

UNIÃO EDUCACIONAL DO NORTE – UNINORTE
FACULDADE BARÃO DO RIO BRANCO – FAB
CURSO DE SISTEMAS DE INFORMAÇÃO
DOCENTE: LUIZ AUGUSTO MATOS DA SILVA



DISCENTE: _____

MATERIAL DE APOIO DA DISCIPLINA

ALGORITMOS

2º. Período / 2006.2

Rio Branco, Agosto de 2006.

APRESENTAÇÃO

Este material é direcionado a você, aluno do Curso de Sistemas de Informação, de maneira que sirva como fonte de pesquisa e apoio à disciplina Algoritmos.

O conteúdo apresentado é resultado de referencial teórico extraído das fontes bibliográficas recomendadas sobre o tema, com o apoio de artigos técnicos e demais publicações.

Na primeira parte objetiva-se a abordagem dos princípios básicos inerentes à Lógica de Programação, dando ênfase às formas de representação e o processo de programação. No capítulo 2, o conteúdo apresentado traz os conceitos de variáveis, constantes, tipos primários de dados com o desenvolvimento de algoritmos através da técnica do Português Estruturado. O capítulo 3 trata dos comandos de entrada, saída, estruturas de controle (simples, condicionais e repetição). O quarto capítulo, lida com o tratamento das estruturas de dados homogêneos: vetores e matrizes, além de métodos de busca e ordenação em tais estruturas. Finalmente, a última parte deste material aborda os conceitos de modularização em algoritmos, com ênfase nas técnicas de análise e desempenho.

Ao término deste curso o aluno deverá ter, além de um raciocínio lógico apurado para a resolução de problemas envolvendo a lógica de programação, o domínio teórico-prático dos conceitos básicos de análise e desenvolvimento de algoritmos, de maneira crítica e sistemática.

Bom estudo!
luizmatos.eti.br



Conceitos de Lógica de Programação

1.1 – Introdução

Lógica de programação
Solucionando os problemas

1.2 – Formas de representação

Linguagem Narrativa
Fluxograma
Diagrama de Chapin
Português Estruturado
Definindo uma forma de representação

1.3 – Processo de Programação

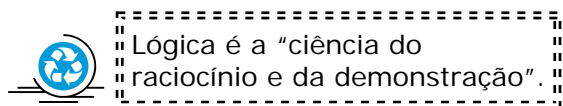
Tipos e etapas da programação

1.4 – Exercícios Propostos

“Toda aritmética pode ser compreendida pelo mecanismo de um pequeno motor analítico.”
(Babbage)

1.1 – Introdução

A lógica faz parte de nosso cotidiano, do nosso pensar, fazer, decidir. Entende-se por lógico o que não permite questionamento; o que tem coerência, é óbvio, certo.



Lógica de programação

A lógica de programação é a maneira pela qual se representa em linguagem convencional (compreensível), instruções que compõem um programa¹ a ser executado por um computador. É o raciocínio lógico do programador² que vai influenciar diretamente o seu produto final, o programa.

Um programa, antes de seu pleno funcionamento, passa pelas seguintes etapas:

o **Levantamento e análise de dados.** Diante da necessidade do usuário de informatização, realiza-se o levantamento de dados e análise das informações, com o objetivo de fazer um projeto de viabilidade.

¹ Conjunto de instruções feitas para serem executadas por um computador.

² Responsável pela(s) etapa(s) de criação de um programa de computador.

o **Projeto lógico.** Serve como base para a fase de desenvolvimento. Divide-se o sistema em subsistemas e representa-os em diagramas, no intuito de tornar o projeto mais claro e possibilitar uma melhor codificação.

o **Projeto físico.** A lógica do programa é representada através de **ALGORITMOS**, etapa por etapa, após os devidos testes até a codificação.

Finalmente, realiza-se a devida documentação, implantação e testes do programa.

Solucionando os problemas

Para solucionar com precisão os problemas de programação devemos ter em mente os seguintes passos:

- o **Interpretar** o problema proposto;
- o **Identificar** os dados disponíveis para a sua resolução;
- o **Dividi-lo** em partes menores, mediante a sua complexidade;
- o **Definir** seu objetivo. Pergunte-se: "Qual o resultado que se deseja alcançar?";
- o Se mesmo assim não encontrar a solução, **reveja** os passos anteriores.



```

=====
enquanto não ENTENDEU faça
  se VEZES <= 4 então
    escreva "Leia a especificação!";
    VEZES ← VEZES + 1;
  senão
    escreva "Pergunte ao professor!";
    ENTENDEU ← verdadeiro;
  fim se;
fim enquanto;
=====

```

1.2 – Formas de representação

São diversas as formas de representação da lógica de programação. Devendo ser todo problema analisado antes de se encontrar a solução, uma das técnicas mais utilizada é o **ALGORITMO** e suas formas de representação.

Linguagem Narrativa

Pode-se considerar o algoritmo como uma **"seqüência de procedimentos finitos, que sendo executados em determinado período de tempo, chegará ao seu objetivo"**.

Ou em outras palavras: **"Um algoritmo é a descrição de um padrão de comportamento, expressado em termos bem definidos, finito de ações executáveis"**.

Imaginemos, por exemplo, a seqüência que tem por objetivo "fazer café".

1. encha de água a chaleira;
2. coloque a chaleira para ferver;
3. prepare o porta-filtro com o filtro sobre o bule;
4. coloque duas colheres de sopa de pó de café no filtro;
5. após a água ter fervido, acrescente aos poucos meio litro de água sobre o filtro;
6. aguarde coar;
7. adoce a gosto.

Sendo esses passos seguidos corretamente, o objetivo de "fazer café" será alcançado. Mas, se um dos componentes faltasse? Certamente, a seqüência seria descrita de outra forma.

1. encha de água a chaleira;
2. coloque a chaleira para ferver;
3. enquanto a chaleira está no fogo;
4. prepare o porta-filtro com o filtro sobre o bule;
5. se houver café, coloque duas colheres de sopa de pó no filtro;
6. senão, desligue a chaleira, vá até o mercado comprar café;
7. repita os itens 2, 3 e 5;
8. após a água ter fervido, acrescente aos poucos meio litro de água sobre o filtro;
9. aguarde coar;
10. adoce a gosto.



```

=====
Algoritmo é uma "seqüência
finita e lógica de instruções".
=====

```

Concluimos que:

o Todo algoritmo é composto por instruções (comandos) finitas e bem definidas, com a finalidade de solucionar um problema proposto.

A construção de um algoritmo segue os passos abaixo:

i) Interpretar o enunciado do problema, a fim de identificar o **objetivo**;

ii) Identificar os dados a serem fornecidos (**entradas** de dados);

iii) Identificar as **saídas** de dados a serem geradas como resultado da solução;

iv) Determinar o que deve **ser feito** para transformar as entradas nas saídas desejadas. (processamento)

v) Construir o algoritmo, fazendo uso de uma **forma de representação** de algoritmos;

vi) Testar a solução. Execução de todas as ações do algoritmo, conforme o fluxo estabelecido, para **validar sua execução** ou detectar possíveis erros.

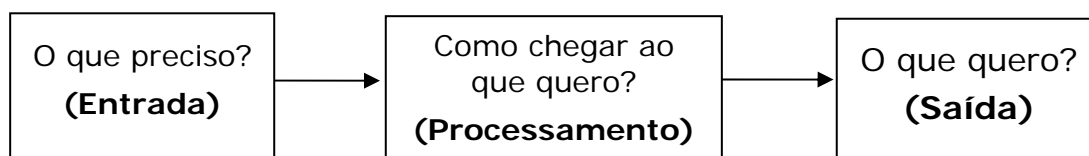


Figura 1. Questões à formalização de um algoritmo.

Desvendando o caminho das pedras ...



Exemplos:

1 - Construir um algoritmo para somar dois números.

Entradas: (o que preciso?)

- primeiro número;
- segundo número.

Processamento: (como chegar ao que quero?)

- somar os números.

Saídas: (o que quero?)

- números somados.

2 - Construir um algoritmo para fazer um suco de laranja.

Entrada:

- laranja;

Processamento:

- cortar a laranja;
- espremer a laranja com espremedor;

Saída:

- suco de laranja.

3 - Construir um algoritmo para calcular a média semestral antes do exame.

Entrada:

- nota 1;
- nota 2.

Processamento:

- somar as notas e dividir por dois;

Saída:

- média semestral antes do exame.



Escreva um algoritmo para descrever como você faz para ir da sua casa até o trabalho/faculdade.

Fluxograma

O fluxograma é utilizado para representar graficamente a **origem dos dados**, o **destino das informações** e os **tipos de armazenamento** dos processos do sistema.

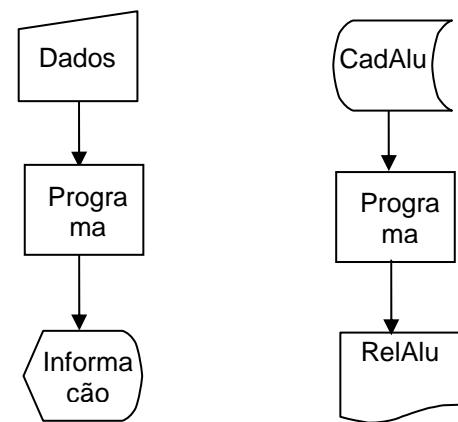
Na intenção de padronizar a utilização dos símbolos, utiliza-se a tabela dos símbolos ANSI³, apresentados abaixo:

SÍMBOLOS ANSI MAIS USADOS	
Processamento	Documento
Fita Magnética	Dados armazenados

Entrada manual	Vídeo
Disco magnético	Linhas de Fluxo

Figura 2. Símbolos Fluxograma

Exemplo:



O primeiro fluxograma representa uma entrada de dados via teclado, o programa em uma visão global e uma saída via monitor de vídeo.

Já o segundo representa uma entrada de dados via disco magnético, o programa uma visão macro e uma saída por impressão de relatório.



Se preocupe com a lógica do programa e não com o símbolo que representa a origem e a saída de dados. Assim, diagramar fica fácil e prático.

³ American National Standards Institute

Diagrama de Chapin

Ainda como forma alternativa de descrição da estrutura dos comandos de um algoritmo, existe o Diagrama de Chapin, "criado por Ned Chapin a partir dos trabalhos de Nassi & Shneiderman".

Uma forma de representação utilizando esta técnica, pode ser assim exemplificada:

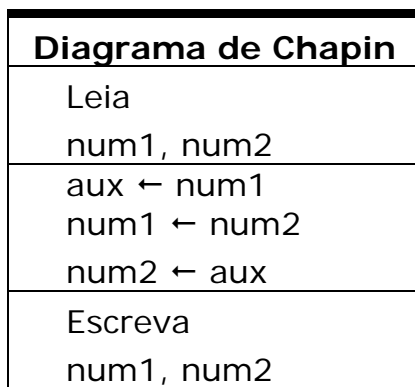


Figura 3. Diagrama de Chapin

Português Estruturado (PORTUGOL)

Considerada uma pseudolinguagem de programação (Português + ALGOL/PASCAL), é também conhecida como Portugol ou Pseudo-Código.

Ou em outras palavras: **"A idéia é permitir que com um conjunto básico de primitivas seja possível ao PROJETISTA pensar na solução do problema e que esta solução seja facilmente implementada no computador"**.

A estrutura utilizada para o Portugol será a seguinte:

```

=====
1  Início algoritmo "NomePrograma";
   variáveis
2
   constantes
3
4  início
5
6  fim.
=====

```

Onde:

1. Denominação do programa;
2. Declaração e classificação de variáveis;
3. Declaração de constantes;
4. Início do bloco principal do programa;
5. Iniciar variáveis de controle o que receberão cálculos, solicitação de entrada de dados, entrada de dados, processamento/cálculos, saída de informações;
6. Final do bloco⁴ principal do programa.

Exemplo:

- Dar entrada a dois números. Trocar os valores das variáveis em que foram armazenados e exibi-los.

```

=====
1  Início algoritmo "TrocaNumero";
   variáveis
   num1, num2, aux: inteiro;
2  início
   leia (num1);
   leia (num2);
   aux ← num1;
   num1 ← num2;
   num2 ← aux;
   escreva num1, num2;
3  fim.
=====

```

⁴ Conjunto de instruções que possui uma função bem definida.

Definindo uma forma de representação

Demonstramos as principais formas de se representar um algoritmo no papel. Existem diversas outras variantes destas técnicas, sendo de escolha do projetista qual a mais adequada à sua situação.

Dentre as apresentadas, a mais difundida a níveis didáticos e computacionais, é o **PORTUGUÊS ESTRUTURADO**. Além de estar presente de maneira sólida nos referenciais bibliográficos, é o método mais próximo de nossa linguagem convencional, facilitando assim a compreensão do algoritmo (do mais simples ao mais complexo).

Otimiza a fase de codificação, onde apenas necessita-se “traduzir” a solução para uma linguagem específica de programação.

1.3 – Processo de Programação

Após a criação da lógica do programa, passamos o programa para uma Linguagem de Programação.

Recomenda-se fortemente que a fase de codificação, ou seja, de transformar a solução construída em linguagem narrativa em uma linguagem de programação, seja realizada somente após do cumprimento de todos os passos de construção de um algoritmo - independente de sua forma de representação.

- Escreva um algoritmo que leia dos números inteiros e efetue a soma.

Início algoritmo “SomaNumeros”;

variáveis

num1: inteiro;

num2: inteiro;

resultado: inteiro;

início

leia num1;

leia num2;

resultado \leftarrow num1 + num2;

escreva resultado;

fim

Fim algoritmo

Tipos e etapas da programação

Considera-se os seguintes tipos de programação:

o **Programação linear**. Os programas são executados linha a linha, seguindo o fluxo do programa. É tido como um modelo tradicional de programação.

o **Programação estruturada**. O programa é dividido em módulos ou sub-rotinas, utiliza-se as estruturas básicas de controle.

o **Programação visual**. O programa é construído baseado em funções que agem sobre um evento externo, iniciado pelo usuário. Como o clique do mouse sobre um botão de comando.

1.4 – Exercícios Propostos



Um bom aprendizado em
ALGORITMOS depende
da prática de exercícios.

1) Defina corretamente os itens abaixo relacionados:

- Lógica de programação
- Programa e programador
- Linguagem narrativa
- Processo de construção de um algoritmo
- Instruções, comandos e blocos
- Formas de representação

2) Descreva como você faz para consultar seu saldo bancário no caixa eletrônico. Identificar separadamente os dados de **entrada**, funções de **processamento** e dados de **saída**.

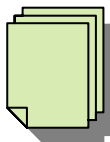
Entrada (O QUE PRECISO?)

Processamento (COMO CHEGAR AO QUE QUERO?)

Saída (O QUE QUERO?)

3) Repita o exercício anterior, sendo que, o objetivo é efetuar um saque bancário no caixa eletrônico.

4) Descreva, em Linguagem Narrativa, as questões 2) e 3). Use como exemplo o problema da "troca de pneu", realizado em sala.



Resolver e entregar na
data definida pelo
professor.

Conceitos Iniciais

2.1 – Introdução

2.2 – Variáveis e Constantes

Variáveis e Memória

Declaração de Constantes

Tipos Básicos de Dados

Regras para criação de identificadores

Palavras reservadas

Comando de atribuição

2.3 – Operadores e Expressões

Operadores: aritméticos, lógicos e relacionais

Prioridades de execução

Expressões Aritméticas

2.4 – Teste de Mesa

“A mente que se abre a uma nova idéia jamais volta ao seu tamanho original.”
(A. Einstein)

2.5 – Algoritmos com Qualidade

2.6 – Exercícios Propostos

2.1 – Introdução

Em toda linguagem de programação as instruções envolvem dois aspectos:

- Sintaxe: forma.
- Semântica: conteúdo.

No PORTUGOL, “a sintaxe é definida, sendo a forma apresentada aceita e respeitada como padrão. Para cada declaração e/ou comando a semântica será devidamente explicada.

2.2 – Variáveis e Constantes

A programação exige o armazenamento de informações a serem utilizadas posteriormente durante a execução do programa. Para isso, faz-se uso da declaração de variáveis e constantes.

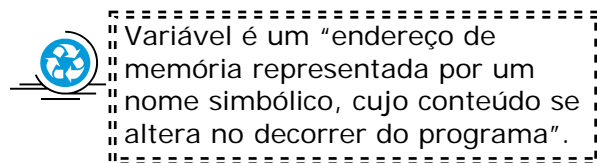
Ou em outras palavras, variável é **“uma porção de memória do computador, onde guardamos um dado ou uma informação. Seu conteúdo pode variar ao longo do tempo durante a execução de um programa, sendo que ela só pode armazenar um valor a cada instante”**.

E, constante é **“um determinado valor fixo que não se modifica com o passar do tempo, durante a execução de um programa”**.

Variáveis e Memória

Imagine que a memória do computador seja um armário com várias gavetas. Cada gaveta possui um rótulo que em determinado momento guarda um conteúdo (informação).

O computador, para poder manipular esses conteúdos, precisa saber qual o local onde o conteúdo se encontra. As linguagens de programação permitem que, em vez de trabalhar com números hexadecimais (endereços físicos que só o computador entende), seja possível, dar nomes para as posições de memória da máquina, permitindo facilitar o manuseio de cada posição de memória. Essa possibilidade permite que o próprio usuário crie o nome das variáveis para as gavetas do nosso armário hipotético. Essas gavetas ora guardam um conteúdo ora outro. Então podemos dizer que os conteúdos das gavetas podem ser alterados no decorrer do programa.



Assim a variável é composta por dois elementos básicos:

- **Conteúdo** - valor atual da variável;
- **Identificador** - nome dado à variável para possibilitar sua manipulação.

Exemplo:

- As informações relativas a um funcionário de uma empresa.

Nome, sexo: caracter;
idade: inteiro;
salario: real;

Os identificadores **Nome**, **Idade**, **Sexo** e **Salário** são as variáveis deste programa. Seus valores podem ser alterados em outro momento da programação.

- Escreva um algoritmo que leia seu nome, idade e altura e os informe.

Inicio algoritmo "LêDados";

variáveis
nome: caractere;
idade: inteiro;
altura: real;
inicio
leia nome;
leia idade;
leia altura;
escreva nome, idade, altura;
fim

Fim algoritmo

Declaração de constantes

Declarar constantes significa reservar uma área da memória RAM, que irá receber um nome (rótulo) cujo conteúdo vai permanecer constante durante toda a execução do programa.

Exemplo:

- A quantidade atual de funcionários de uma empresa é 30.

Quant_Func = 30;

O identificador **Quant_Func**, receberá o valor 30 (total de funcionários), que permanecerá constante durante a execução do programa.

- Escreva um algoritmo que mostre o valor da constante **pi**.

Inicio algoritmo "Pi";

constantes
pi=3,14;
inicio
escreva pi;
fim

Fim algoritmo

Tipos Básicos de Dados

A variável tem que ser definida segundo o conjunto de valores que ela receberá. Os tipos de dados referem-se à classificação dos valores que a variável poderá receber segundo a linguagem de programação que se está utilizando.

Os tipos básicos (ou primitivos) de dados previstos na maioria das linguagens de programação são:

- **Inteiro**: qualquer número inteiro, negativo, nulo ou positivo. Exemplos: 14; 22; 0; -1 e 83.

- **Real:** qualquer número real, negativo, nulo ou positivo. Exemplos: -14.3; 0; 54 e 6.5.
 - **Caracter:** qualquer conjunto de caracteres alfanuméricos/especiais. Representado entre aspas (""). Exemplos: "Informática"; "João"; "Maria".
 - **Lógico:** pode armazenar apenas os valores verdadeiro (V) e falso (F).
- Identifique quais os tipos dos dados abaixo. Utilize **(I)**nteiro, **(R)**eal, **(C)**aractere e **(L)**ógico.
- () 1000
 - () "-900"
 - () VERDADEIRO
 - () -1,56
 - () 34
 - () 10,0
 - () FALSO
 - () "José Francisco"
 - () "Verdadeiro"

Regras para criar identificadores

As variáveis e as constantes têm um nome (rótulo) que as **identifica** dentro do código. Esse nome deve ser criado conforme as seguinte regras:

- o primeiro caractere deve ser uma letra;
- os nomes devem ser formados por caracteres pertencentes às letras do alfabeto e aos números.
- não utilizar espaços em branco entre os caracteres do identificador. Se o identificador for

mais de uma palavra, pode separá-las com o *underline* (_);

- os nomes escolhidos devem ser explicativos do seu conteúdo;
- nomes de variáveis longos dificultarão a codificação. Os nomes não devem ultrapassar o total de caracteres permitido pela linguagem.
- não utiliza-se acentuação ou cedilha;
- os nomes dados às variáveis não podem ser os mesmos nomes das palavras reservadas da linguagem de programação que será utilizada.

Palavras reservadas

São palavras que possuem significado especial na construção do algoritmo ou programa. Tais palavras não podem ser utilizadas como identificadores em um algoritmo.

algoritmo	variaveis	constantes
inicio	fim	se
senão	enquanto	para
faça	repita	ate
escolha	caso	mod
leia	escreva	div
E	OU	NÃO

Tabela 1. Palavras reservadas

- Assinale com um X os nomes válidos para uma variável.
- () ENDEREÇO
- () 21BRASIL
- () CIDADE3
- () NOME_USUARIO
- () NOME*USUARIO

- Escreva um algoritmo para calcular a área de um quadrado.

Início algoritmo "AreaQuadrado";

variáveis

lado, area: inteiro;

início

leia lado;

area \leftarrow lado * lado;

escreva area;

fim

Fim algoritmo

"Quanto mais tempo se leva na construção de um algoritmo no papel, menos tempo leva-se codificando-o."



Comando de atribuição

É o comando que indica o que a variável vai receber em seu conteúdo em determinado momento. Nas linguagens de programação são representados pelo sinal de igualdade (=) ou dois-pontos e igualdade (:=), dependendo da linguagem a ser utilizada.

Sintaxe:

Identificador \leftarrow valor;

Exemplo:

Media \leftarrow 0;

Lê-se: Media recebe zero.



Devemos iniciar, com valor nulo, as variáveis que receberão cálculos no decorrer do algoritmo.

2.3 – Operadores e Expressões

Para a realização de cálculos, comparações e expressões, utilizamos os operadores aritméticos, lógicos e relacionais.

Operadores: aritméticos, lógicos e relacionais

Operadores Aritméticos	
Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
**	Exponenciação
SQR	Raiz Quadrada
MOD	Resto da divisão inteira
DIV	Quociente divisão inteira
Utilizados para cálculos matemáticos	

Exemplos:

a) $X \leftarrow A**B-B**C$;

b) $D \leftarrow A*(B+C)$;

c) $Y \leftarrow \text{SQR } A$;


d) $Z \leftarrow Q-P*C$

e) $N \text{ MOD } 2 \leftarrow 0$;

f) 3 MOD 2 (resto de divisão de três por dois)

g) 3 DIV 2 (quociente de divisão de três por dois)

h) SQR 8 (raiz quadrada de oito)



Dividendo \leftarrow 3	2 \rightarrow Divisor
MOD \leftarrow 1	1 \rightarrow DIV

Operadores Relacionais	
Operador	Descrição
=	Igual
< >	Diferente
> =	Maior ou igual a
< =	Menor ou igual a
>	Maior
<	Menor
<i>Utilizados para expressões</i>	

Exemplos:

- a) media >= 7,0;
- b) matricula < > 0;
- c) salario > 3.000,00;
- d) nome ← "João Silva";
- e) contador <= 30;

Operadores Lógicos	
Operador	Descrição
E	Conjunção
OU	Disjunção
NÃO	Negação
<i>Utilizados para condições</i>	

Exemplo:

- a) (n1>n2) E (n1>n3);
- b) (n1>n2) OU (n1>n3);
- c) (n1>n2) NÃO (n1>n3);

Tabela de Decisão

A tabela de decisão, ou Tabela Verdade, serve para orientar nossas escolhas quando nos depararmos com as situações de condições no algoritmo.

Operador lógico E

Condição1		Condição2	Resultado
V	<u>E</u>	V	V
V	<u>E</u>	F	F
F	<u>E</u>	V	F
F	<u>E</u>	F	F

Operador lógico OU

Condição1		Condição2	Resultado
V	<u>OU</u>	V	V
V	<u>OU</u>	F	V
F	<u>OU</u>	V	V
F	<u>OU</u>	F	F

Operador lógico NÃO

Condição		Resultado
VERDADEIRO	<u>NÃO</u>	FALSO
FALSO	<u>NÃO</u>	VERDADEIRO

Prioridades de execução

Quando houver situação com operações mistas, as prioridades para execução devem ser as mesmas regras que as adotadas na matemática:

- 1) Efetuar operações embutidas em parênteses mais internos;
- 2) Efetuar exponenciação e funções;
- 3) Efetuar multiplicação e divisão (*, /);
- 4) Efetuar adição e subtração (+, -);
- 5) Efetuar operações relacionais (>, <, < >, =, <=, >=);
- 6) Efetuar operações lógicas (NÃO, E, OU).

Expressões Aritméticas

Consiste em um conjunto de variáveis e/ou valores, separados por caracteres especiais, que indicam as operações que devem ser executadas.

Exemplos:

Expressão	Comentário
$(3 + 5) * (4 * (10 - 7)) / 2$	Opera-se o que estiver nos parênteses mais internos
$(3 + 5) * (4 * 3) / 2$	Operam-se os dois parênteses que restaram
$8 * 12 / 2$	Só há * e / - , então opera-se da esquerda para a direita
$96 / 2$	
48	Resultado final

Expressão	Comentário
$6 * (9 + 3 * 2) / (2 * 4 - 11)$	Operam-se as multiplicações nos parênteses
$6 * (9 + 6) / (8 - 11)$	Operam-se os parênteses
$6 * 15 / -3$	So há * e / - opera-se da esquerda para a direita
$90 / -3$	Observe o sinal negativo antes do 3
-30	Resultado final

Até agora só vimos expressões com constantes. Na maioria das vezes, porém, haverá expressões que combinarão variáveis e constantes. Quando houver uma variável em uma expressão, o cálculo deve ser feito usando-se o valor da variável naquele momento.

Exemplos:

Expressão	Comentário
A: inteiro;	Declaração da variável
A ← 3;	Atribuição de valor
escreva A * 5;	Será exibido o valor 15, pois A=3, logo $3 * 5 = 15$

Expressão	Comentário
A, B : inteiro;	Declaração da variável
A ← 10;	Atribuição de valor
B ← A * 3;	O valor de B é 30 ($10 * 3$)
escreva B - A + 1;	Será exibido o valor 21 ($30 - 10 + 1$)
B ← 40;	Novo valor para B
escreva B - A + 1;	Agora será exibido o valor 31 ($40 - 10 + 1$), pois o valor de B mudou

Expressão	Comentário
A: <u>inteiro</u> ;	Declaração da variável
A \leftarrow 3 * 2;	O valor de A é 6
A \leftarrow A + 1;	Agora o valor de A passa a ser 7 (6, o valor anterior, mais 1)
<u>escreva</u> A * A;	Será exibido o valor 49 (7 * 7)

Expressão	Comentário
X: <u>inteiro</u> ;	Declaração da variável
<u>escreva</u> "Digite um número :";	
<u>leia</u> X;	Obtenha um número qualquer do usuário
Escreva X * 2;	Será exibido o dobro do número digitado (qualquer que ele seja)

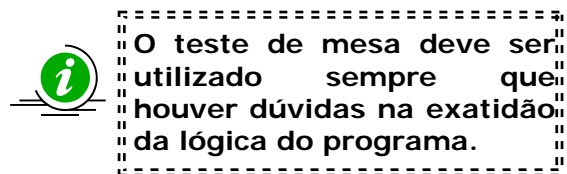
2.4 - Teste de Mesa

É uma técnica utilizada para simulação da execução do algoritmo que construímos, verificando se sua lógica está correta ou não.

Simulamos a solução, passando por todas as partes do algoritmo para testar a resolução e encontrar erros se houver.

Exemplo:

- Para calcular a média entre duas notas, poderíamos testar nosso programa com os dados da tabela "Antes do Processamento". Tais dados devem seguir o fluxo do programa e o resultado final deve ser o da tabela "Depois do Processamento".



Antes do Processamento

Nome	Nota_1	Nota_2
João da Silva	8,0	6,5
Afonso Direito	10,0	10,0
Luana Pontes	4,0	7,0

Depois do Processamento

João da Silva	7,25
Afonso Direito	10,0
Luana Antunes	5,5

2.5 - Algoritmos com Qualidade

1) Algoritmos devem ser feitos para serem lidos por seres humanos. Tenha em mente que seus algoritmos deverão ser lidos e entendidos por outras pessoas (e por você mesmo) de tal forma que possam ser corrigidos, receber manutenção e ser modificados.

2) Escreva os comentários no momento em que estiver escrevendo o algoritmo. Um algoritmo não documentado é um dos piores erros que um programador pode cometer e é sinal de amadorismo. Como o objetivo de se escrever comentários é facilitar o entendimento do algoritmo, eles devem ser tão bem concebidos quanto o próprio algoritmo. E a melhor maneira de se conseguir isso é escrevê-los nos momentos de maior intimidade com os detalhes, ou seja, durante a resolução do problema.

3) Os comentários deverão acrescentar alguma coisa; não apenas para frasear os comandos. O conjunto de comandos nos diz o que está sendo feito; os comentários deverão nos dizer por quê.

4) Use comentários no prólogo. Todo algoritmo ou procedimento deverá ter comentários no seu prólogo para explicar o que ele faz e fornecer instruções para seu uso. Alguns destes comentários seriam

- uma descrição do que faz o algoritmo
- como utilizá-lo
- explicação do significado das variáveis mais importantes
- estruturas de dados utilizadas
- autor, data de escrita, etc.

5) utilize espaços em branco para melhor legibilidade. Espaços em branco, inclusive linhas em branco, são valiosas para melhorar a aparência de um algoritmo.

- deixe uma linha em branco entre as declarações e o corpo do algoritmo.

- deixar uma linha em branco antes e outra depois de um comentário

- separar grupos de comandos que executam funções lógicas distintas por uma ou mais linhas em branco.

- utilizar brancos para indicar precedência de operadores. Ao invés de "A+B *C" é mais amigável a forma "A+ B*C".

6) Escolha nomes representativos para suas variáveis. Os nomes das variáveis deverão identificar, o melhor possível, as quantidades que elas representam. Por exemplo, $X \leftarrow Y + Z$ é muito menos claro que $\text{PRECO} \leftarrow \text{CUSTO} + \text{LUCRO}$. Este é o princípio mais importante da legibilidade de algoritmos.

7) um comando por linha é suficiente. A utilização de vários comandos por linha é prejudicial por várias razões, dentre elas:

2.6 – Exercícios Propostos



EXERCÍCIOS PROPOSTOS

Instruções:

- Formar grupos de até 4 alunos;
- Utilizar até 3 páginas;
- Exagere e cite as fontes de pesquisa.

1) Efetuar pesquisa bibliográfica sobre:

- Operadores: aritméticos, lógicos e relacionais;
- Expressões: hierarquia das operações;
- Comandos de atribuição, entrada e saída.

2) Escreva um algoritmo que leia seu nome, idade e altura e os escreva na tela.

3) Escreva um algoritmo para calcular a área de um círculo.

$$\text{Área do círculo} = 3,1415 * \text{raio}^2$$

4) Desenvolva os algoritmos utilizando a técnica do português estruturado para os seguintes programas:

a) Ler uma temperatura em graus Celsius e apresentá-la convertida em graus Fahrenheit. A fórmula da conversão é $F \leftarrow (9 * C + 160) / 5$, sendo F a temperatura em Fahrenheit e C a temperatura em Celsius.

b) Ler uma temperatura em graus Fahrenheit e apresentá-la convertida em graus Celsius. A fórmula de conversão é $C \leftarrow (F - 32) * (5/9)$, sendo F a temperatura em Fahrenheit e C a temperatura em Celsius.

c) Ler dois inteiros (variáveis A e B) e imprimir o resultado do quadrado da diferença do primeiro valor pelo segundo.

d) Calcular e apresentar o valor do volume de uma lata de óleo, utilizando a fórmula: $\text{VOLUME} \leftarrow 3.14159 * \text{RAIO}^2 * \text{ALTURA}$