



Universidade Federal da Bahia

Departamento de Ciência da Computação

**INTRODUÇÃO À LÓGICA DE
PROGRAMAÇÃO
MAT 146**

Revisada em Janeiro de 2004

Claudia Gama
claudiag@ufba.br

Caro Aluno:

Esta apostila foi elaborada a partir das minhas notas de aula e dos livros da bibliografia recomendada. Ela visa auxiliar na compreensão dos conceitos abordados ao longo do curso, bem como promover a prática de algoritmos com exercícios de fixação e exercícios propostos. Nela você vai encontrar:

- Resumo dos principais tópicos
- Referências bibliográficas
- Exercícios de fixação
- Exercícios propostos (alguns incluem solução)

Mas atenção: esta apostila deve ser usada somente como material complementar e nunca como única fonte de estudo.

Prof^a. Claudia Gama

Legenda

Na apostila são usados alguns símbolos para chamar a atenção de partes relevantes e ajudar na organização do conteúdo da apostila. Veja os símbolos e seus significados abaixo.



Sugestão de leitura de capítulo ou seções de um livro. A leitura é sugerida para obter-se um entendimento completo do tópico apresentado na apostila.



Sugestões de pesquisas na Internet de assuntos complementares ou curiosidades.



Exercícios de Fixação: são exercícios simples para verificação do entendimento de conceitos.



Sugestão de Exercícios Complementares em livros da bibliografia.



Exercícios Propostos: exercícios para solidificar o entendimento. Altamente recomendáveis!

ÍNDICE







Parte I - Introdução à Lógica de Programação

1.	Algoritmos -----	6
1.1.	Exemplo: Construindo um Algoritmo -----	6
	Y Exercícios Propostos -----	8
2.	Resolução de Problemas através de Computadores -----	9
2.1.	Programação -----	9
3.	Algoritmos Computacionais -----	11
3.1.	Diretrizes para Elaboração de Algoritmos -----	11
3.2.	Descrição de Algoritmos -----	12
3.3.	Estruturas de Dados -----	13
3.4.	Operações Básicas -----	15
	✋ Exercícios de Fixação -----	17
3.5.	Estruturas de Controle -----	18
	✋ Exercícios de Fixação -----	21
3.6.	Considerações sobre o Uso de Variáveis em Algoritmos -----	22
3.7.	Técnicas de Elaboração e Verificação de Algoritmos -----	24
	✋ Exercícios de Fixação -----	25
	Y Exercícios Propostos -----	25

Apêndices – Parte I

1. SOLUÇÕES DE EXERCÍCIOS
2. NOTAÇÃO PARA FLUXOGRAMAS

Parte II - Lógica de Programação com Pascal

4.	Linguagens de Programação de Alto Nível-----	27
4.1.	Tipos de Linguagens de Programação -----	28
4.2.	Compilação e Execução de Programas -----	28
	Exercícios de Fixação-----	29
5.	A Linguagem de Programação Pascal-----	30
5.1.	Estrutura Geral-----	30
5.2.	Estruturas de Dados em Pascal -----	30
5.2.1.	Tipos simples de dados-----	30
5.2.2.	Tipos simples definidos pelo usuario-----	31
5.3.	Estruturas de Controle -----	32
	Exercícios Propostos-----	35
5.4.	Estruturas de Dados : Tipos de Dados Estruturados -----	36
5.4.1.	Vetores -----	36
	Exercícios de Fixação-----	39
5.4.2.	Matrizes -----	40
	Exercícios de Fixação-----	41
5.5.	Algoritmos de Classificação e Busca -----	42
5.6.	Procedimentos e Funções-----	44
	Exercícios de Fixação-----	46
5.7.	Registros e Tabelas-----	49
5.8.	Arquivos Diretos e Seqüenciais -----	51
	Exercícios de Fixação-----	53
5.9.	Variáveis Dinâmicas: Ponteiros -----	55

Apêndices – Parte II

3. SOLUÇÕES DE EXERCÍCIOS - PROGRAMAS PASCAL

Referências Bibliográficas

Forbellone, André L. V. Eberspächer, Henri F. *Lógica de Programação - A Construção de Algoritmos e Estruturas de Dados*; Makron Books. 1993.

Gottfried, Byron B. *Programação em Pascal*; Schaum / McGraw Hill, 1988.

Tremblay, Jean-Paul; Bunt, Richard B. *Ciência dos Computadores - Uma Abordagem Algorítmica*; McGraw Hill. 1983.

Guimarães; Lages *Algoritmos e Estruturas de Dados*; Livros Técnicos e Científicos Editora. 1985

Wirth, Niklaus *Algoritmos e Estruturas de Dados*; Editora PHB, 1986.

Ziviani, Nivio *Projeto de Algoritmos*; Livraria Pioneira Editora

Saliba, Walter L. C. *Técnicas de Programação - Uma Abordagem Estruturada*; Makron Books

Carvalho, Sérgio E. R. *Introdução à Programação com Pascal*; Editora Campus

Farrer, Harry et al. *Pascal Estruturado*; Editora Guanabara Dois. 1985.

Shmitz, Eber; Teles, Antônio *Pascal e Técnicas de Programação*; Livros Técnicos e Científicos Editora. 1986.

PARTE I. INTRODUÇÃO À LÓGICA DE PROGRAMAÇÃO

1. ALGORITMOS



Leitura: “Lógica de Programação” - Forbellone

Cap. 01

Definimos **Algoritmo** como a seqüência de passos que visam atingir um objetivo bem definido.

Os algoritmos são utilizados no dia-a-dia para a solução dos mais diversos problemas.

☑ Alguns exemplos genéricos de algoritmos usados no nosso cotidiano são: uma coreografia, um manual de instruções, uma receita de bolo, a solução de uma equação do 2º grau, uma pesquisa na lista telefônica, etc.

O que todas essas coisas tem em comum?

Elas podem ser vistas como uma serie finita e bem definida de passos ou regras que, quando realizadas, atingem um objetivo previamente definido.

Assim, outra definição para algoritmos poderia ser:

Algoritmo é a descrição de um conjunto de ações que, obedecidas, resultam numa sucessão finita de passos, atingindo um objetivo esperado.

Dessa forma, vemos que, o que importa no algoritmo é o efeito das ações para a obtenção do resultado esperado.

📖 São propriedades de algoritmos:

- ações simples e bem definidas (não ambíguas);
- seqüência ordenada de ações;
- seqüência finita de passos.

1.1 Exemplo: Construindo um Algoritmo

Considere o seguinte problema:

Temos três hastes. Uma das hastes serve de suporte para três discos de tamanhos diferentes. Os discos menores são sempre colocados sobre os discos maiores. A figura abaixo mostra uma possível situação inicial das hastes e discos.



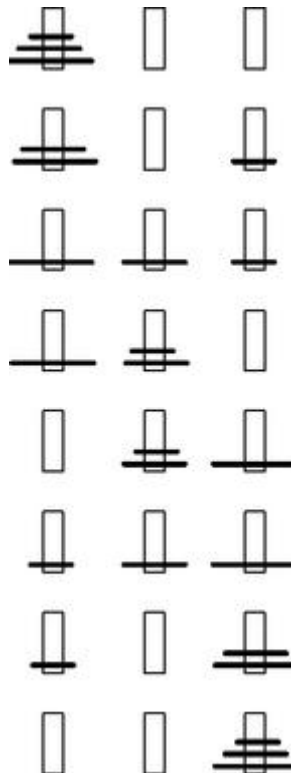
Desejamos mover todos discos para outra haste, porém só podemos movimentar um disco de cada vez e um disco maior nunca pode ser colocado sobre um disco de menor tamanho.

Solução: Em forma narrativa

Nomeamos as hastes como 1, 2 e 3 e os discos como p , m e g . Considera-se que inicialmente os discos estão na haste 1. Os passos são:

1. move o disco p para a haste 2.
2. move o disco m para a haste 3.
3. move o disco p para a haste 3.
4. move o disco g para a haste 2.
5. move o disco p para a haste 1.
6. move o disco m para a haste 2.
7. move o disco p para a haste 2.

Podemos também representar a solução em forma gráfica, desenhando as hastes e a posição dos discos a cada momento (ou passo).



Como podemos ver, com 3 discos precisamos de, no mínimo, 7 passos para solucionar o problema. Quantos passos seriam necessários para se mover 4 discos? E para n discos?

🎁 Curiosidade: Este problema foi criado em 1883 pelo matemático francês Edouard Lucas (1842-1891), que também criou uma lenda curiosa para enunciar o problema. Deixo para vocês procurarem na Internet a lenda das Torres de Hanói.

Exercícios Propostos

P1. Temos três recipientes de tamanhos distintos: o primeiro com capacidade para 8 litros, outro para 5 litros e o terceiro com capacidade para 3 litros. O recipiente de 8 litros está totalmente cheio. Deseja-se colocar 4 litros em dois recipientes. Considere que os recipientes não sejam graduados.

P2. Numa determinada noite, acontece uma queda de energia. Você sabia que poderia encontrar uma vela na gaveta da cozinha, um lampião embaixo da cama, fusíveis de reserva no armário da sala e fósforos na estante da cozinha. Descreva a sequência de passos que poderia ser utilizada para diagnosticar e resolver o problema, o que pode ser previsto em duas possibilidades:

- a) o fusível queimou;
- b) a queda é na estação da companhia elétrica.

Exercícios Complementares

Faça pelo menos 02 Exercícios Propostos do Livro do Forbellone – Cap. 1




Visite o site da disciplina e tente resolver os problemas de logica sugeridos:

<http://www.im.ufba.br/mat146/ProblemaLogica>

2. RESOLUÇÃO DE PROBLEMAS ATRAVÉS DE COMPUTADORES

Os computadores podem ser usados de forma eficiente na solução de certos tipos de problemas. Os problemas que suportam tratamento por computador, em geral, envolvem grandes quantidades de dados ou são problemas de natureza complexa, exigindo a execução de um grande número de passos para alcançar a solução. Basicamente são problemas na área de processamento de dados e na área científica.

 O Computador é uma **ferramenta** que permite a realização do processamento automático (ou eletrônico) de dados.

Define-se por Processamento de Dados qualquer atividade que, utilizando informações (ou dados), efetua transformações para obter novas informações (ou dados) como resultado.



Porém, a tarefa desempenhada pelos computadores é apenas parte do processo de solução de problemas.

As etapas na solução de problemas são:

- i) Entendimento do problema;
- ii) Criação de uma seqüência de operações (ou ações) que, quando executadas, produzem a solução para o problema;
- iii) Execução desta seqüência de operações.
- iv) Verificação da adequação da solução.

As etapas de *entendimento do problema*, *criação de seqüência de ações* e *verificação da adequação da solução* são tarefas desempenhadas por pessoas. Já a *execução das operações* pode ser desempenhada por computadores.

Os computadores tem a capacidade de executar processos complicados e com grande quantidade de informações com **rapidez** e **confiabilidade**.

2.1 Programação

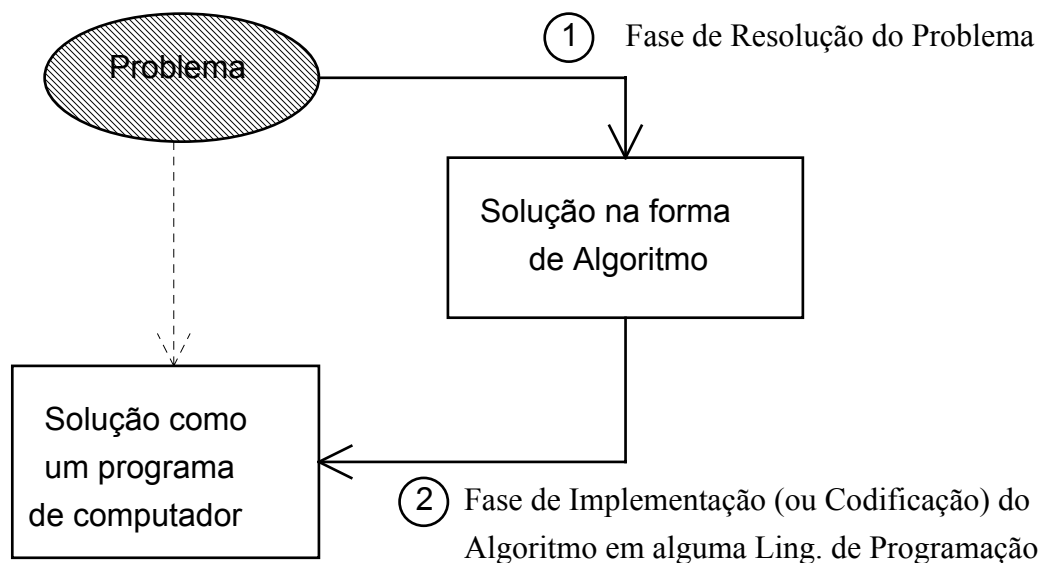
Programação é a seqüência de planejamento, projeto, escrita e testes de instruções desempenhadas pelo computador. É uma arte e uma ciência. Arte porque existem muitas maneiras de se realizar o trabalho de programação. Existe espaço para uma considerável dose de criatividade. É também uma ciência, porque existem algumas regras que devem ser seguidas, porque é necessário o uso de lógica e porque existem alguns métodos rigorosos de programação que asseguram a eficiência, economia e a utilidade dos programas gerados.

O trabalho de programação pode se tornar mais fácil se o dividirmos sistematicamente em partes menos complexas (esta técnica denomina-se “*dividir para conquistar*”).

Um programa é considerado **confiável** quando conseguir fazer com que o computador cumpra com o objetivo proposto. Os programas construídos devem ser **eficazes**, realizando a tarefa definida e **eficientes**, utilizando os melhores meios para realizá-la.

O maior problema na construção de programas é a complexidade; esta complexidade representa a quantidade de situações diferentes que um problema pode apresentar e que devem ser previstas na solução do mesmo. Portanto, ao se construir um programa, o objetivo principal é vencer a complexidade do problema a ser solucionado.

A fim de lidar com esta complexidade, podemos dividir a programação em duas fases distintas:



① Modelização (ou Resolução) do Problema : determinação do modelo de solução para o problema proposto na forma de um **algoritmo computacional**.

Assim, a elaboração de um algoritmo é o primeiro passo para a preparação de um programa de computador. Este algoritmo deve ser independente da linguagem de programação que será utilizada.

② Implementação: é a transformação (ou codificação) do algoritmo em alguma **Linguagem de Programação** adequada ao modelo elaborado.

Leitura complementar: definição de Programação no livro “Algoritmos e Estruturas de Dados” (Wirth)

Definições Importantes:

"Programas são formulações completas de algoritmos abstratos, baseados em representações específicas de dados." [Wirth]

"Programação Estruturada é a arte ou técnica de construir e formular algoritmos de forma sistemática." [Wirth]

✧ Quem foi Nicklaus Wirth? Em 1984 ele recebeu o mais prestigioso prêmio técnico da ACM (Association for Computing Machinery) – o “A.M. Turing Award” por sua valiosa contribuição para a computação. O resto deixo para vocês pesquisarem. Aproveite e procure saber também sobre a ACM. Qual a relevância dessa associação? O que ela faz?

3. ALGORITMOS COMPUTACIONAIS

O algoritmo é uma sequência de instruções, onde cada instrução representa uma AÇÃO que deve ser entendida e realizada. Surge então uma Questão:

Como saber se já temos detalhes suficientes em um algoritmo para que a AÇÃO possa ser entendida e realizada ?

Vai depender do agente que irá executar o Algoritmo. No caso de **algoritmos computacionais**, sabemos que o computador possui um conjunto limitado de instruções e o algoritmo deve ser expresso nos termos destas instruções.

O computador utiliza dois conceitos básicos para construir e interpretar algoritmos:

- ✓ Estruturas de Dados ⇒ para manipulação das informações
- ✓ Estruturas de Controle ⇒ para manipulação das ações

3.1 Diretrizes para a Elaboração de Algoritmos

As diretrizes apresentadas abaixo são genéricas e podem ser usadas ou adaptadas na organização dos passos que comporão a solução de um determinado problema (ou seja, na criação de um algoritmo para atingir um objetivo determinado).

1. Identificação do problema: determinar o que se quer resolver ou qual objetivo a ser atingido.
2. Identificação das “entradas de dados”: informações fornecidas, a partir das quais se desenvolverão os cálculos.
3. Identificação das “saídas de dados”: as informações a serem geradas como resultado.
4. Identificação das regras e limitações do problema ou das limitações do agente executante (ex: se o agente fosse uma calculadora não-científica, iriam existir limitações no cálculo de funções, por exemplo).
5. Determinação do que deve ser feito para transformar as “entradas” em “saídas”. Neste ponto deve ser determinada a sequência de ações que leve à solução do problema. Para isto é preciso:
 - 5.1. observar as regras e limitações já identificadas;
 - 5.2. determinar ações possíveis de serem realizadas pelo agente.

6. Construção do Algoritmo, utilizando uma das formas de representação de algoritmos (ver Tópico 3.2)
7. Teste da solução - execução de todas as ações do algoritmo, seguindo o fluxo estabelecido para verificar se ele está realmente gerando os resultados esperados ou detectar possíveis erros em sua descrição (veja detalhes sobre Teste de Algoritmos no Tópico 3.7)

Exemplo:

Imagine o seguinte problema: Calcular a média final dos alunos da 6ª Série. Os alunos realizarão quatro provas: P1, P2, P3 e P4. A Média Final é calculada por:

$$(P1 + P2 + P3 + P4) / 4.$$

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)

$$(P1 + P2 + P3 + P4)/4$$

c) *Quais serão os dados de saída?*

R: O dado de saída será a média final

3.2 Descrição de Algoritmos

A descrição de um algoritmo de forma clara e fácil de ser seguida ajuda no seu desenvolvimento, depuração (correção de erros) e a subsequente transformação do mesmo num programa.

❑ Descrição Narrativa

Especificação verbal dos passos em linguagem natural.

Desvantagens: a linguagem natural é prolixa e imprecisa e frequentemente pouco confiável como um veículo de transferir informação.

Sua utilização pode ser adotada, entretanto, para a apresentação de comentários sobre o algoritmo (ou parte dele), esclarecendo ou realçando pontos específicos.

❑ Fluxograma

Uso de ilustrações gráficas para transmitir informações (Ex. Gerais: mapas, diagramas explicativo para montagem de aparelhos, etc.).

Um fluxograma mostra, de forma gráfica, a lógica de um algoritmo, enfatizando passos individuais e o fluxo de execução.

Desvantagens: utilização questionável de fluxogramas detalhados, pois *obscurecem* a estrutura do programa.

Para algoritmos computacionais usaremos os Diagramas de Nassi-Schneiderman (*Veja Apêndice II no final*).

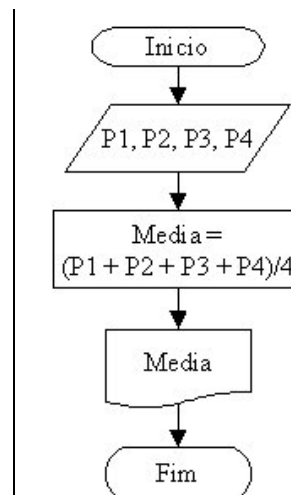
❑ Linguagem Algorítmica ou Pseudolinguagem

Linguagem especial para expressão de algoritmos; funciona como uma "linguagem simplificada de programação", utilizando expressões concisas e pré-definidas para representar as ações e o fluxo de execução. É uma descrição textual, estruturada e regida por regras que descrevem os passos executados no algoritmo. Utiliza-se palavras-chaves, indentação, apenas um passo por linha, normalmente usa-se um símbolo para indicar o final de um passo (como por exemplo o ponto-e-virgula “;”), etc.

Usaremos o Portugol, pois ela possui uma estrutura similar ao Pascal.

☑ Exemplo: Duas representações para o algoritmo do problema das medias dos alunos da 6ª Série.

1. Receba a nota da prova1
2. Receba a nota da prova2
3. Receba a nota da prova3
4. Receba a nota da prova4
5. Some todas as notas e divida
o resultado por 4
6. Mostre o resultado da divisão



3.3 Estruturas de Dados



Leitura: “Lógica de Programação” - Forbellone

Cap. 02

Um aspecto fundamental na construção de algoritmos computacionais são as **estruturas de dados**, que representam as informações do problema a ser resolvido.

Tais estruturas estão organizadas em tipos distintos de informações. Dentro do escopo das estruturas de dados, definimos os termos CONSTANTE, VARIÁVEL e IDENTIFICADOR.

❑ Tipos Primitivos de Dados

São os grupos de informações que o computador manipula. Podem ser:

1. Numéricos
 - a) inteiros Ex: 1 -4 100 0 -905 ...
 - b) reais Ex: 1,3 816,97 3,0 -0,0055 ...
1. Não-numéricos
 - a) alfanuméricos Ex: "CASA" "livro" "18" 'R\$ 55,36'
 - b) lógicos ou booleanos Ex: Falso, Verdadeiro (ou False, True)

☑ Obs: O delimitador usado para alfanuméricos pode ser : “ ” ou ‘ ’

☐ Constantes

Representam valores constantes, ou seja, que não variam no decorrer do algoritmo.

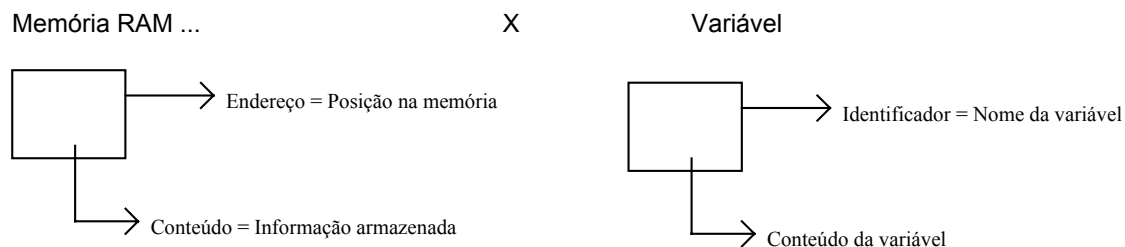
Ex: 148, "Opção:", -8.69, "Tecle algo para continuar", Falso ...

☐ Variáveis

Representam informações cujos valores são modificados ao longo do tempo (Ex. Genérico: a idade de um indivíduo). Podemos definir variáveis também como um local onde um determinado valor (de um dos tipos definidos) é armazenado. Assim, a variável é composta de dois elementos básicos:

- ✓ conteúdo - valor atual da variável
- ✓ identificador - nome dado à variável para possibilitar sua manipulação

O conceito de **variável**, na verdade, corresponde a “posições de memória RAM”, onde serão armazenados os dados manipulados pelo programa quando este for implementado.



☐ Identificador


Nome de um local onde se pode colocar qualquer valor do conjunto de valores possíveis de um tipo básico associado. Usado para manipular todos os dados variáveis do algoritmo. Pode também ser usado para rotular valores constantes (ex: uso do identificador PI para representar o valor constante 3,14).

Regras para Definição de Identificadores em Pseudolinguagem (Portugol):

- ✓ devem começar por um caracter alfabético (uma letra);
- ✓ pode ser seguido de letras e/ou dígitos;
- ✓ é permitido o uso do caracter especial “_” situado entre letras e/ou dígitos.

Ex. de identificadores válidos: Operador, Nome, X, y, ENDEREÇO, Aluno_01

- ☒ O identificador deve representar de forma significativa o conteúdo desejado, ou seja, deve ser um MNEMÔNICO.

 Procure a definição de “mnemônico” num dicionário. Depois descubra como este termo e’ empregado em programação. O que são variáveis mnemônicas?

❑ Declaração de Variáveis

É a criação (ou definição) de locais na memória rotulados com o identificador da variável (ou constante) que será utilizada no algoritmo para a manipulação de um determinado tipo de informação.

Sintaxe Geral em Portugol:

Lista de Variáveis : Tipo ;

O Tipo pode ser inteiro, real, caracter (alfanumérico) ou booleano.

Ex: NUM, X : inteiro;
 parcela : real;
 resposta : booleano;
 PI : real; { definição de uma constante real }
 NOME, ENDEREÇO : caracter;

- ☒ Neste momento, as variáveis ainda não contém nenhum valor associado a elas.
- ☒ A declaração de variáveis deve ser feita antes da definição das ações do algoritmo.
- ☒ Os comentários representam qualquer texto que explique uma ação ou um dado usado no algoritmo. Eles podem ser colocados em qualquer ponto do algoritmo, utilizando { } ou * * como delimitadores.

3.4 Operações Básicas

❑ Comando de Atribuição

Serve para atribuir (ou associar) um valor a uma variável ou constante.

Sintaxe Geral em Pseudolinguagem:

identificador ← expressão ;

onde *expressão* pode ser um(a): Constante, Variável, Expressão matemática, Função matemática, Expressão booleana, etc.

Ex: NOME ← ‘Fulano de Tal’
 PI ← 3.14
 Erro ← Verdadeiro (ou, simplesmente, Erro ← T)
 Media ← (P1 + P2)/2

❑ Funções Matemáticas

nome da função (argumento)

☑ O *argumento* é a informação que é dada à função para que ela possa ser efetuada adequadamente. Pode ser uma constante ou uma variável.

Exemplos de Funções Matemáticas que utilizaremos:

Sin (X)	- Função que calcula o seno da variável X
SQRT (y)	- Função que calcula a raiz quadrada de y
int (z)	- Função que retorna a parte inteira da variável z (que deve ser do tipo real)


❑ **Operadores Aritméticos**

+ -	(unários)
**	exponenciação
*	multiplicação
/	divisão
+ -	soma e subtração (binários)
div	divisão truncada (ou inteira) (ex: $x \text{ div } y$)
mod	resto da divisão inteira

❑ **Operadores Relacionais** < <= > >= = <> (diferente)

❑ **Operadores Lógicos** not and or

** Podemos usar parênteses para alterar ordem de prioridade das operações.

 Ver resumo dos operadores e a ordem de prioridade das operações em “Lógica de Programação” – Forbellone Cap. 02

❑ Comandos de Entrada e Saída de Dados

Representam as ações básicas de algoritmos para recebimento e apresentação de dados, respectivamente.

Entrada de Dados: informações que são fornecidas ao programa pelo “usuário” durante a resolução do problema.

Representação em Portugol:

leia (identificador, identificador, ...);

Ex: leia (altura, idade, sexo);

Saída de Dados: informações que são mostradas ao usuário como resposta ao problema.

Representação em Pseudolinguagem:

Escreva (expressão, expressão, ...);

Ex: escreva (' Seu peso ideal eh : ' , peso_ideal) ;
 escreva (' Media final = ' , (P1 + P2)/2) ;

❑ Esquema Genérico de Algoritmos em PseudoLinguagem

Declaração de Variáveis / Constantes INICIO inicialização de variáveis / constantes {comentários} bloco de comandos de entrada de dados {comentários} bloco de comandos de cálculo {comentários} bloco de comandos de saída de dados FIM

Exemplo: Algoritmo em pseudolingugem para o problema do cálculo da média final dos alunos da 6ª Série.

```

* declaração de variáveis
Real: P1, P2, P3, P4, Media;
Inicio
    * comandos de entrada de dados
    Leia (p1);
    Leia (p2);
    Leia (p3);
    Leia (p4);
    * processamento- Calculo da media
    Media ← (P1 + P2 + P3 + P4)/4;
    * saída de dados
    Escreva ( ' Media final = ' , Media);
Fim
    
```

Exercícios de Fixação

F1. Faça 02 exemplos para cada um dos conceitos abaixo:

- entrada de dados
- saída de dados
- declaração de variáveis
- inicialização de variáveis
- atribuição do resultado de uma expressão aritmética a uma variável

F2. Utilizando tipos primitivos de dados, crie declarações de variáveis, que armazenem as seguintes informações: o nome de uma figura geométrica, a quantidade de lados, a área, o perímetro e se a figura é regular ou não.

 Exercícios Complementares: Livro Forbellone - Cap. 02 - Exercícios Propostos

3.5 Estruturas de Controle: Sequencial, Condicional e de Repetição

	Leitura: “Lógica de Programação” - Forbellone	Cap. 03
---	---	---------

❑ Estrutura de Controle Seqüencial

Conjunto de comandos que são executados numa seqüência linear, de cima para baixo, na mesma ordem em que aparecem.

Sintaxe Geral:

Comando 1; Comando 2; Comando 3;
--

❑ Estrutura de Controle Condicional ou de Seleção

Permite a escolha de um grupo de ações para serem executadas de acordo com a aceitação ou não de certas condições.

São testados parâmetros e, a depender de seus valores, tomamos um caminho ou outro.

As condições que são testadas num Algoritmo são do tipo lógica (booleana), portanto podem ter somente dois resultados: Verdadeiro ou Falso. Assim, a seleção de ações pode seguir, no máximo, duas alternativas: uma se a condição for verdadeira e outra se a condição testada for falsa. Existem seleções mais simples e seleções compostas outras seleções.

Seleções Simples

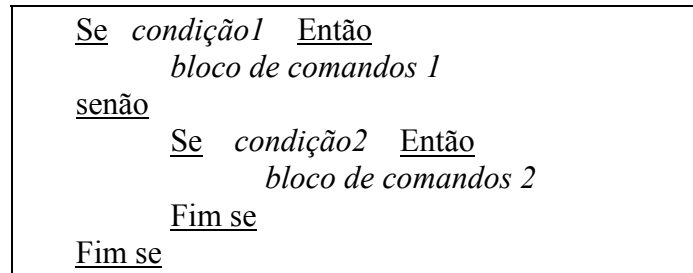
Opção 1: Sintaxe Geral

<u>Se condição</u> <u>Então</u> <i>bloco de comandos 1</i> <u>Fim se</u>
--

Opção 2

<u>Se condição</u> <u>Então</u> <i>bloco de comandos 1</i> <u>senão</u> <i>bloco de comandos 2</i> <u>Fim se</u>
--

Seleções Compostas: Aninhamento de condições



☑ Observação: as palavras sublinhadas na Sintaxe Geral são chamadas de “palavras reservadas” e sempre aparecem na estrutura. As palavras em *itálico* são indicações do tipo de expressão ou comando que devem ser usados.

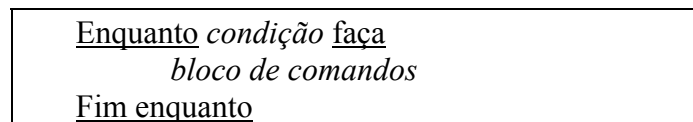
❑ Estrutura de Repetição

Podemos ter repetições condicionais (a repetição ocorre condicionada a uma condição lógica) e incondicionais (que tem um número pré-definido de repetições).

a) Repetição Condicional: existem dois tipos distintos de Repetição Condicional.

Tipo 1 - Condição testada no início da repetição

Sintaxe Geral:



Características:

- Testa a condição antes da execução do bloco.
- Enquanto a condição for verdadeira, o bloco de comandos é executado. Assim, o bloco de comandos pode ser executado 0 ou mais vezes.
- Para a execução do bloco quando a condição se tornar falsa.

☑ Exemplo:

Elabore um algoritmo para determinar o menor número fornecido de um conjunto de valores inteiros positivos dados. Considere que o número *zero* indica o encerramento do conjunto de dados de entrada.

valor, menor : inteiro

Início

leia (valor)

menor ← valor * *inicialização da variável que vai guardar o menor valor* *

Enquanto (valor < > 0) faça

 Se valor < menor então

 menor ← valor

```

        fim se
        leia (valor) * entrada do próximo elemento do conjunto *
    fim enquanto
    escreva (' O menor valor do conjunto é ', menor)
Fim
    
```

Tipo 2 - Condição testada no final da repetição

<u>Repita</u> <i>bloco de comandos</i> <u>até condição</u>
--

Características:

- Testa a condição após da execução do bloco.
- Enquanto a condição for verdadeira, o bloco de comandos é executado. Assim, o bloco de comandos é executado pelo menos uma vez.
- Para a execução do bloco quando a condição se tornar verdadeira (denominada de Condição de Parada).

b) Repetição Incondicional - N.º pré-definido de repetições

Sintaxe Geral:

<u>Para</u> <i>variável de controle</i> = <i>valor inicial</i> <u>até</u> <i>valor final</i> <u>Faça</u> <i>bloco de comandos</i> <u>Fim para</u>

- Repete o bloco de comandos (*valor final* - *valor-inicial* + 1) vezes
- Incrementa **automaticamente** a *variável de controle* cada vez que o bloco é executado (incremento “default” de 1 até alcançar o *valor final*)
- Se o *valor final* definido for menor que o *valor inicial*, o laço de repetição não é executado nenhuma vez.
- A *variável de controle* deve ser do tipo primitivo **inteiro**.
- A variável usada como controle da estrutura **não pode ser modificada dentro do bloco!**

☑ Exemplo:

Elabore um algoritmo para calcular o fatorial de N, onde N é um número inteiro (maior ou igual a zero).

Considere que: Se $N > 0$ então $N! = 1 \times 2 \times 3 \times \dots \times N$
 $N = 0$ então $N! = 1$

Algoritmo em Pseudolinguagem

n, fat, acum : inteiro

{ *acum e' a variável de controle* }

Início

leia (n)

* *inicialização de fat*

fat \leftarrow 1

Para acum = 2 até n faça

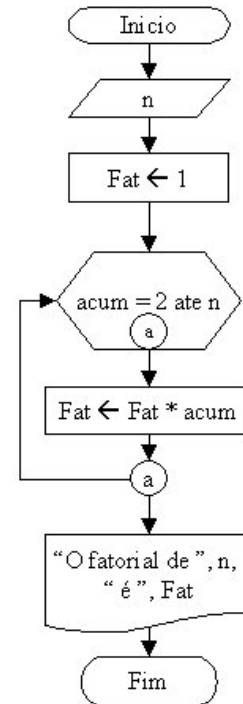
fat \leftarrow fat * acum

fim para

escreva (' o fatorial de ', n ,
' é ', fat)

Fim

... e em Fluxograma:



Teste da Solução: Testando o algoritmo acima para os seguintes valores de n: 0, 1 e 3

N	Fat	acum	Escreva ...
0	1	X	fatorial = 1
1	1	X	fatorial = 1
3	1	2	
	1*2	3	
	1*2*3	4	fatorial = 6



Exercícios de Fixação

F1. Faça 02 exemplos para cada um dos conceitos abaixo:

- condição lógica composta
- estrutura de seleção simples
- estrutura de seleção encadeada
- estrutura de seleção de múltipla escolha

F2. Observe o algoritmo e responda:

```

X, N : inteiro
Inicio
    leia (N , X)
    Y → 1
    Enquanto ( X > 0 ) faça
        inicio
            Y → Y * N
            X → X - 1
        fim
    Fim enquanto
    escreva ( Y )
Fim
    
```

- a) Qual o objetivo do algoritmo acima (i.e.: qual o problema que ele está solucionando)?
- b) O algoritmo está correto? Como você fez para testá-lo?
- c) Altere o algoritmo, utilizando a Estrutura *Repita ... Até*
- d) Altere o algoritmo, utilizando a Estrutura *Para ... Faça*
- e) Qual das três opções de algoritmo é a melhor na sua opinião? Por que?

3.6 Considerações sobre o Uso de Variáveis em Algoritmos Computacionais

As variáveis representam a informação manipulada pelo algoritmo, e portanto, pode aparecer em diversas situações:

1. Armazenar dados que são fornecidos pelo usuário

Ex: leia (**DIA, MES, ANO**)

2. Armazenar resultados de expressões

Ex: **RESULT** ← (A + B * C) / (D - E)

3. Acumuladores: Acumular valores

Ex: **ACUMULA** ← 1
 Enquanto **ACUMULA** < 100 faça
 leia (NUM)
 ACUMULA ← **ACUMULA** + NUM
 Fim enquanto

4. Contadores: Contar valores

Ex: **CONTA** ← 0
 Repita
 CONTA ← **CONTA** + 1
 escreva ("Repeti o laço ", **CONTA**, " vezes ")
 até **CONTA** > 20

5. Finalizadores: Finalizar repetições

Ex₁: leia (**NUM**)
 Enquanto **NUM** < > 0 faça *{condição para parada »» NUM = 0 }*
 Seno ← SIN (NUM)
 escreva (Seno)
 leia (**NUM**)
 Fim enquanto

Ex₂: Repita
 leia (oper1, oper2)
 divis ← oper1/oper2
 escreva (' Continuar (sim/nao) : ')
 leia (**resp**)
 ate **resp** = 'nao' *{condição de parada}*

6. Sinalizadores ou Flags: Sinalizar erros, condições de falha de teste, etc.
 Variável que informa que uma determinada situação foi alcançada.

Ex: **INVALIDA** ← 0
 Se (dia<1 or dia>31) or (mes<1 or mes>12) então
 INVALIDA ← 1
 senão

 fim se
 Se **INVALIDA** = 0 então
 escreva (' Data válida ')
 senão
 escreva (' Data inválida ')
 fim se

- ☒ Muitas vezes são usadas variáveis do tipo booleano como sinalizadores.

Ex: **INVALIDA**: booleano

...
 Leia (dia, mes, ano)
** inicializa a variavel booleana com um valor False ou True*
INVALIDA ← False
 Se (dia<1 or dia>31) então
 ** Muda o valor do flag para sinalizar um erro ou mudança de estado*
 INVALIDA ← True
 senão
 Se (mes<1 or mes>12) então
 INVALIDA ← True
 fim se
** Note que a condição não usa operadores lógicos do tipo (Invalida = True)*
 Se **INVALIDA** então
 escreva (' Erro na Data - dia invalido ou mes invalido ')
 fim se

3.7 Técnicas de Elaboração e Verificação de Algoritmos



Leitura: “Ciência dos Computadores - Uma abordagem algorítmica” - Tremblay
Cap. 07

❑ Refinamentos Sucessivos “top-down”

Esta é uma técnica de elaboração de algoritmos que divide o desenvolvimento do mesmo em diferentes fases. O problema inicial é subdividido em subproblemas menores, e estes em partes ainda menores, e assim sucessivamente. A cada divisão são levados em conta mais detalhes sobre a especificação do problema. Esta é uma maneira de lidar com a complexidade do problema. Mais adiante, quando falarmos de Procedimentos e Funções veremos na prática o uso de Refinamentos Sucessivos Top-Down.

❑ Regras para tornar seu algoritmo mais claro e legível

a) Utilize comentários.

Escreva os comentários no momento que estiver escrevendo o algoritmo. Um programa mal documentado é um dos piores erros que um programador pode cometer. O melhor momento para se escrever os comentários é aquele em que o programador tem maior intimidade com o algoritmo, ou seja, durante a sua confecção. Existem 02 tipos de comentários que devem ser usados.

Prólogo ou Cabeçalho: são comentários que contêm uma identificação geral do algoritmo.

- O que faz o algoritmo (programa ou módulo);
- Como chamá-lo ou utilizá-lo;
- Significado dos parâmetros, variáveis de entrada, de saída e variáveis mais importantes;
- Arquivos utilizados;
- Outros módulos utilizados;
- Métodos especiais utilizados, com referências nas quais possa se encontrar mais informações;
- Autor, data de escrita e última atualização;

Comentários de linha: são comentários de passos que podem ser mais obscuros para o leitor, como o uso de variáveis como acumuladores, contadores, flags, etc.

b) Utilize identificadores mnemônicos: escolha nomes representativos para variáveis, funções, constantes, tipos, etc. Evite usar letras quando a variável representa algo concreto (ex: $X \leftarrow Y + Z$ é muito menos claro que $\text{Preço} \leftarrow \text{Custo} + \text{Lucro}$). Também evite identificadores longos (ex: use *nome* ou *nAluno* ao invés de *nome_do_aluno* para representar “nome de aluno”).

c) Utilize indentação para mostrar a estrutura lógica do programa. Crie suas regras básicas de indentação e procure segui-las ao escrever um algoritmo. Uma boa regra é indentar blocos de comandos internos a uma estrutura de controle.

d) Utilize espaços em branco para melhorar a legibilidade. Os espaços em branco são valiosos para melhorar a aparência de um programa. Você pode por exemplo:

- Deixar uma linha em branco entre as declarações e o corpo do programa;
- Separar grupos de comandos que executam funções lógicas distintas por uma ou mais linhas em branco.

Um comando por linha é suficiente. A utilização de vários comandos por linha é prejudicial por vários motivos, dentre eles destacam-se o fato do programa tornar-se mais ilegível e ficar mais difícil de ser depurado.

❏ Testes

Um tipo de teste (denominado Teste de Mesa ou Teste Exaustivo) de um algoritmo pode ser feito através de uma simulação do mesmo, aonde são dados valores para os dados de entrada e vai-se preenchendo uma tabela aonde são colocadas todas as variáveis e constantes do algoritmo. Segue-se o fluxo de execução (como se estivesse executando o algoritmo em um computador imaginário). A cada comando de atribuição ou cálculo de expressão o valor das variáveis deve ser atualizado. Ao final do teste podemos ter as seguintes situações:

- ✓ o resultado esperado foi alcançado; ou
- ✓ foram detectados erros nos comandos; ou
- ✓ foram detectados erros no fluxo de execução - erro nas estruturas de controle.

Procure fazer testes relevantes como, por exemplo, aqueles que verificam casos extremos e casos de exceções. Com o teste é possível identificar se e em que ponto o algoritmo está falhando e fazer a correção. Algoritmos errados podem gerar resultados errados ao serem executados por um computador ou mesmo não gerar nada, se o computador detectar erros graves na sequência de operações.



Exercícios de Fixação

Teste o algoritmo abaixo. Faça um **Teste de Mesa**, usando a tabela ao lado do algoritmo.

👉 lembre-se de testar os “casos críticos” (casos extremos e casos de exceções)!

X, N : inteiro

Início

leia (N , X)

$Y \rightarrow 1$

Enquanto (X > 0) faça

início

$Y \rightarrow Y * N$

$X \rightarrow X - 1$

fim

Fim enquanto

escreva (Y)

Fim

N	X	Y	(X > 0)



Exercícios Propostos

P1. Elabore um algoritmo que verifique se um número positivo é primo ou não. Faça um teste exaustivo da solução encontrada.

- P2. Elabore um algoritmo que calcule os 20 primeiros termos da Série de Fibonacci. A série de Fibonacci é formada pela seguinte seqüência: 1, 1, 2, 3, 5, 8, 13, ... etc. Inclua um teste de mesa do algoritmo.
- P3. Determine o maior e o menor valor de um conjunto de números inteiros positivos. Considere que o conjunto de dados de entrada termina quando é fornecido o número -5.
- P4. Escreva um algoritmo que leia n números inteiros e determine se cada um deles é um número da seqüência de Fibonacci ou não.
- P5. O algoritmo abaixo tem como objetivo determinar o valor do somatório S, dado pela série $S = X - X^2 / 3! + X^4 / 5! - X^6 / 7! + \dots$ usando os 20 primeiros termos da série, porém o algoritmo não está correto. Corrija os erros encontrados (se preciso, rescreva o algoritmo) e acrescente comentários para aumentar a legibilidade do algoritmo. Faça um Teste de Mesa.

```

inteiro : X , F , S
Inicio
    leia ( X )
    S ← 1
    Fat ← 1
    Para I = 1 até 20 faça
        Para F = 1 até ( 2 * I ) faça
            Fat = Fat * F
        Fim Para
        S ← ( S + ( X ** ( 2 * I ) ) ) / Fat
    Fim Para
    Escreva ( " O somatório é' " , S )
Fim
    
```

- P6. Elabore um algoritmo que leia uma massa de dados contendo SEXO , DATA DE NASCIMENTO e ESTADO CIVIL (Casado/ Solteiro/ Divorciado/ Outros) de um grupo de 100 pessoas e determine, ao final:
- Média de Idade das mulheres
 - Estado Civil **mais** prevalente entre os entrevistados e o de **menor** ocorrência
- P7. Foi feita uma pesquisa de audiência de canal de TV em várias casas numa certa cidade, num certo dia. Para cada casa visitada é fornecido o número do canal (4, 5, 7, 11) e o número de pessoas que estavam assistindo TV. Elabore um algoritmo que leia um número indeterminado de dados (terminando quando for lido um canal igual a zero) e calcule a percentagem de audiência para cada emissora, mostrando ao final, o número de cada canal e sua respectiva audiência.

No Apêndice I encontra-se a solução de alguns dos exercícios propostos

Exercícios Complementares

Livro do Forbellone -

Cap. 03

Exercícios de Fixação e Exercícios Propostos