR* antes da sua declaração.

Referência:

```
NomeDoProcedimento(variáveis)
```

OBS: As variáveis passadas aos procedimentos são associadas aos parâmetros do procedimento de acordo com a ordem das variáveis e da lista de parâmetros.

14.2 - Funções

Uma função é semelhante a um procedimento, sendo que esta deve retornar, obrigatoriamente, um valor em seu nome, desta forma, é necessário declarar, no cabeçalho da função, qual o seu tipo.

Declaração:

```
Função NomeDaFunção [(parâmetros)] : tipo_da_função
Variáveis
Inicio
Comandos
NomeDaFunção ← (expressão de retorno)
Fim;
```

Referência:

```
NomeDaFunção(parâmetro)
```