ALGORITMOS

I

1. INTRODUÇÃO AO ALGORITMO

1) CONCEITOS BÁSICOS

a) Computadores fazem aquilo que mandamos e não o que queremos
b) Não deve haver nenhuma ambigüidade ou possibilidade de interpretações alternativas

2) DIVIDIR PARA CONQUISTAR

a) Programação não é exatamente uma tarefa fácil
b) Pode ser simplificada dividindo-se sucessivamente o problema em problemas menores
c) Duas grandes divisões na elaboração de uma solução em programação
    i) Fase de resolução de problemas (elaboração do algoritmo da solução)
    ii) Fase de implementação da solução (em uma linguagem de programação)
        (1) Se o algoritmo é preciso, a implementação é direta

3) ALGORITMO

a) Sequência ordenada, e sem ambigüidade, de passos que levam à solução de um dado problema
b) Exemplos comuns:
    i) Receita de cozinha, indicações sobre como chegar a um determinado endereço, manual de montagem de bricolagem, kits de aeromodelos, etc.
c) Características essenciais:
    i) Devem ser simples
    ii) Não devem ter ambigüidades
    iii) Devem estar numa ordem cuidadosamente definida
    iv) Devem resolver o problema em um número finito de passos

4) EXEMPLO INICIAL

a) Algoritmo para substituir uma lâmpada queimada: inicialmente dois passos
    i) Remova a Lâmpada queimada
    ii) Coloque a nova lâmpada
b) Esses dois passos pressupõem conhecimento de muitos passos intermediários
    i) Deve-se ser preciso e detalhado o suficiente para quem for executar a operação consiga a executar apenas com as instruções fornecidas
c) Detalhamento do passo 4.1.i. do algoritmo acima:
    i) Posicione a escada debaixo da lâmpada queimada
    ii) Suba na escada até que a lâmpada possa se alcançada
    iii) Gire a lâmpada queimada no sentido anti-horário até que se solte
    iv) Desça a escada
d) Detalhamento do passo 4.1.ii do algoritmo acima
    i) Escolha uma nova lâmpada de mesma potência da queimada
    ii) Suba a escada
    iii) Posicione a nova lâmpada no soquete
    iv) Gire-a no sentido horário até que ela ser firme
    v) Desça a escada
    vi) Acenda a luz para testar a lâmpada
    vii) Se não acender, repita desde o passo 4.3.ii
        (1) O algoritmo seria mais eficiente se o passo 4.3.ii fosse executado antes do passo 4.4.i, isto é, subir a escada já com a nova lâmpada em mãos
e) O passo 4.4.i pode ser detalhado da seguinte maneira
    i) Selecione uma lâmpada

ii) Verifique a potência da Lâmpada
(1) Se a potência não é a mesma, repita 4.5.i até encontrar uma que sirva
(a) descarte a lâmpada
(b) selecione uma nova lâmpada

5) CONCEITOS IMPORTANTES:

a) Decisão: Se (a condição for satisfeita), Então (faça isso), Senão (faça aquilo)
b) Repetição condicional: Repita até que (a condição seja satisfeita, então continue)
c) Repetição simples (Repita um determinado número de vezes, então continue)
d) O algoritmo para substituição de lâmpadas passa a ser:

```
1. Posicione a escada debaixo da lâmpada queimada
2. Selecione uma nova lâmpada para substituição
        Repita até encontrar uma com potência adequada (E funcionando)
                Verifique a potência da lâmpada (E se está funcionando)
                Se a potência não for a mesma (OU não estiver funcionando)
                        Descarte a lâmpada selecionada
                        Selecione uma lâmpada
3. Repita até que a lâmpada possa ser alcançada
        Suba um degrau da escada
4. Repita até que a lâmpada saia do soquete
        Gire a lâmpada no sentido anti-horário
5. Posicione a nova lâmpada no soquete
6. Repita até que a lâmpada saia do soquete
        Gire a lâmpada no sentido horário
7. Repita até alcançar o chão
        Desça um degrau
8. Guarde a escada
```

```
SE (condição)
        ENTÃO (ação 1)
        SENÃO (ação 2)
FIM SE
```

6) DETALHAMENTO DO ALGORITMO

a) Pode continuar quase que indefinidamente
b) As operações indicadas devem ser simples e sem ambigüidade
c) As decisões necessárias devem ser imediatas
d) A ordem na qual os passos devem ser seguidos deve ser claramente expressa
e) O detalhamento deve prosseguir até chegar ao nível de entendimento do "Executor"
f) As linguagens de programação têm um número limitado de instruções
i) Os algoritmos devem ser expressos nos termos dessas instruções (comandos)

7) Exercícios: Utilize apenas a estrutura "Repita até"

a) Elabore um algoritmo representando sua saída de casa pela manhã — exemplo
i) Inicie pelo passo "dormindo na cama". Inclua as atividades matinais normais
b) Elabore um algoritmo para trocar um pneu furado.
i) Admita que estão disponíveis um macaco, chave de roda, e pneu "bom". O macaco está abaixado, o carro está parado e o triângulo já foi colocado
c) Elabore um algoritmo para fazer pipoca numa panela de fogão
i) Admita que estão disponíveis todos os ingredientes, fogão a gás e fósforo
d) Desenvolva um algoritmo para substituir um vidro quebrado de uma janela
i) Em uma lista à parte, descreva os materiais utilizados
e) Elabore um algoritmo para ir de ônibus de um ponto a outro dentro da cidade
i) Admita que está em um ponto, que paga em dinheiro e deve receber troco
f) Elabore um algoritmo para realizar uma chamada de telefone fixo
i) Admita que é um telefone fixo, funcionando perfeitamente; você está próximo dele, e não repetirá a ligação se o telefone estiver ocupado ou não atender

desafio

2. DADOS, TIPOS DE DADOS E OPERAÇÕES PRIMITIVAS

1) TIPOS DE DADOS

a) Computadores lidam com diversos tipos de dados
   i) Cada tipo de dado é representado de forma diferente no computador
   ii) Cada tipo de dado requer um diferente tipo de instrução para processamento

2) NÚMEROS INTEIROS (ou números de ponto fixo)

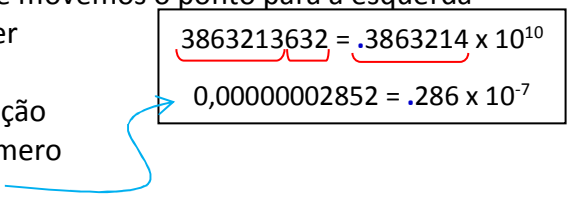
a) Não tem componentes decimais nem ponto ou vírgula
a) Podem ser negativos ou positivos (ex: -7; 485; -1815)

3) NÚMEROS REAIS

a) Tem componente decimal e utiliza o ponto* como separador decimal (*formato EUA)
b) Frações são representadas por aproximações decimais
c) Podem ser negativos ou positivos (ex: 2.34; -1.582; 0.034; 125.2)

4) NOTAÇÃO CIENTÍFICA (ou de Ponto Flutuante)

a) Computadores trabalham com número limitado de dígitos, em geral 6 a 7
   i) Números de precisão dupla operam com 15 ou 16 dígitos
   ii) O número de dígitos disponíveis varia de máquina para máquina
a) Solução: representação do número em potências de 10:
   i) 3863213632 é representado como 3.863×10^9
      (1) Obs: o primeiro é inteiro e o segundo é real
   i) O número de dígitos conservados depende da exatidão requerida no cálculo
a) Procedimento de conversão de número inteiro ou real para notação científica:
   i) Adicione 1 à potência de 10 a cada casa que movemos o ponto para a esquerda
   ii) Decida o número de dígitos que deseja reter
   iii) Arredonde o último dígito
a) O ponto decimal ocupa sempre a primeira posição
   i) Por isso ele não é armazenado junto ao número
a) Nomenclatura: Fração (.286) e expoente (-7)



5) CADEIA DE CARACTERES

a) É uma sequência de itens conhecidos pelo computadores
a) Exemplo: A, B, C, a, b, c, 1, 2, 3, @, #, " ",), *, etc.
   i) Obs: os números armazenados como caracteres se comportam como "letras"
a) Em geral são armazenados em computadores entre apóstrofos (aspas simples)
   i) São os chamados "delimitadores de caracteres"
   ii) A cadeia de caracteres pode ter qualquer tamanho, mas na prática, pode haver um limite, por exemplo 255 caracteres incluindo os espaços
   iii) Exemplo: 'Estado de São Paulo', '23/04/2011', '34 + 12 = 46'
a) Quando é necessário incluir o apóstrofo na cadeia, inclui-se mais um apóstrofo antes:
   i) Joana D´Ark é representado como 'Joana D''Ark'
a) O computador armazena duas informações:
   i) A cadeia propriamente dita, excluindo os apóstrofos e
   ii) O número de caracteres da cadeia

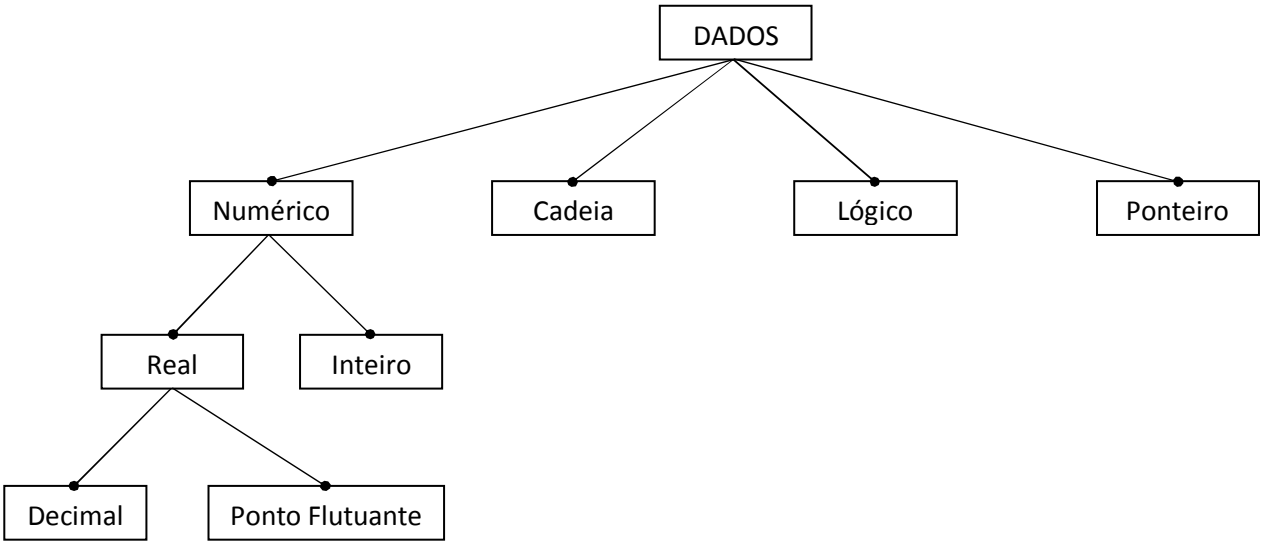
6) DADOS LÓGICOS

a) Só tem dois valores: verdadeiro ou falso

7) PONTEIRO

a) Armazena um determinado Endereço de Memória

8) CLASSIFICAÇÃO DOS TIPOS DE DADOS PRIMITIVOS



9) OPERAÇÕES COM OS ELEMENTOS DE DADOS NUMÉRICOS

a) Quatro operações básicas com dados numéricos:
   i) Soma, Subtração, Multiplicação, Divisão
      (1) Operações adicionais: Exponenciação e Radiciação
   ii) Operações com dados não numéricos serão vistas posteriormente

a) Representação das operações
   i) Soma e subtração utilizam a representação usual: " + " e " – "
   ii) Multiplicação utiliza o asterisco " * " em lugar do símbolo usual " x "
   iii) Divisão utiliza a barra " / " em lugar do símbolo usual " ÷ "
   iv) Exponenciação não tem representação usual, utiliza-se o " ^" ou "↑"
   v) Radiciação utiliza-se a representação da exponenciação " ^ " ou "↑" com valores <1

a) Resultados de operações com os diferentes tipos de dados numéricos
   i) Os diferentes tipos de dados numéricos são escritos por nós e armazenados pelo computador em diferentes formas
   iii) Números Reais
      (1) Soma, subtração, multiplicação e divisão: resultam em número real
      (2) Exponenciação e radiciação: resultam em número real
   iv) Números Inteiros:
      (1) Soma, subtração e multiplicação: resulta em inteiro
      (2) Divisão: resulta em número inteiro: 10 / 4 = 2
         (a) atenção à transformação: 10*1.0 / 4 = 2.5
      (3) Exponenciação: resulta em número inteiro
      (4) Radiciação: resulta em número real (atenção à transformação)
   v) Operações com diferentes tipos de dados (inteiro com real):
      (1) Resultam sempre em número real

10) EXERCÍCIOS

a) Dar o tipo de cada uma das seguintes constantes

a) 613		e) -3.012×10^{15}	
b) 613.0		f) 17×10^{12}	
c) -613		g) -28.3×10^{-33}	
d) '613'		h) 'fim de questão'	

1.1. Dar o resultado e o tipo de cada uma das seguintes expressões

a) 5^2 + 3		
b) 6 + 19 – 0.3		
c) $3.0^{5.0}$ +1		
d) 1 / 4 + 2		
e) 29.0 / 7 + 4		
f) 3 / 6.0 - 7		

1.2. Expressar como um número real na forma de ponto flutuante, com seis dígitos significativos, os seguintes valores:

a) π	3.1415926535897932	
b) e	2.7182818284590452	
c) Constante de Avogadro	$6{,}02214179 \times 10^{23}$	
d) Massa elétron (kg)	$9{,}1093897 \times 10^{-31}$	
e) diâmetro do átomo (cm)	$\sim 10^{-8}$	
f) valor de parsec (km)	30856775800000	

3. VARIÁVEIS E EXPRESSÕES

1) MANIPULAÇÃO DE DADOS

a) Através de expressões mais elaboradas das operações fundamentais (+, -, *, /, ↑)
b) Variável: entidade que possui um valor e tem um nome
   i) O valor da variável pode "variar" durante a execução do programa
      (1) Exemplo: teorema de Pitágoras: $a^2 = b^2 + c^2$ onde a, b e c são variáveis
c) Expressões: fórmulas definidas por relações expressas, aplicadas a cálculos específicos
   i) Permitem determinar valor de uma variável
      (1) Exemplo: $a = \sqrt{b^2 + c^2}$
d) Variáveis permitem especificação de fórmulas gerais de cálculo
      (1) Variáveis têm nomes e podem receber valores diferentes no decorrer do cálculo
      (2) Em um determinado instante só pode ter um único valor

Caso/Variável	a	b	c
a	5	4	3
b	13	5	12

e) Usualmente utilizam-se nomes mais significativos para variáveis, como:
   i) LADO1, LADO2, HIPOTENUSA - isso torna o programa mais legível e "depurável"
f) Regras para nomear variáveis (variam um pouco entre as linguagens de programação)
   i) Devem iniciar sempre por uma letra
   ii) Podem conter letras, números e alguns caracteres especiais
   iii) Não podem conter brancos (espaços) – utiliza-se eventualmente o "underline: "_ "
g) Exemplo de nomes válidos de variáveis:
   i) TOTAL, AREA_TRIANGULO, UM_NOME_MUITO_EXTENSO,
h) Exemplo de nomes inválidos de variáveis
   i) 2ANDAR (não começa com letra), SEGUNDO ANDAR (inclui espaço), X+Y (inclui +)
i) Exemplo de expressão utilizando variáveis
   i) AREA_TRIANGULO = BASE * ALTURA / 2
   ii) Se, em diversos momentos, BASE e ALTURA tiverem valores diferentes, AREA_TRIANGULO terá resultados diferentes nesses momentos

2) TIPOS DE VARIÁVEIS

a) São equivalentes aos tipos de constantes: Inteiro, Real, Cadeia (texto), Lógico, Ponteiro
b) Uma variável deve ser de um único tipo
   i) Esse tipo é atribuído no início do programa ou dinamicamente no decorrer deste
      (1) Após definição do tipo, deve-se manter esse tipo no programa todo
   ii) Se um valor de outro tipo for atribuído a essa variável, ocorrerá um erro (tipos incompatíveis) ou conversão de tipos (tipos compatíveis) – ver item 4.

3) OPERAÇÃO DE ATRIBUIÇÃO

a) Forma usual para atribuir um valor a uma variável – será utilizado o símbolo "←"
   i) Exemplos: LADOA ← 16; LADOB ← 13; SALDO ← -320.40
   ii) A variável mantém o valor atribuído até eventual nova atribuição
   iii) A atribuição é uma operação "destrutiva" pois o valor anterior é perdido
      (1) O valor é armazenado em uma posição da memória (palavra)

4) CONVERSÕES DE TIPO

a) Uma variável de um tipo só armazena valores daquele tipo;
b) Se um valor de outro tipo for atribuído a ela ocorrerá:
   i) Um erro, se os tipos forem incompatíveis (texto x número x ponteiro x lógico);
   ii) Uma conversão de tipo do valor, se os tipos forem numéricos (Real x Inteiro);
c) Exemplos de operações de atribuição:
   i) VARIAVEL_INTEIRA ← VALOR_REAL : o valor real será convertido para Inteiro;
   ii) VARIAVEL_REAL ← VALOR_INTEIRO : o valor inteiro será convertido para Real;
   iii) Outras atribuições resultarão em erro
      (1) VARIAVEL_INTEIRA ← VALOR CADEIA : ocorrerá erro de incompatibilidade;
      (2) VARIAVEL_LOGICA ← VALOR_REAL : ocorrerá erro de incompatibilidade;
      (3) VARIAVEL_CADEIA ← VALOR INTEIRO : ocorrerá erro de incompatibilidade;
      (4) Etc.
   iv) Obs: conversão de Inteiro para Real não altera o valor calculado,
      mas conversão de Real para Inteiro altera o valor (arredonda ou trunca);
      Números reais ocupam mais memória

5) EXPRESSÕES

a) Operação de Atribuição (←) pode ter qualquer expressão do lado direito;
b) Expressão é uma combinação de variáveis, constantes e operadores;
   i) O resultado da avaliação da expressão é atribuído à variável indicada;
      (1) Variável ← Expressão
      (2) INT ← 3 + 16 + 8 (onde INT é uma variável tipo Inteiro);
   ii) A Expressão pode também conter variáveis, com valores definidos préviamente:
      (1) TERMO1 ← 13.6 + 7.4 (21.0)
      TERMO2 ← 0.7 * 28.6 (20.02)
      RESULTADO ← TERMO1/TERMO2 (1.048951)
      (2) As variáveis TERMO1 e TERMO2 mantém seus valores após o processo;
      (3) Todos os termos do lado direito da expressão devem ter seus valores definidos previamente antes da operação de atribuição final (RESULTADO←)

6) MODIFICANDO UM VALOR ARMAZENADO

a) O valor de uma variável pode ser alterado na execução do programa:
   i) X ← 0 ... A ← 0 ... X ← A + 1 (resulta em X=1)
b) Uma variável pode ter seu valor alterado na própria expressão:
   i) X ← 0 ... X ← X + 1
c) Muitas linguagens utilizam como símbolo de atribuição o sinal "="
   i) X = X + 1 (o que não tem sentido do ponto de vista matemático)

7) PRIORIDADE DOS OPERADORES

a) É a prioridade convencional da matemática
   i) Parenteses: ( )
   ii) Atribuição de Sinal: + ou -
   iii) Exponenciação e Radiciação: ^ ou ↑
   iv) Multiplicação e Divisão: * ou /
   v) Soma e Subtração: + ou -

Os Parênteses agrupam blocos que devem ser executados em primeiro lugar.

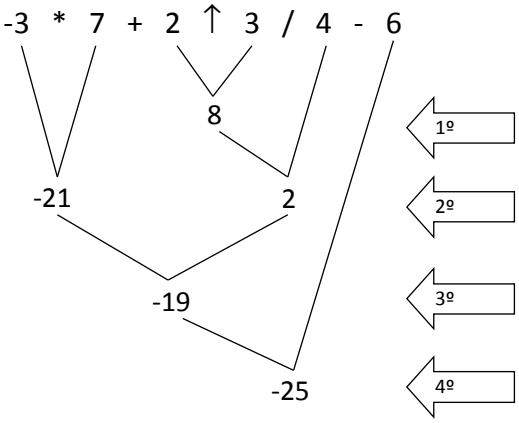
Exponenciação e atribuição de sinal são aplicados da direita para a esquerda.

Operadores de mesma prioridade são executados na ordem em que aparecem, da esquerda para a direita.

8) AVALIAÇÃO DE EXPRESSÕES

a) É a aplicação direta das regras de prioridade

b) Exemplo: -3*7+2↑3/4-6:



c) Expressões com significados alterados por parênteses:

i) 12 - 2 / 2 + 3 e (12 - 2) / (2 + 3)

d) Em expressões mais complexas é recomendável o uso de parênteses, mesmo quando desnecessários do ponto de vista exclusivo do cálculo, para facilitar a leitura da expressão (e simplificar a depuração do programa);

9) FUNÇÕES EMBUTIDAS

a) Complementam as operações convencionais da matemática, simplificando o cálculo;

b) Algumas funções embutidas úteis:

i) ABS(*e*) .....: valor absoluto de uma expressão real ou inteira – $|e|$

ii) SQRT(*e*) .....: raiz quadrada de um valor maior que zero

iii) TRUNC(*e*) ...: valor truncado $|T| \leq |e|$

iv) ROUND(*e*) ..: valor arredondado $|T| \leq |e + 0{,}5|$

v) LOG(*e*).....: logarítmo na base e

vi) LOG10(*e*)....: logarítmo na base 10

vii) EXP(*e*).....: exponencial de *e*

viii) SIN(*e*) .....: seno de *e*

ix) COS(*e*).....: coseno de *e*

x) TAN(*e*).....: tangente de *e*

10) ENTRADA E SAÍDA

a) Depende em alto grau da linguagem de programação e do sistema de computador;

b) Aqui serão utilizados abstrações desses comandos:

i) *Leia*: para leitura de valores em algum dispositivo de entrada (e atribuição a variáveis);

ii) *Escreva*: para mostrar os resultados em algum dispositivo de saída (tela, impressora);

c) Comando **Leia**: Forma do comando: Leia(*lista de entrada*):

i) A lista de entrada definine as variáveis às quais os valores devem ser atribuídos, na mesma ordem em que são encontrados no fluxo de entrada de dados:

(1) Leia(A, B, C): o primeiro valor é atribuído à variável A, o segundo à B, etc.

d) Exemplo: fluxo de entrada com os seguintes valores: -16, 3, 7, 21, 16, 0, 4, 8, 1.

i) Leia(A, B, C) A←-16, B←3, C←7

ii) Leia(D, E, F, G) D←21, E←16, F←0, G←4

iii) Leia(X, Y) X←8, Y←1

e) Importante: É uma operação destrutiva (destrói o valor anterior da variável) e é necessário manter a compatibilidade de tipos (variáveis declaradas);

f) Comando **Escreva**: Forma do comando: Escreva(*lista de saída*):
   i) A lista de saída pode ser qualquer variável, expressão ou constante;

g) Exemplo:
```
NOTA1←73.0  NOTA2←65.0  NOTA3←94.0  NOTA4←87.0
MEDIA←(NOTA1+NOTA2+NOTA3+NOTA4)/4.0
Escreva(MEDIA)
        o resultado seria: 79.75
Escreva(NOTA1, NOTA2, NOTA3, NOTA4, MEDIA)
        o resultado seria: 73.0  65.0  94.0  87.0  79.75
Escreva('Notas Individuais: ', NOTA1, NOTA2, NOTA3, NOTA4)
Escreva('Média Final: 'MEDIA)
        o resultado seria:  Notas individuais: 73.0  65.0  94.0  87.0
                            Média Final: 79.75
```

h) Pode-se imprimir também constantes e variáveis tipo cadeia:
   i)
```
NOME_DADO←'DONALDO'
SOBRENOME←'PATO'
Escreva(SOBRENOME, ',', NOME_DADO), 3)
        o resultado seria: PATO, DONALDO 3
```

i) Em muitas linguagens é possível incluir uma expressão como parte da lista de saída:
   i)
```
Escreva('NOTAS INDIVIDUAIS:', NOTA1, NOTA2, NOTA3, NOTA4)
Escreva('MÉDIA FINAL:', (NOTA1+NOTA2+NOTA3+NOTA4)/4.0)
```

11) EXERCÍCIOS

a) Dê o valor da variável RESULTADO após a execução da seguinte sequência de operações

i)	RESULTADO←3.0 * 6		obs: todas as variáveis são reais
ii)	X←2.0	Y←3.0	RESULTADO←X↑Y – X
iii)	RESULTADO←4	X←2	RESULTADO←RESULTADO*X

b) Dê o valor de cada uma das expressões inteiras seguintes:

i) 16 * 6 – 3 * 2		iv) 3 + 2 * (18 – 4↑2)	
ii) -2↑3		v) 2↑2 * 3	
iii) (28 + 3 * 4) / 4		vi) 8 – 30 / 6	

c) Suponha que A, B e C sejam variáveis reais e que I, J e K sejam variáveis inteiras.
Dados: A = 4.0, B = 6.0 e I = 3, qual seria o valor final dos comandos seguintes:

i) C←A * B - I	C=	iv) K←TRUNC(B/A+4.7)	K=
ii) K← I / 4 * 6	K=	v) J←ROUND(A/ (5/I))	J=
iii) C←B / A + 1.5	C=	vi) K←ABS(A-B) * 2 + 1	K=

d) Escreva as seguintes expressões matemáticas como expressões de computador:

i) $\frac{a}{b}+1$		v) $(a+\ b)\frac{c}{d}$	
ii) $\frac{a+b}{c-d}$		vi) $[(a+b)^c]^d$	
iii) $\frac{a+\frac{b}{c}}{d-\frac{e}{f}}$		vii) $\frac{sen\ a+\cos a}{tg\ a}$	
iv) $a+\frac{b}{c-d}$		viii) $\frac{-b+\sqrt{b^2-4ac}}{2a}$	

4. DESCRIÇÃO DE ALGORITMOS

1) ALGORITMO:

a) Primeiro passo para a preparação de um programa de computador;
b) É uma seqüência de passos em uma ordem cuidadosamente definida;
c) Passos devem ser expressos claramente e sem ambigüidade;
d) Auxilia a documentação posterior do programa;

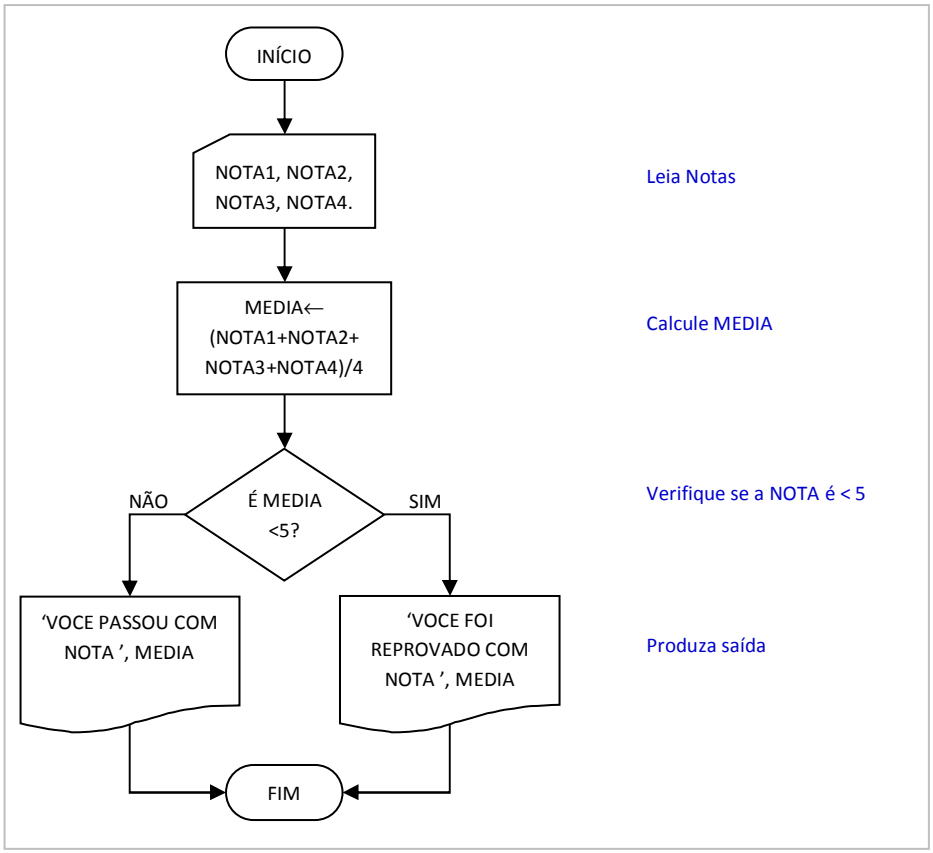
2) DESCRIÇÃO NARRATIVA:

a) Método direto de expressar um algoritmo (p/ex: receita de bolo);
b) Linguagem natural prolixa, imprecisa e pouco confiável;
c) Risco de interpretação incorreta e/ou perda de informação;
d) Não é o meio mais adequado para expressar algoritmos de programação;
e) É utilizada no estágio de documentação do programa.

3) FLUXOGRAMA

a) Símbolos usuais:

b) Exemplo de fluxograma simples:



c) Recurso visual para transmitir idéias:
   i) Mostra a lógica de um algoritmo;
   ii) Enfatiza os passos individuais e suas interconexões.
d) Em programação, pode mostrar a LÓGICA, mas obscurece a ESTRUTURA;
   i) Utiliza-se, então, o Algoritmo e, onde necessário esclarecer a lógica, o Fluxograma;

4) LINGUAGEM ALGORÍTMICA:

a) Linguagem simplificada de programação que inclui:
   i) A descrição da prosa e
   ii) A concisão do fluxograma.

b) O resultado é similar a muitas linguagens de programação:
   i) A passagem do Algoritmo para uma linguagem específica é operação quase direta;

c) A escolha da linguagem de programação a ser utilizada depende principalmente da(s):
   i) Natureza da aplicação em particular;
   ii) Características desejadas na linguagem;
   iii) Disponibilidade da linguagem no sistema de computadores a ser utilizado.

d) Exemplo: Dados o comprimento dos lados de um triângulo retângulo (variáveis LADO1 e LADO 2), determine e imprima o comprimento do terceiro lado (HIPOTENUSA). Suponha que todas as variáveis sejam inteiras.

NARRATIVA
1. Ler dados conhecidos
2. Calcular os quadrados dos lados conhecidos
3. Calcular o comprimento do terceiro lado
4. Sair resultados
5. Terminar

IMPLEMENTACÃO EM LINGUAGEM ALGORITMICA
1. Leia (LADO1, LADO2)
2. Q1 ← LADO1↑2 Q2 ← LADO2↑2
3. HIPOTENUSA ← SQRT(Q1+Q2) (*obs: SQRT() indica operação Raiz Quadrada*)
4. Escreva ('LADOS=', LADO1, LADO2, HIPOTENUSA)
5. Saída

ALGORITMO
1. [Ler dados conhecidos] Leia (LADO1, LADO2)
2. [Calcular os quadrados dos lados conhecidos] Q1←LADO1↑2 Q2←LADO2↑2
3. [Calcular o comprimento do terceiro lado] HIPOTENUSA ← SQRT(Q1+Q2)
4. [Sair resultados] Escreva ('LADOS=', LADO1, LADO2, HIPOTENUSA)
5. [Terminar] Saída

e) O algoritmo é expresso como uma sequência de passos numerados:
   i) Cada passo inicia-se com uma breve descrição entre colchetes;
   ii) Abaixo vem um comando, ou série de comandos, que descrevem as ações do passo;
   iii) Os comandos são executados na ordem estabelecida;
   iv) O comando "Saída" termina a execução do algoritmo no ponto onde é encontrado;
      (1) Saída representa o fim 'lógico' do algoritmo;
      (2) Um algoritmo pode ter diversos fins lógicos;
      (3) O fim físico é representado pelo símbolo '□'.

5) RASTREANDO UM ALGORITMO:

a) Rastrear é executar manualmente, com dados representativos, os passos do algoritmo;
   i) Registra-se manualmente (planilha eletrônica/editor de texto) os valores tomados pelas variáveis em cada passo do algoritmo;

b) Exemplo: supondo que as medidas conhecidas são: LADO1=3, LADO2=4
   i) Executamos o algoritmo preenchendo a tabela abaixo:

Passo	LADO1	LADO2	Q1	Q2	HIPOTENUSA	Saída
1	3	4	?	?	?	?
2	3	4	9	16	?	?
3	3	4	9	16	5	?
4	3	4	9	16	5	LADOS= 3, 4, 5

Obs: o símbolo '?' é utilizado para mostrar que a variável ainda não recebeu valor neste ponto.

6) METODOLOGIA DE SOLUÇÃO – SUMÁRIO

a) Assegurar-se do entendimento completo das especificações do problema:
   i) Indicar saída esperada para diversas amostras de dados de entrada;

b) Formular esboço geral do algoritmo, despreocupando-se com detalhes:
   i) Assegurar que a estratégia de solução está correta;
   ii) Rastrear o algoritmo com diversas amostras de dados;

c) Identificar e listar variáveis necessárias à execução:
   i) Incluir nome, tipo e indicação do propósito da variável;
   ii) Essa lista pode ser aumentada ou diminuída no decorrer do desenvolvimento;

d) Retornar aos passos individuais detalhando cada um deles:
   i) Verificar a cada desdobramento se os novos passos executam a função proposta;

e) Terminada a versão inicial, rastreie cuidadosamente todos os passos do algoritmo:
   i) Utilize diversas amostras de dados de entrada;

f) Validado o algoritmo, implemente em uma linguagem de programação apropriada.
   i) Obs.1: para problemas simples, essa abordagem parece ser desnecessária, mas é importante dominar a técnica para a abordagem de problemas mais complexos.
   ii) Obs.2: nem sempre existe um algoritmo adequado para determinados tipos de problemas. Se não conseguimos construí-los não implica que não existem.

7) EXERCÍCIO: RASTREAMENTO DO CALCULO DE MÉDIAS

a) Preparar algoritmo para ler temperatura na escala Celsius e imprimir equivalente em Fahrenheit: $F = \frac{9}{5}\,°C + 32$

b) Preparar algoritmo para ler nome e sobrenome e imprimir sobrenome e nome:
   i) Exemplo: entrada: 'FULANO', 'DE TAL' saída: 'DE TAL', 'FULANO'

c) Preparar algoritmo para calcular o desvio padrão de cinco números (amostral)
   i) $\sigma = \sqrt{\frac{1}{4.0}\sum_{i=1}^{5}(x_i - \bar{x})^2}$ onde x1, x2, ..., x5 são valores reais e $\bar{x}$ é a média

d) Preparar algoritmo para ler os valores *a*, *b* e c e imprimir o valor do discriminante Δ
   i) $\Delta = b^2\text{-}4ac$ e todos os valores são reais

e) Preparar algoritmo para ler a quantidade de cada item comprado e calcular a conta:
   i) Entrada de quantidades: (Hamburguer, Cheeseburguer, Fritas, Refrigerante, Shake)
   ii) Preços: Hamburguer: 6,50; Cheese: 7,50; Fritas: 3,50; Refrigerante: 1,0; Shake: 5,0

f) Preparar algoritmo para ler custo de fábrica do carro e imprimir preço ao consumidor:
   i) Preço = custo + 45% impostos + 12% lucro revendedor (sobre custos + impostos)

g) Preparar algoritmo para ler comprimento dos 3 lados do triângulo e calcular a área:
   i) Área=$\sqrt{T(T-L1)(T-L2)(T-L3)}$ onde $T = \frac{L1+L2+L3}{2}$

5. APLICAÇÕES DE ALGORITMOS

1) RELATÓRIO DE NOTAS DE ALUNOS

a) Calcular a média final do aluno a partir das notas parciais e seus respectivos pesos:
b) Variáveis do programa:
   i) NOME (cadeia), Nome do estudante
   ii) LABORATORIO (inteiro) Nota do laboratório peso 20%
   iii) PROVA_PARCIAL (inteiro) Nota da prova parcial peso 30%
   iv) EXAME_FINAL (inteiro) Nota do exame final peso 50%
   v) MEDIA (real) Média final do estudante
c) A solução em linguagem narrativa é direta:
   i) Ler o nome do estudante e suas notas em cada componente da média,
   ii) Calcular a média ponderada dos componentes
   iii) Imprimir o nome, as notas e a média final do estudante
d) O Algoritmo resultante é direto:

```
1. [Ler dados de entrada]
        Leia(NOME, LABORATORIO, PROVA_PARCIAL, EXAME_FINAL)

2. [Calcular média]
        MEDIA ← 0.2*LABORATORIO+0.3*PROVA_PARCIAL+0.5*EXAME_FINAL

3. [Imprimir resultados]
        Escreva(´Nome: ‘, NOME)
        Escreva(‘Nota do Laboratório: ‘, LABORATORIO)
        Escreva(´Nota da Prova Parcial: ‘, PROVA_PARCIAL)
        Escreva(‘Nota do Exame Final: ‘, EXAME_FINAL)
        Escreva(‘Média Final: ‘, MEDIA)

4. [Terminar]
        Saída
```

a) Observação: quando se multiplica um valor inteiro por um valor real o resultado é real; A formatação da saída será abstraída: é altamente dependente da linguagem escolhida;
b) Rastreando o Algoritmo: Para verificar se o algoritmo está funcionando corretamente, utilizam-se dados de entrada cujos resultados são conhecidos, e verifica-se sua saída;
   i) É importante testar condições extremas dos dados (condições de contorno), como, por exemplo, todas as notas iguais a 100 e todas as notas iguais a zero;
c) Usando dados de entrada abaixo, verifique o resultado:
   i) ‘Moacir Rodolfo’, 72, 68, 65;
   ii) A média final calculada será 67.3;
d) Quando os testes não indicam erro, duas coisas podem estar ocorrendo, mas nunca saberemos com certeza qual delas é a verdadeira:
   i) O algoritmo não contém erros;
   ii) Os erros do algoritmo não foram detectados nos testes;
e) Erros frequentemente resultam de circunstâncias que não foram previstas. Uma mentalidade “sabotadora” auxilia um pouco nessa fase de testes do programa.

2) MEDINDO A TAXA DE INFLAÇÃO

a) O algoritmo deve ler o nome do produto, o preço no mês atual, o preço no mês anterior e calcular a taxa de inflação desse produto nesse mês. O resultado deve ser apresentado em dois formatos: diferença algébrica (absoluta) e diferença percentual.

b) A primeira tentativa de solução, em linguagem natural:

1. Ler a descrição do item, o preço atual e o preço anterior;
2. Calcular a diferença algébrica no preço;
3. Calcular a diferença percentual no preço;
3. Imprimir resultados
4. Terminar

c) As variáveis definidas previamente para a solução do problema e sua descrição;

i) Esse passo é muito importante em algoritmos mais complexos ou extensos;

1. ITEM (cadeia)	Descrição do item,
2. PRECO_ATUAL (real)	Preço pago neste mês
3. PRECO_ANTERIOR (real)	Preço pago no mês anterior
3. DIFER_ALGEB (real)	Diferença algébrica
4. DIFER_PERC (real)	Diferença Percentual

d) O algoritmo final é de implementação direta

```
1. [Ler dados de entrada]
        Leia(ITEM, PRECO_ATUAL, PRECO_ANTERIOR)
2. [Calcular a diferença algébrica dos preços]
        DIFER_ALGEB ← PRECO_ATUAL – PRECO_
3. [Calcular a diferença percentual dos preços]
        DIFER_PERC ← DIFER_ALGEB / PRECO_ANTERIOR * 100
4. [Imprimir resultados]
        Escreva(´Item Comprado: ‘, ITEM)
        Escreva(‘Preço no mês atual: ‘, PRECO_ATUAL)
        Escreva(´Preço no mês anterior: ‘, PRECO_ANTERIOR)
        Escreva(‘Diferença algébrica: ‘, DIFER_ALGEB)
        Escreva(‘Diferença percentual: ‘, DIFER_PERC)
5. [Terminar]
        Saída
```

e) Testando o algoritmo com três conjuntos de dados

i) ‘Pacote de Queijo’, 6.54, 6.00
ii) ‘Feijão’, 2.30, 2.40
iii) ‘Leite’, 2.00, 1.70

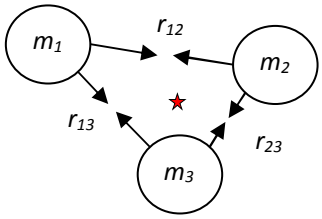
3) EXERCÍCIOS DO CAPÍTULO

a) Você foi encarregado pelos organizadores de uma competição internacional de preparar um serviço de "tradução simultânea" para resultados da competição relatados em unidades métricas. Desenvolver algoritmos para manusear os seguintes eventos:
   i) Converter resultados de salto em altura, relatados em metros, para pés e polegadas (ft e in) (1 pé vale 12 polegadas e 1 metro vale 39.37 polegadas).
   ii) Dado o tempo para uma corrida de 100 metros, calcular seu tempo para 100 jardas. Supor que o atleta corra a uma velocidade constante (1 jarda = 3 pés ou 0.9144 metros).

b) Uma lista de taxas de câmbio de 1977, para troca de moeda estrangeira, fornece a seguinte tabela de equivalência:
   (1) 100 Francos franceses = 21.55 dólares canadenses
   (2) 1 Dólar americano = 1.06 dólares canadenses
   (3) 100 Marcos alemães = 43.20 dólares canadenses
   (4) 1 Libra inglesa = 1.84 dólares canadenses
   (5) 100 coroas suecas = 24.25 dólares canadenses .
   (6) 100 dracmas gregos = 2.95 dólares canadenses
   i) Desenvolver algoritmos para fazer as seguintes conversões;
      (a) Ler uma quantidade em francos franceses e imprimir o equivalente em dólares canadenses.
      (b) Ler uma quantidade em dólares americanos e imprimir o equivalente em ambos, coroas suecas e francos franceses.
      (c) Ler uma quantidade em dracmas gregos e imprimir o equivalente em libras inglesas.
      (d) Ler uma quantidade em dólares canadenses e imprimir o equivalente em ambos, dólares americanos e marcos alemães.

c) A Companhia de Carros Usados João Honesto paga a seus empregados um salário de R$ 6.000,00 por mês mais uma comissão de R$ 500,00 para cada carro vendido mais abono de 5% do valor da venda. Todo mês a companhia prepara um registro para cada vendedor, contendo seu nome, o número de carros vendidos e o valor total das vendas. Preparar um algoritmo para calcular e imprimir o salário do vendedor num dado mês. Testar o algoritmo completamente, utilizando um conjunto apropriado de dados.

d) Três massas, m_1, m_2 e m_3 , estão separadas por distâncias r_{12}, r_{13} e r_{23}, como mostra a Figura abaixo. Se G é a constante de gravitação universal, a energia de coesão mantendo a massa das partículas juntas é dada pela fórmula:



$$E = G\left(\frac{m_1 m_2}{r_{12}} + \frac{m_1 m_3}{r_{13}} + \frac{m_2 m_3}{r_{23}}\right)$$

   i) Preparar um algoritmo para ler valores de m_1, m_2 e m_3; r_{12}, r_{13} e r_{23}; calcular e imprimir a energia de coesão, imprimindo também os valores iniciais dados. Para massa em quilogramas e distância em metros, G = 6.67 X 10-11 N. m2 /kg2 . Os valores de m_1, m_2 e m_3, são fornecidos no primeiro registro de dado e os valores de r_{12}, r_{13} e r_{23} são fornecidos no segundo. Assumir que todos os dados são fornecidos como valores reais.

e) Um sistema de equações lineares da forma: $\begin{cases} ax + by = c \\ dx + ey = f \end{cases}$

pode ser resolvido utilizando-se as seguintes fórmulas:

$$x = \frac{ce-bf}{ae-bd} \qquad \text{e} \qquad y = \frac{af-cd}{ae-bd}$$

i) Preparar um algoritmo para ler o conjunto de coeficientes (a, b, c e d, e, f) e imprimir a solução (x e y). Existem casos para os quais este algoritmo não funciona?

f) O custo de seguro contra granizo numa comunidade típica de fazendeiros é 3.5% do valor de cobertura solicitado por acre, multiplicado pelo número de acres plantados. Supondo que as possibilidades de colheitas sejam limitadas a trigo, aveia e cevada, preparar um algoritmo para ler a cobertura desejada e o número de acres plantados para cada uma das três plantações e calcular o custo total do prêmio do seguro para este cliente.

g) Recentemente, conduziu-se uma experiência para determinar a aceleração da gravidade em Saskatoon. Deixou-se cair uma bola a partir do repouso, do alto de vários edifícios. O tempo gasto para atingir o solo foi registrado em cada caso. Um total de cinco medidas foram feitas, com os seguintes resultados:

Altura do prédio (m)	Tempo para atingir o solo (s)
69	3.74
114	4.84
216	6.64
129	5.13
48	3.11

i) Preparar um algoritmo para calcular a aceleração gravitacional g, a partir destes dados, utilizando a fórmula:

$$y = \frac{1}{2}gt^2$$

onde *y* representa a distância percorrida pela bola e *t* o tempo gasto. Cada um dos cinco casos mostrados dará um valor de *g*; o "melhor valor" para *g* para a localidade de Saskatoon, - com base nestes resultados, é dado pela média aritmética das cinco medidas. O algoritmo deve ler os dados fornecidos e imprimir o valor médio encontrado. O formato de entrada é: altura-tempo, altura-tempo, altura-tempo, etc. de cada lançamento.

h) Embora a velocidade da luz seja constante, não importando a velocidade relativa da fonte e do observador, o comprimento de onda e a freqüência mudam - um efeito previsto pela primeira vez por Johann Doppler e conhecido como o "Efeito Doppler". O comprimento de onda λ emitido por uma fonte movendo-se em direção ao observador com uma velocidade v é comprimido por uma quantidade $\Delta\lambda$, que é dada pela fórmula:

$$\Delta\lambda = \frac{v\lambda}{c}$$

onde c é a velocidade da luz. Suponha que um avião esteja voando em direção a uma estação de rádio com uma velocidade constante de 360 km/h (10^4 cm/s). Se a estação de rádio está emitindo num comprimento de onda de 30 metros, a mudança no comprimento de onda devido ao “Efeito Doppler” é:

$$\Delta\lambda = \frac{v\lambda}{c} = \frac{(10^4 cm/s)*(3*10^3 cm)}{3*10^{10} cm/s} = 10^{-3}\text{cm}$$

Então, o piloto do avião deve ajustar o receptor para um comprimento de onda de 3,000 cm menos 10^{-3}cm, ou seja, 2999.999 cm, enquanto está se aproximando da estação, e então para o comprimento de onda de 3000.001 cm enquanto estiver se afastando da estação (observação: utilizar padrão americano: ponto separando casas decimais).

i) Preparar um algoritmo para ler o coprimento de onda emitido pela estação e a velocidade de aproximação do avião, e então, imprimir o valor real de λ com o qual o piloto receberá o sinal.

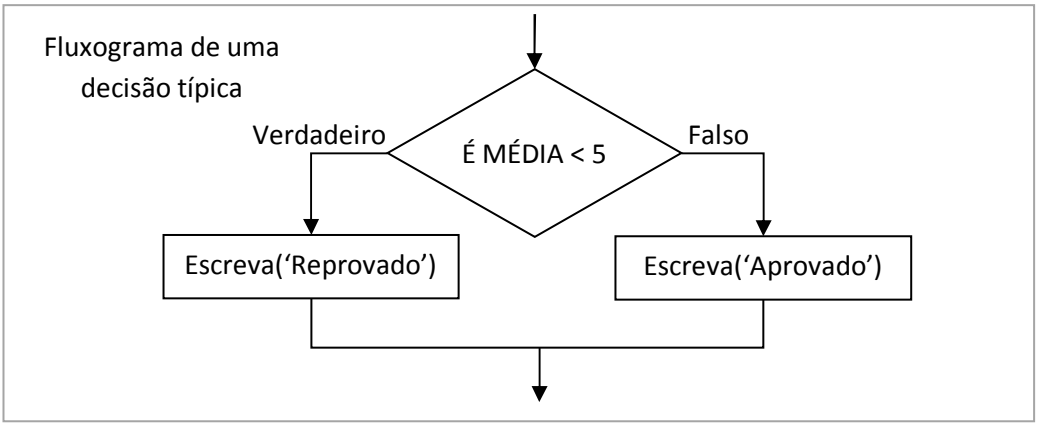
6. ESTRUTURAS DE DECISÃO E DE CONTROLE

6.1. JUSTIFICATIVA

a) Algoritmo simples resolvem problemas simples:
   i) Entrada – cálculo – saída
   ii) Não envolve tomada de decisões no decorrer do processo
b) Em situações de repetição ou mais complexas é necessário a utilização de duas estruturas:
   i) Decisão: Se-Então-Senão (ou simplesmente estrutura 'Se')
   ii) Repetição: Repita Até que uma CONDIÇÃO seja satisfeita
c) Exemplo: calcular a nota de 250 alunos, indicando "Aprovado" ou "Reprovado"
   i) Repita Até terminarem os dados:
      (1) calcule a média.
      (2) Se MEDIA < 5 Então Escreva('Reprovado') Senão Escreva('Aprovado')

6.2. SELEÇÃO DE ALTERNATIVAS

a) A Unidade Aritmética e Lógica do processador é o "cérebro" do computador
   i) A Unidade Aritmética executa os cálculos necessários;
   ii) A Unidade Lógica fornece a habilidade de tomar decisões;



   iii) A decisão é especificada numa expressão lógica: MEDIA < 5;
   iv) O resultado dessa expressão pode ser verdadeiro ou falso;
   v) O fluxo do programa é alterado conforme o resultado da expressão;

6.2.1. Construção Se – Então – Senão

a) Se "condição"
   Então (alternativa verdadeira) faça isso
   Senão (alternativa falsa) faça aquilo

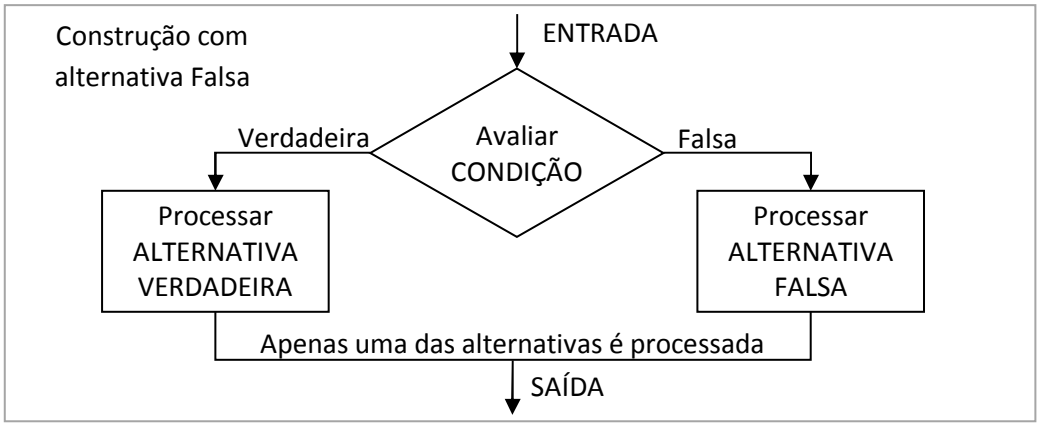
```
Se MEDIA < 5
Então Escreva('Reprovado')
Senão Escreva('Aprovado')
```

d) Cada componente dessa estrutura pode ser consideravelmente mais complexo;
   i) Para tornar mais clara a estrutura, utiliza-se recuos de texto a cada componente;
e) O conjunto de operadores utilizados para expressar condições é indicado abaixo:

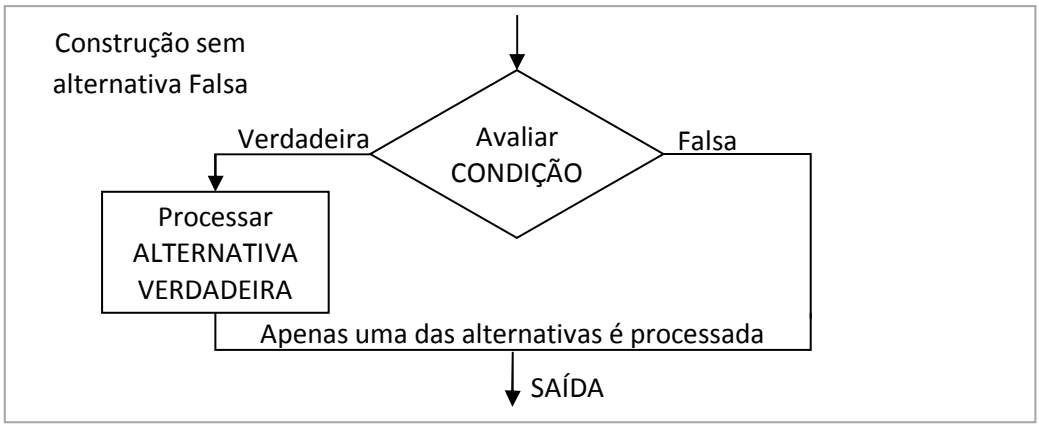
Operador	Significado
>	Maior do que
<	Menor do que
=	Igual a
≥	Maior do que ou igual a
≤	Menor do que ou igual a
≠	Não igual a (diferente de)

Os operadores podem ser aplicados em contextos numéricos e não numéricos. Num contexto não numérico, por exemplo, podem ser utilizados na classificação em ordem alfabética

f) A estrutura Se-Então-Senão tem uma entrada e uma saída apenas:



g) Eventualmente, pode não existir uma ação especial se a condição é falsa (ou verdadeira):



h) Exemplo: Algoritmo para ler dois valores, determinar o maior deles e imprimir o valor com uma mensagem de identificação. Assumir que todas as variáveis são inteiras.

```
1. [Ler dados de entrada]
        Leia(VALOR1, VALOR2)
2. [Determinar o maior]
        Se VALOR1 > VALOR2
        Então | MAIOR ← VALOR1
              | Escreva('Primeiro valor é o maior')
        Senão | MAIOR ← VALOR2
              | Escreva('Segundo valor é o maior')
3. [Imprimir resultados]
        Escreva(´Valor maior: ', MAIOR)
4. [Terminar]
        Saída
```

i) Análise do algoritmo:
   i) Passo 1: Lê dois valores, por exemplo 2 e 6, e atribui esses valores às variáveis VALOR1 e VALOR2, respectivamente;
   ii) Passo 2: A condição VALOR1 > VALOR2 é avaliada:
      (1) Se a condição for verdadeira, o valor da variável VALOR1 é atribuído à variável MAIOR, e o algoritmo imprime 'Primeiro valor é o maior';
      (2) Se a condição for falsa, o valor da variável VALOR2 é atribuído à variável MAIOR e o algoritmo imprime 'Segundo valor é o maior';
   iii) Passo 3: o algoritmo imprime o valor da variável MAIOR precedido de 'Valor maior:'

j) Linguagem Estruturada:
   i) Recuos equivalentes para estruturas equivalentes;
   ii) Recuos maiores para estruturas subordinadas.

k) Exemplo: Em uma prévia eleitoral, um candidato só é escolhido se o total de seus votos excede a soma de votos dados a todos os outros candidatos. Se nenhum candidato é escolhido numa rodada particular, o candidato com o menor número de votos é eliminado e um novo escrutínio é conduzido. Suponha que em uma rodada particular, quatro candidatos estejam concorrendo. O algoritmo deve determinar o resultado dessa votação particular. A votação e nomes de cada candidato são fornecidos em ordem decrescente de votos, em dois registros de entrada.

```
1. [Ler dados de entrada]
        Leia(VOTOS1, VOTOS2, VOTOS3, VOTOS4)
        Leia(NOME1, NOME2, NOME3, NOME4)

2. [Verificar o resultado da votação]
        Se VOTOS1 > (VOTOS2 + VOTOS3 + VOTOS4)
        Então   Escreva(NOME1, 'Nomeado com', VOTOS1, 'Votos')
        Senão   Escreva('Sem nomeação –', NOME4, 'Eliminado da votação')

3. [Terminar]
        Saída
```

l) Rastreamento do Algoritmo:
   i) Entrada:

```
764, 419, 307, 175
'Marcelo', 'Marcos', 'Lindolfo', 'Alexandre'
```

Passo	VOTOS1	VOTOS2	VOTOS3	VOTOS4	NOME1	NOME2	NOME3	NOME4	CONDIÇÃO	SAÍDA
1	764	419	307	175	Marcelo	Marcos	Lindolfo	Alexandre	-	-
2	764	419	307	175	Marcelo	Marcos	Lindolfo	Alexandre	falsa	Sem nomeação Alexandre Eliminado da votação

Obs: a construção do algoritmo completo para verificar resultados da votação é muito mais complexo.

6.2.2. Estruturas Se-Então-Senão aninhadas (ou embutidas)

a) Em algumas aplicações, uma (ou as duas) das alternativas da estrutura pode(m) envolver outras decisões. Por exemplo, determinar o maior de três valores dados A, B e C. Primeiramente, verifica-se se A é maior que B, e então, compara-se o resultado com C.

```
1. [Ler os valores dados]
        Leia(A, B, C)

2. [Determinar o maior valor, comparando os pares]
        Se A > B
        Então   Se A > C
                Então MAIOR ← A
                Senão MAIOR ← C
        Senão   Se B > C
                Então MAIOR ←B
                Senão MAIOR ←C

3. [Imprimir o maior valor]
        Escreva('Maior valor: ', MAIOR)

4. [Terminar]
        Saída
```

A diagramação estruturada facilita a compreensão e o rastreamento da estrutura lógica do algoritmo.

m) Observação: a diagramação estruturada (recuos) pode alterar o sentido da estrutura:

```
Se Condição 1
Então  Se Condição 2
          Então Sequencia 1
          Senão Sequência 2
```

```
Se Condição 1
Então  Se Condição 2
          Então Sequencia 1
Senão Sequência 2
```

Obs: pode não haver comandos em uma das condições

n) Exemplo: Gratificação de natal baseada em dois critérios: número de horas-extras trabalhadas e o número de horas que o empregado faltou ao trabalho. Cada empregado tem um registro com o nome, número de horas extras trabalhadas e o número de horas de faltas. A fórmula para calcular o prêmio é subtrair de suas horas-extras dois terços das horas de faltas e distribuir o prêmio e acordo com a tabela abaixo. Observação: nenhum funcionário pode cair em mais de uma categoria.

Hora extra – 2/3 faltas	Prêmio em Reais
> 40 horas	5000
> 30 mas ≤ 40 horas	4000
> 20 mas ≤ 30 horas	3000
> 10 mas ≤ 20 horas	2000
≤ 10 horas	1000

Variável	Tipo	Descrição
NOME	Cadeia	nome do funcionário
H_EXTRAS	Inteiro	horas-extras trabalhadas
FALTAS	Inteiro	horas faltadas ao trabalho
H_BASE	Real	horas-base para o prêmio
PREMIO	Real	valor do bônus a ser pago

```
1. [Ler os valores dados]
        Leia(NOME, H_EXTRAS, FALTAS)

2. [Calcular o valor da hora-base para enquadramento no prêmio]
        H_BASE ← H_EXTRAS – FALTAS * 2 / 3.0

3. [Determinar o valor do prêmio]
        Se H_BASE > 40
        Então PREMIO ← 5000
        Senão  Se H_BASE > 30
                  Então PREMIO ← 4000
                  Senão  Se H_BASE > 20
                            Então PREMIO ← 3000
                            Senão  Se H_BASE > 10
                                      Então PREMIO ← 2000
                                      Senão PREMIO ← 1000

4. [Imprimir resultados]
        Escreva('Prêmio para ', NOME, 'R$', PREMIO)

5. [Terminar]
        Saída
```

i) Suponha que um empregado tenha trabalhado 50 horas-extras e faltado apenas 3. Qual é o valor do Prêmio? Calcule o valor de H_BASE: 48;

ii) Suponha que outro empregado tenha trabalhado 30 horas-extras e faltado 12 horas. Qual é o valor do Prêmio? Calcule o valor de H_BASE: 22;

o) Exemplo: Algoritmo para ler três números A, B e C, e imprimir o maior e o menor valor. Assumir que os três valores são diferentes.

Variável	Tipo	Descrição
A, B, C	Inteiro	3 números diferentes
MIN	Inteiro	menor valor
MAX	Inteiro	maior valor

```
1. [Ler os valores dados]
        Leia(A, B, C)
2. [Determinar o maior e o menor valor]
        Se A < B
        Então  Se A < C
               Então  MIN ← A                    A < B, C
                      Se B > C
                      Então MAX ← B              A < B > C
                      Senão MAX ← C              A < B < C
               Senão  MIN ← C
                      MAX ← B                    C < A < B
        Senão  Se A > C
               Então  MAX ← A                    B, C < A
                      Se B > C
                      Então MIN ← C              C < B < A
                      Senão MIN ← B              B < C < A
               Senão  MAX ← C
                      MIN ← B                    B < A < C
3.  [Imprimir resultados]
        Escreva('Maior valor:', MAX, 'Menor valor:', MIN)
4. [Terminar]
        Saída
```

6.2.3. Sumário do comando Se-Então-Senão (ou simplesmente Se):

a) Ferramenta útil e poderosa para a resolução de problemas;
p) Praticamente todas as linguagens de programação tem alguma forma desse comando;
q) A decisão fornece duas alternativas de desvio do programa;
   i) Não é necessário ter ações nas duas alternativas – uma delas pode ser omitida;
r) Comandos Se aninhados (embutidos) permite testar múltiplas condições;
   i) Excesso de condições podem levar a confusões;
s) Escrita estruturada é muito importante para a clareza das construções;
   i) É fundamental nas construções aninhadas;
t) Comentários e espaçamentos adequados são contribuições importantes para a clareza da documentação do algoritmo.

EXERCÍCIOS

1.a. Modificar o algoritmo RELATÓRIO (abaixo) de modo que cada linha impressa contenha A média final e uma mensagem de resultado: Aprovado / Reprovado.

1.b. Modificar o algoritmo RELATÓRIO (abaixo) de modo que haja atribuição de Conceito de acordo com sua nota final. Assumir que os conceitos são baseados na tabela abaixo.

```
1. [Ler dados de entrada]
        Leia(NOME, LABORATORIO, PROVA_PARCIAL, EXAME_FINAL)
2. [Calcular média]
        MEDIA ← 0.2*LABORATORIO+0.3*PROVA_PARCIAL+0.5*EXAME_FINAL
3. [Imprimir resultados]
        Escreva('Nome: ', NOME)
        Escreva('Nota do Laboratório: ', LABORATORIO)
        Escreva('Nota da Prova Parcial: ', PROVA_PARCIAL)
        Escreva('Nota do Exame Final: ', EXAME_FINAL)
        Escreva('Média Final: ', MEDIA)
4. [Terminar]
        Saída
```

RELATÓRIO

Variáveis do programa:	
NOME	(cadeia)
LABORATORIO	(inteiro)
PROVA_PARCIAL	(inteiro)
EXAME_FINAL	(inteiro)
MEDIA	(real)

Média	Conceito
80 – 100	A
70 – 79	B
60 – 69	C
50 – 59	D
< 50	E

2. Preparar um Algoritmo para ler a base e a altura de um triângulo e imprimir a área deste triângulo (área = base * altura / 2). Durante a preparação desse algoritmo, é possível que se cometa um erro e entrem valores negativos para a base ou para a altura. Isto é indesejável, pois a área impressa será negativa. Prever no algoritmo a possibilidade de verificar valores negativos na entrada. Se um valor negativo é encontrado, imprimir uma mensagem identificando este valor como base ou altura (isso permite corrigir o erro mais facilmente). Testar o algoritmo cuidadosamente. Tomar cuidado no caso de ambos os valores serem negativos. Isso produziria uma área positiva e o erro não seria detectado.

3. Revisar o algoritmo MAX_3 para levar em conta a possibilidade de entrarem valores iguais.

MAX_3

```
1. [Ler os valores dados]
        Leia(A, B, C)
2. [Determinar o maior valor, comparando os pares]
        Se A > B
        Então  Se A > C
               Então MAIOR ← A          B, C < A
               Senão MAIOR ← C          B< A < C
        Senão  Se B > C
               Então MAIOR ←B           A, C < B
               Senão MAIOR ←C           A < B < C
3. [Imprimir o maior valor]
        Escreva('Maior valor: ', MAIOR)
4. [Terminar]
        Saída
```

4. Preparar um algoritmo para ler os comprimentos dos três lados de um triângulo (S1, S2, S3) e determinar que tipo de triângulo temos, com base nos casos abaixo. Sejam A o maior dos lados de S1, S2 e S3 e B e C os outros dois. Então:

Se $A \geq B + C$	nenhum triângulo é formado
Se $A^2 = B^2 + C^2$	um triângulo retângulo é formado
Se $A^2 > B^2 + C^2$	um triângulo obtusângulo é formado
Se $A^2 < B^2 + C^2$	um triângulo acutângulo é formado

5. A Secretaria do Meio Ambiente mantém três listas de indústrias conhecidas por serem altamente poluentes da atmosfera. Os resultados de várias medidas são combinados para formar o que é chamado de "índice de poluição". Isto é controlado regularmente. Normalmente os valores caem entre .05 e .25 (Bom). Se o valor atingir .30 (Ruim), as indústrias da lista A serão chamadas a suspender as operações até que os valores retornem ao intervalo normal. Se o índice atingir .40 (Péssimo), as indústrias da lista B serão notificadas também. Se o índice exceder .50 (Crítico), indústrias de todas as três listas serão avisadas para suspenderem as atividades. Preparar um algoritmo para ler o índice de poluição e indicar as notações apropriadas.

6. Preparar um algoritmo para ler um valor inteiro e determinar se ele é par ou impar. Generalização: ler *m* e *n* e determinar se *m* é divisível por *n (m* e *n* são inteiros).

7. Modificar o algoritmo NOMEAÇÃO de modo que não seja necessário classificar previamente os resultados da votação. Assumir que não existem candidatos com o mesmo número de votos, isto é, os votos são diferentes entre si.

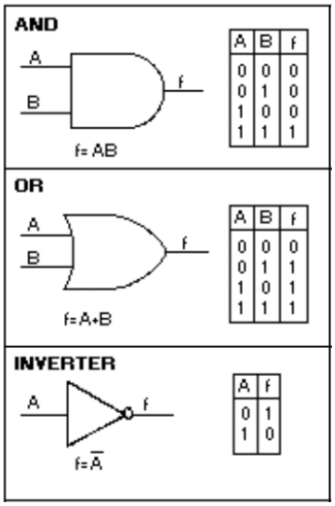
NOMEAÇÃO

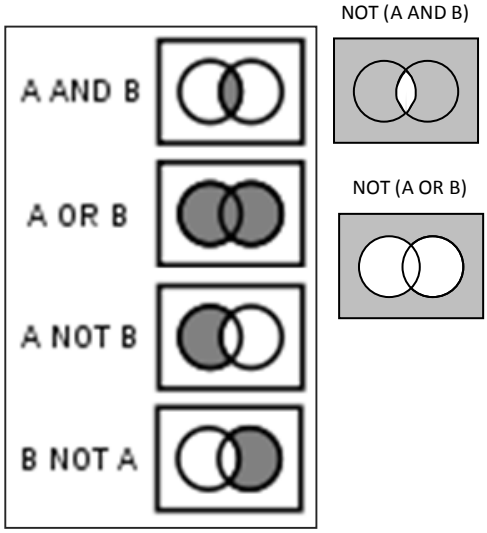
```
1. [Ler dados de entrada]
        Leia(VOTOS1, VOTOS2, VOTOS3, VOTOS4)
        Leia(NOME1, NOME2, NOME3, NOME4)
2. [Verificar o resultado da votação]
        Se VOTOS1 > (VOTOS2 + VOTOS3 + VOTOS4)
        Então   Escreva(NOME1, 'Nomeado com', VOTOS1, 'Votos')
        Senão   Escreva('Sem nomeação –', NOME4, 'Eliminado da votação')
3. [Terminar]
        Saída
```

7. UTILIZAÇÃO DE CONDIÇÕES COMPOSTAS

a) Condições simples em alguns casos conduzem a soluções complicadas e muito difíceis de entender – ver algoritmo MAX_3 (exercício 3.2.2. pg. 21);

b) Condições compostas utilizam os conectivos lógicos “e”, “ou” e “não”, cujo resultado é mostrado nas “tabelas-verdade”:





Obs: A aplicação em algoritmos tem o seguinte resultado:

“E”: o resultado é verdadeiro somente se as duas condições forem verdadeiras;

“OU”: o resultado é verdadeiro se pelo menos uma das duas condições for verdadeira;

“NÃO: aplica-se a apenas uma condição: { o resultado é verdadeiro se a condição é falsa.
o resultado é falso se a condição é verdadeira.

c) Exemplos – considerar UM ← 1

i) (UM < 2) e (UM < 0) FALSO

ii) (UM < 2) ou (UM < 0) VERDADEIRO

iii) Não (UM < 2) FALSO

Obs: não é necessário colocar as condições entre parênteses – é um recurso de clareza;
(os operadores relacionais tem prioridade sobre os operadores conectivos)

d) As condições compostas podem ser utilizadas tanto nos comandos “Se” quando nos comandos “Repita” condicionais.

e) Exemplo: algoritmo MAX_3 (maior de 3 valores) utilizando condições compostas:

```
1. [Ler os valores dados]
        Leia(A, B, C)

2. [Determinar o maior valor]
        Se A > B
        Então   Se A > C
                Então MAIOR ← A
                Senão MAIOR ← C
        Senão   Se B > C
                Então MAIOR ←B
                Senão MAIOR ←C

3. [Imprimir o maior valor]
        Escreva('Maior valor: ', MAIOR)

4. [Terminar]
        Saída
```

```
1. [Ler os valores dados]
        Leia(A, B, C)

2. [Determinar o maior valor]
        Se A > B  e  A > C então MAIOR ← A
        Se B > A  e  B > C então MAIOR ← B
        Se C > A  e  C > B então MAIOR ← C

3. [Imprimir o maior valor]
        Escreva('Maior valor: ', MAIOR)

4. [Terminar]
        Saída
```

Algoritmo mais claro, mais fácil de entender, com condições claramente definidas

Tanto o embutimento quanto as condições compostas tem seu limite na clareza do algoritmo.

f) Exemplo: algoritmo MAX_MIN_3
Supor que os 3 números são diferentes

```
1. [Ler os valores dados]
        Leia(A, B, C)

2. [Determinar o maior e o menor valor]
        Se A < B
        Então   Se A < C
                Então   MIN ← A
                        Se B > C
                        Então MAX ← B
                        Senão MAX ← C
                Senão   MIN ← C
                        MAX ← B
        Senão   Se A > C
                Então   MAX ← A
                        Se B > C
                        Então MIN ← C
                        Senão MIN ← B
                Senão   MAX ← C
                        MIN ← B

3.  [Imprimir resultados]
        Escreva('Maior:', MAX,
                'Menor:', MIN)

4. [Terminar]
        Saída
```

```
1. [Ler os valores dados]
        Leia(A, B, C)

2. [Determinar o maior e o menor valor]
        Se A > B  e  B > C então  MAX ← A          A>B>C
                                  MIN ← C
        Se B > C  e  C > A então  MAX ← B          B>C>A
                                  MIN ← A
        Se C > B  e  B > A então  MAX ← C          C>B>A
                                  MIN ← A
        Se C > A  e  A > B então  MAX ← C          C>A>B
                                  MIN ← B
        Se A > C  e  C > B então  MAX ← A
                                  MIN ← B          B>A>C
        Se B > A  e  A > C então  MAX ← B
                                  MIN ← C

3.  [Imprimir resultados]
        Escreva('Maior:', MAX,  'Menor:', MIN)

4. [Terminar]
        Saída
```

Variável	Tipo	Descrição
A, B, C	Inteiro	3 números diferentes
MIN	Inteiro	menor valor
MAX	Inteiro	maior valor

EXERCÍCIOS

1. Suponha que M e N sejam variáveis inteiras com valores 6 e 12 respectivamente. Quais das seguintes condições são verdadeiras?

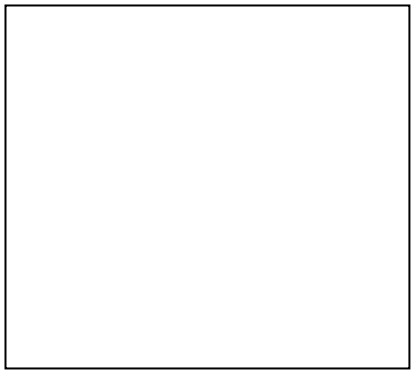
a) $2 * 1 \leq N$ ☐	b) $2 * M - 1 < N$ ☐	c) $M > 0$ e $M \leq 10$ ☐
d) $M > 25$ ou ($M < 50$ e $N < 50$) ☐	e) $M < 4$ ou $N > 5$ ☐	f) Não $M > 6$ ☐

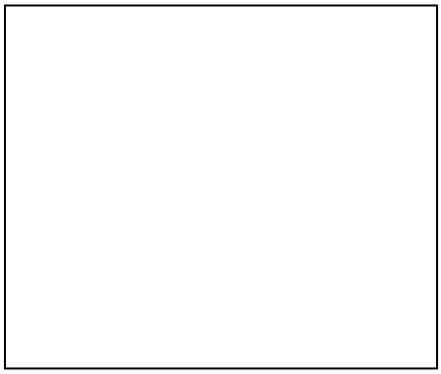
a)	b)	c)	d)	e)	f)

2. Assuma que A, B, C e D são variáveis e S1, S2, S3 e S4 são comandos ou alternativas.
   a) Expresse as condições necessárias para a execução de S1, S2, S3 e S4 no seguinte comando como condições compostas separadas:

```
Se A > B
Então   Se B ≤ C
        Então   Se C ≠ D
                Então S1
                Senão S2
        Senão S3
Senão S4
```

b) Expresse a seguinte estrutura “se” utilizando somente condições simples embutidas:
Se (A < B e C ≠ D) e (B > D ou B = D) Então S1





3. Recomendam-se estudantes para bolsas de estudo em função de seu desempenho anterior. A natureza das recomendações é baseada na seguinte tabela:

Média	Recomendação
≥ 90	Altamente recomendado
≥ 80 mas < 90	Fortemente recomendado
≥ 70 mas < 80	Recomendado
< 70	Não recomendado

Cada solicitação de bolsa é inserida em um registro com o seguinte formato:

Nome do estudante, Média Geral

Preparar um Algoritmo para ler o conjunto de cartões dos solicitantes de bolsas e preparar uma relação dando o nome de cada estudante, sua média e a recomendação. No final da relação (indicada por um registro-sentinela com o estudante de nome ‘FIM DE ARQUIVO’), imprimir a média geral dos solicitantes e o número de recomendações de cada tipo.

4. O Ibitiúva Esporte Clube deseja aumentar o salário de seus 20 jogadores registrados. O ajuste salarial deve obedecer à seguinte tabela:

Salário atual			Aumento
		≤ 900.00	20%
> 900.00	a	1300.00	10%
> 1300.00	a	1800.00	5%
		> 1800.00	0%

Preparar um algoritmo para ler o nome e o salário atual de cada jogador e imprimir seu nome, o salário atual e o salário ajustado. No final da impressão, dar o total pago atualmente e o total ajustado segundo as regras acima.

5. Numa classe, são feitos cinco exames (A, B, C, D, E).
Pedem-se estatísticas para determinar o número de estudantes que:
   a) Passou em todos os exames;
   b) Passou em A, B e D mas não em C ou E
   c) Passou em A e B, C ou D, mas não em E

Preparar um algoritmo apropriado, utilizando o método fim-de-arquivo para indicar o fim dos dados de entrada.

ALGORITMOS

II

8. ENLAÇAMENTO

a) Computadores são particularmente bem adequados a aplicações nas quais operações são repetidas muitas vezes. Para isso, o LAÇO é uma construção fundamental.
   i) Exemplo: cálculo do fatorial de um número N: N!= N*(N-1)*(N-2)*...*(2)*(1);5

b) Algoritmo FAT_N (fatorial de N): calcula o fatorial de um valor dado
As variáveis N, PRODUTO e MULTIPLICADOR são do tipo Inteiro:

```
1. [Ler N]
        Leia(N)
2. [Inicializar variáveis]          (obs: conceito novo)
        PRODUTO ← 1
        MULTIPLICADOR ← N
3. [Estabelecer o laço]
        Repita até o passo 5 Enquanto MULTIPLICADOR ≥ 1
4. [Calcular o produto parcial]
        PRODUTO ← PRODUTO * MULTIPLICADOR
5. [Reajustar o multiplicador]
        MULTIPLICADOR ← MULTIPLICADOR -1
6. [Imprimir resultados]
        Escreva('Fatorial de', N, '=', PRODUTO)
7. [Terminar]
        Saída
```

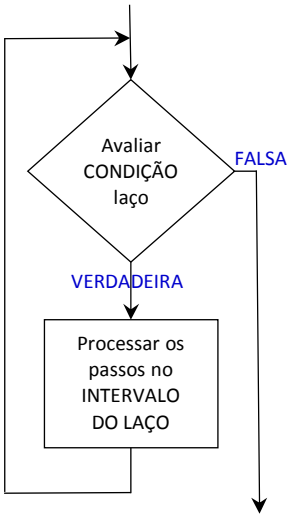
   ii) O laço é controlado pelo comando Repita no passo 3, que controla a repetição dos passos 4 e 5 enquanto o valor da variável MULTIPLICADOR for maior ou igual a 1.

8. 1. Laços Condicionais

a) Na notação algorítmica o laço condicional é expresso da seguinte forma:

Repita até o passo S_n enquanto "condição" for verdadeira

b) Nós nos referimos aos passos especificados no comando (passos a serem repetidos) como "amplitude do laço";
   i) A "condição" é avaliada antes de se "entrar" no laço
   ii) Os passos do laço são repetidos enquanto a "condição" permanecer verdadeira;
   iii) Quando a "condição" se tornar falsa, o curso do algoritmo é desviado para o primeiro passo após o laço (após os comandos da amplitude do laço);
   iv) Em algumas linguagens a amplitude do laço é claramente definida e não é necessário indicar o passo final do laço (este é claramente indicado);



c) A estrutura Repita tem um único ponto de entrada e um único ponto de saída;
   i) Este ponto é o próprio comando Repita;
   ii) É fundamental garantir que o laço tenha um ponto de saída, caso contrário o laço entra em "loop" e não termina naturalmente.

d) Rastreamento do algoritmo FAT_N considerando N = 3

	Passo	N	PRODUTO	MULTIPLICADOR	Condição	Saída
	1	3	?	?		
	2	3	1	3		
	3	3	1	3	verdadeira	
1	4	3	3	3		
	5	3	3	2		
	3	3	3	2	verdadeira	
2	4	3	6	2		
	5	3	6	1		
	3	3	6	1	verdadeira	
3	4	3	6	1		
	5	3	6	0		
	3	3	6	0	falsa	
	6	3	6	0		Fatorial 3 = 6
	7	Terminar (saída)				

i) O passo 1 lê o valor de entrada (3) e o atribui à variável N;
O passo 2 inicializa as variáveis PRODUTO (valor 1) e MULTIPLICADOR (valor de N = 3);
O passo 3 testa se o valor de MULTIPLICADOR (3) é maior ou igual a 1 (é maior) e
então entra no laço (passos 4 e 5);
O passo 4 executa a multiplicação (1 * 3) e atribui o resultado a PRODUTO (3)
O Passo 5 subtrai 1 do valor MULTIPLICADOR, que passa a ter o valor 2,
e retorna ao passo 3 – este testa a condição (2 ≥ 1) e entra no laço outra vez;

ii) Quando o valor de MULTIPLICADOR é igual a Zero, o algoritmo pula para o passo 6.

e) Um aspecto importante é o rastreamento das variáveis de controle do laço:

Repetição	PRODUTO	MULTIPLICADOR
0*	1	2
1	3	2
2	6	1
3	6	0

* Repetição zero: antes da entrada no laço

f) Exemplo: Algoritmo FAT_N para N = 6

Repetição	PRODUTO	MULTIPLICADOR
0	1	6
1	6	5
2	30	4
3	120	3
4	360	2
5	720	1
6	720	0

i) Observação: pode-se alterar a condição do laço para Enquanto Multiplicador > 1
pois a multiplicação por 1 do passo 5 não altera o resultado;

ii) Cuidados especiais devem ser tomados na definição das condições do primeiro e último passos da estrutura Repita, onde os erros ocorrem com mais freqüência;
(1) O que aconteceria se o valor de entrada N fosse igual a zero? Está correto?

8. 2. Laços Controlados por Entrada

a) Utilizado para repetir os passos para um número determinado ou indeterminado de dados de entrada;

b) Exemplo: Algoritmo RELATÓRIO: lê o nome do aluno, as notas de laboratório, exame intermediário e exame final e calcula a média final do aluno. Algoritmo original:

```
1. [Ler dados de entrada]
        Leia(NOME, LABORATORIO, PROVA_PARCIAL, EXAME_FINAL)
2. [Calcular média]
        MEDIA ← 0.2*LABORATORIO+0.3*PROVA_PARCIAL+0.5*EXAME_FINAL
3. [Imprimir resultados]
        Escreva(´Nome: ‘, NOME)
        Escreva(‘Nota do Laboratório: ‘, LABORATORIO)
        Escreva(´Nota da Prova Parcial: ‘, PROVA_PARCIAL)
        Escreva(‘Nota do Exame Final: ‘, EXAME_FINAL)
        Escreva(‘Média Final: ‘, MEDIA)
4. [Terminar]
        Saída
```

c) Para ler os dados de todos os alunos da classe e calcular as respectivas médias finais, utiliza-se a estrutura Repita. O problema agora é quando parar o algoritmo. Apresentaremos três formas de resolver esse problema:

i) Registro inicial indicando quantos dados devem ser lidos (número de alunos);
ii) Registro final indicando que não há mais dados;
iii) Estrutura que termina automaticamente quando não há mais dados para leitura

d) Entrada controlada por Contador (I): registro adicional no início do conjunto de dados:

```
1. [Ler contador de dados (número de registros a processar)]
        Leia(ESTUDANTES)
2. [Inicializar contador de registros processados]
        CONTADOR ← 0
3. [Iniciar repetição controlada por Contador]
        Repita até passo 7 enquanto CONTADOR < ESTUDANTES
4. [Ler os dados (registro) do próximo aluno]
        Leia(NOME, LABORATORIO, PROVA_PARCIAL, EXAME_FINAL)
5. [Calcular média]
        MEDIA ← 0.2*LABORATORIO+0.3*PROVA_PARCIAL+0.5*EXAME_FINAL
6. [Imprimir resultados]
        Escreva(´Nome: ‘, NOME)
        Escreva(‘Nota do Laboratório: ‘, LABORATORIO)
        Escreva(´Nota da Prova Parcial: ‘, PROVA_PARCIAL)
        Escreva(‘Nota do Exame Final: ‘, EXAME_FINAL)
        Escreva(‘Média Final: ‘, MEDIA)
7. [Atualizar o contador de registros processados
        CONTADOR ← CONTADOR + 1
4. [Terminar]
        Saída
```

Analise cuidadosamente o laço desse algoritmo, considerando os valores assumidos pela variável CONTADOR.

e) Entrada controlada por Sentinela (II): registro adicional no final do conjunto de dados:

```
1. [Ler primeiro registro de dados]
        Leia(NOME, LABORATORIO, PROVA_PARCIAL, EXAME_FINAL)

2. [Iniciar repetição controlada por Sentinela]
        Repita até passo 5 enquanto NOME ≠ FIM      (assume-se que não há nome FIM)

3. [Calcular média]
        MEDIA ← 0.2*LABORATORIO+0.3*PROVA_PARCIAL+0.5*EXAME_FINAL

4. [Imprimir resultados]
        Escreva(´Nome: ‘, NOME)
        Escreva(‘Nota do Laboratório: ‘, LABORATORIO)
        Escreva(´Nota da Prova Parcial: ‘, PROVA_PARCIAL)
        Escreva(‘Nota do Exame Final: ‘, EXAME_FINAL)
        Escreva(‘Média Final: ‘, MEDIA)

5. [Ler os dados (registro) do próximo aluno]
        Leia(NOME, LABORATORIO, PROVA_PARCIAL, EXAME_FINAL)

6. [Terminar]
        Saída
```

Analise cuidadosamente o laço desse algoritmo

i) O fim dos dados é indicado por um valor NOME = FIM;
ii) Para considerar um caso onde não existem dados (o primeiro registro é FIM) utiliza-se o recurso de ler o primeiro registro antes de se entrar no laço;
iii) O registro sentinela deve incluir todos os dados, mesmo que vazios: (FIM,0,0,0);

f) Entrada controlada por Fim de Arquivo (III): sem registro inicial ou final:

```
1. [Iniciar repetição controlada por Fim de Arquivo]
        Repita até passo 4 enquanto existirem dados para ler

2. Ler os dados (registro) do próximo aluno]
        Leia(NOME, LABORATORIO, PROVA_PARCIAL, EXAME_FINAL)
        Se não existirem mais dados
        Então Saída

3. [Calcular média]
        MEDIA ← 0.2*LABORATORIO+0.3*PROVA_PARCIAL+0.5*EXAME_FINAL

4. [Imprimir resultados]
        Escreva(´Nome: ‘, NOME)
        Escreva(‘Nota do Laboratório: ‘, LABORATORIO)
        Escreva(´Nota da Prova Parcial: ‘, PROVA_PARCIAL)
        Escreva(‘Nota do Exame Final: ‘, EXAME_FINAL)
        Escreva(‘Média Final: ‘, MEDIA)
```

Analise cuidadosamente o laço desse algoritmo

i) Esse método depende dos recursos da linguagem de programação específica;
ii) O controle “fim de arquivo” deve ser fornecido pela própria linguagem;
iii) Observe que o comando “saída” não é o último passo do algoritmo.

8.3. Laços Contados

a) Utilizado quando se deseja executar o algoritmo um número fixo de vezes:
Repita “intervalo do laço” para “valores da variável de controle do laço”;
   i) Repita até o passo 11 para ID = 1, 2, ..., 10
   ii) Repita para VAL = 1, 2, ..., N
   iii) Repita até o passo 8 para INDEX = -9, -8, ..., -2
   iv) Repita para CONTADOR = 2, 4, 6, ..., 18

b) A Variável de controle do laço determina o número de vezes que o laço é executado;
   i) A cada execução do laço, essa Variável tem seu valor ajustado automaticamente;

c) Os valores dessa variável são especificados como parte do comando de repetição;
   i) O laço termina quando a lista de números tiver sido totalmente utilizada;

d) Exemplo: algoritmo para imprimir as 20 primeiras potências de 2:

```
1. [Calcular e imprimir a potência de N]                POTENCIA    inteiro
        Repita para POTENCIA = 1, 2, ..., 20
                Escreva('Potência de', POTENCIA, '=', 2 ↑ POTENCIA)
2. [Terminar]
        Saída
```

   i) Quando o comando de repetição está incluído no mesmo passo que o comando de controle, os passos devem ser recuados em relação ao controle.

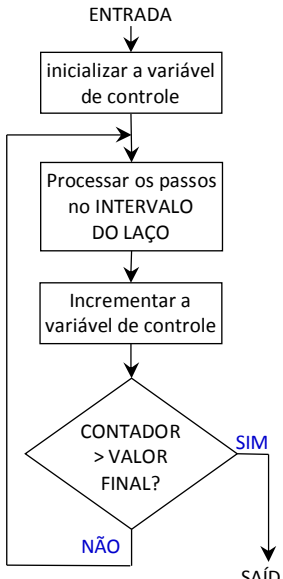
e) Generalizando o algoritmo para as primeiras N potências de 2:

```
1. [Ler número de potências requeridas]                 N           inteiro
        Leia(N)                                         POTENCIA    inteiro
2. [Calcular e imprimir a potência de N]
        Repita para POTENCIA = 1, 2, ..., N
                Escreva('Potência de', POTENCIA, '=', 2 ↑ POTENCIA)
2. [Terminar]
        Saída
```

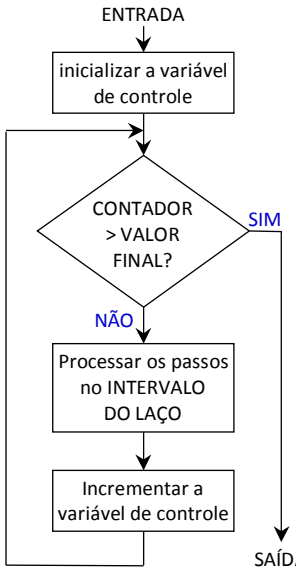
f) Resumo: define-se um valor inicial, um valor final e o valor do incremento do contador;

g) O teste [contador : valor final] é feito antes ou após a execução dos comandos do intervalo de repetição (laço) dependendo da linguagem de programação;

TESTE APÓS A EXECUÇÃO DOS COMANDOS
- passos executados no mínimo uma vez -



TESTE ANTES DA EXECUÇÃO DOS COMANDOS
- possibilidade de nenhum passo executado -



h) Exemplo: Algoritmo FATORIAL utilizando a estrutura Repita

```
1. [Ler N]
        Leia(N)
2. [Inicializar Produto]
        PRODUTO ← 1
3. [Calcular Fatorial]
        Repita para MULTIPLICADOR = N, N-1, ..., 1
                PRODUTO ← PRODUTO * MULTIPLICADOR
6. [Imprimir resultados]
        Escreva('Fatorial de', N, '=', PRODUTO)
7. [Terminar]
        Saída
```

N	inteiro
PRODUTO	inteiro
MULTIPLICADOR	inteiro

8.4. Laços Encaixados

a) Como na estrutura Se-Então-Senão, é possível encaixar estruturas Repita;
   i) A construção interna deve ser inteiramente embutida na construção externa;

b) Exemplo: algoritmo FATORIAL para diversos valores de N:

```
1. [Iniciar laço de leitura]
        Repita até passo 5 enquanto existirem dados para ler
2. [Ler o próximo valor de N]
        Leia(N)
        Se não existirem mais dados
        Então Saída ──────────────────────────► SAÍDA
3. [Inicializar Produto para o atual valor de N]
        PRODUTO ← 1
4. [Calcular Fatorial para o atual valor de N]
        Repita para MULTIPLICADOR = N, N-1, ..., 1
                PRODUTO ← PRODUTO * MULTIPLICADOR
5. [Imprimir resultados]
        Escreva('Fatorial de', N, '=', PRODUTO)
```

N	inteiro
PRODUTO	inteiro
MULTIPLICADOR	inteiro

Analise cuidadosamente os laços desse algoritmo

   i) O intervalo do laço externo vai do passo 2 ao passo 5;
   ii) O intervalo do laço interno está contido no passo 4;
   iii) O algoritmo termina quando a leitura de N retorna a mensagem "Fim de Arquivo";
      (1) "Fim de Arquivo" é uma mensagem específica de linguagens de programação.

8.5. Sumário

a) Conceito de Laço: repetição controlada por condição:
   i) Laço Condicional: condição expressa na cláusula “Enquanto”;
      (1) Risco maior de “loops” infinitos;
   ii) Laço Contado: condição expressa no valor do contador (número de repetições);
      (1) Controle por Número de repetições fornecido;
      (2) Controle por Final de Arquivo;
      (3) Laços Contados (valor inicial, valor final, incremento)

b) Laços embutidos: aninhamento de laços
   i) Laço interno é executado completamente a cada passagem do laço externo
   ii) Para maior clareza, recomenda-se no máximo três níveis de embutimento;

c) Comando “Vá Para” (GO TO):
   i) Comando controvertido que não é muito utilizado em programação estruturada;
   ii) Grande risco de confusão e dificuldade de rastreamento da execução do programa;
   iii) Recomenda-se evitar esse comando ou somente utilizá-lo em situações excepcionais.

EXERCÍCIOS

1. Em cada um dos seguintes segmentos de algoritmo, indicar se o laço termina ou não e o valor da variável TOTAL. Se não termina, porquê? Todas as variáveis são inteiras.

ALGORITMO		TOTAL	Termina?	Porquê?	Rastreamento
a)	1. CONTADOR ← 0 TOTAL ← 0 2. Repita enquanto CONTADOR ≥ 0 TOTAL ← TOTAL + 2				P T C 1 2 3 4 5 6 7 8 9 10 11 12
b)	1. CONTADOR ← 0 TOTAL ← 0 2. Repita enquanto CONTADOR ≤ 10 TOTAL ← TOTAL + 2 CONTADOR ← CONTADOR + 1				
c)	1. TOTAL ← 0 2. Repita para INDEX = 1, 2, ..., 10 TOTAL ← TOTAL + 1 INDEX ← INDEX - 1				

2. Em cada um dos seguintes segmentos, dê o valor que será impresso para a variável VAR. Assumir que todas as variáveis são inteiras

a)	1. VAR ← 0 2. Repita para INDEX = 1, 2, ..., 10 VAR ← VAR + 1 3. Escreva(VAR)	c)	1. VAR ← 0 2. Repita o passo 3 para INDEX1 = 1, 2, ..., 15 3. Repita para INDEX2 = 1, 2, ..., 8 VAR ← VAR + 1 4. Escreva(VAR)
b)	1. VAR ← 0 2. Repita para INDEX = 4, 8..., 36 VAR ← VAR + 1 3. Escreva(VAR)	d)	1. VAR ← 0 2. Repita para INDEX = 10, 9, ..., 1 VAR ← VAR + 1 INDEX ← INDEX + 1 3. Escreva(VAR)

	Passo	1	2	3	4	5	6	7	8	9	10	11	
a)	INDEX	-	1	2	3	4	5	6	7	8	9	10	**a)**
	VAR	0											
b)	INDEX	-	4	8	12	16	20	24	28	32	36	**b)**	
	VAR	0											
c)	INDEX	-	1	2	3	4	5	6	7	8	INDEX * 15	**c)**	
	VAR	0											
d)	INDEX	-											**d)**
	VAR	0											

OBS: Tabela para cálculo do valor da variável VAR

3. Um programador está preocupado em relação a seu desempenho num curso de computação. Em seu primeiro programa ele cometeu um erro; em seu segundo, dois erros; no terceiro, quatro erros; assim por diante. Ele está cometendo, por programa, duas vezes o número de erros que cometeu no programa anterior. O curso dura treze semanas, com dois problemas por semana. Preparar um algoritmo para calcular o número de erros que este programador presume cometer em seu programa final.

4. Preparar um algoritmo para calcular a soma da seguinte série de 100 termos:

$$1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{6} + \frac{1}{8} - \frac{1}{10} + \frac{1}{12} - \ etc.$$

5. O antigo filósofo grego Zeno é, talvez, melhor conhecido pelo paradoxo de Aquiles e a tartaruga. Aquiles e a tartaruga disputam uma corrida. Aquiles corre dez vezes mais rápido do que a tartaruga, porém a tartaruga tem 100 metros de vantagem. Pode-se argumentar que Aquiles nunca ultrapassará a tartaruga, pois, quando ele atingir o ponto onde a tartaruga estava, ela já estará um pouco à frente. Projetar um algoritmo para ler a velocidade da tartaruga e calcular o tempo no qual Aquiles ultrapassará a tartaruga.

6. Os regulamentos de pesca do Estado de São Paulo impõem um limite no peso total de pesca de um dia. Suponha que você planeje levar seu computador portátil em sua próxima pescaria e deseje um programa para calcular quando você excedeu seu limite. Preparar um algoritmo que leia o limite diário (em kg) e, então, leia os valores de entrada um por um (os pesos dos peixes à medida que são apanhados) e imprima uma mensagem no ponto quando este limite é excedido. Um peso de zero indica o fim de entrada. Após o registro de cada peixe, o algoritmo deve imprimir o peso total de peixes obtido até aquele ponto.

7. Um número é, por definição, primo se ele não tem divisores, exceto 1 e ele próprio. Preparar um algoritmo para ler um número e determinar se ele é ou não um número primo.

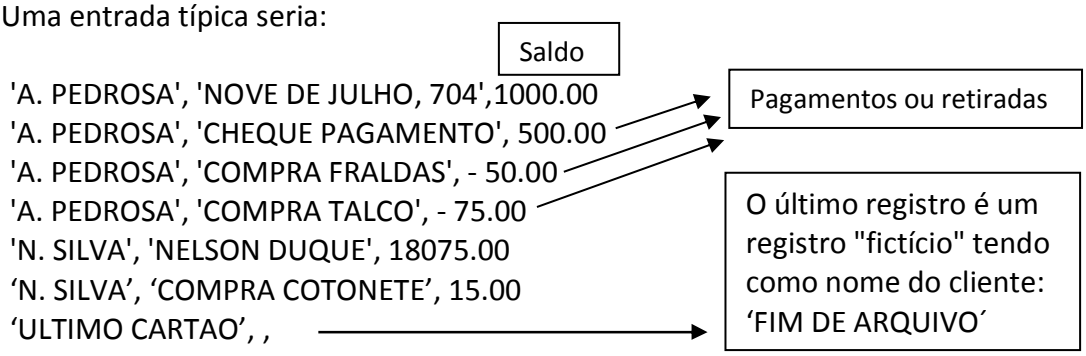
8. Um motorista acaba de retomar de um feriado prolongado. Em cada parada de reabasteci-mento ele registrou a leitura de seu odômetro e a quantidade de gasolina comprada (suponha que ele tenha enchido o tanque cada vez). Além disso, suponha também que ele tenha enchido o tanque antes de partir e imediatamente após retomar, registrando as leituras do odômetro em cada posto. Preparar um algoritmo para ler em primeiro lugar o número total de reabastecimentos feitos (incluindo o primeiro e o último) e, a seguir, os dados registrados relativos à compra de gasolina e calcular:
   (a) A quilometragem obtida por litro de gasolina entre cada par de paradas de reabastecimento.
   (b) A quilometragem obtida por litro de gasolina em toda a viagem.

9. Suponha que cada time de futebol, participante do Campeonato Brasileiro, tenha uma lista oficial de 23 jogadores. Os times são obrigados a preparar um registro para cada um de seus jogadores com o formato:

   nome do jogador, peso, idade

   Os dados dos quarenta times são coletados e enviados à CBF para análise. Um único conjunto de registros é preparado, com os registros agrupados por time individual; isto é, os primeiros 23 registros são do time 1, os próximos 23 do time 2 e assim por diante. Preparar um algoritmo para ler este conjunto de registros e calcular a seguinte estatística:
   (a) O peso médio e a idade média para cada um dos quarenta times.
   (b) O peso médio e a idade média de todos os participantes da Taça de Ouro.

10. Uma empresa VENDE TUDO S.A. utiliza um computador para preparar relatórios aos seus clientes. Para cada cliente um conjunto de registros de dados é preparado, contendo informações de seus pagamentos e de suas compras no mês. Os dados para cada cliente iniciam com um registro especial que contém seu nome, endereço e saldo dos meses anteriores. Este registro é seguido pelos registros de transação, contendo o nome do cliente, a descrição da transação e seu valor. O valor das compras terá um sinal negativo. Uma entrada típica seria:



   Preparar um algoritmo para produzir um relatório da situação de cada cliente. Este relatório teria a forma:

   VENDE TUDO S.A.
   SQN 203 Bloco B
   BRASILIA

   PARA: A. PEDROSA
   NOVE DE JULHO 704

ITEM	PAGAMENTOS	RETIRADAS	TOTAL
SALDO			1000.00
CHEOUEPAGAMENTO	500.00		1500.00
COMPRAS FRALDAS		50.00	1450.00
COMPRAS TALCO		75.00	1375.00
TAXA SERVICO		.50	1374.50
JUROS PAGOS	3.75		1378.25

   A taxa de serviço deve ser calculada para cada cliente como 0.25 por nota fiscal de cada compra. Os juros devem ser calculados pelo algoritmo à razão de 1 % somente para os valores que estão acima de 1000.00. Um registro de um novo cliente é detectado por uma mudança de nome.

9. VETORES E CONJUNTOS

a) Uma variável só pode ter um valor em um determinado momento;
b) Um Vetor é um conjunto monodimensional de variáveis;
c) O Índice é a posição da variável dentro do vetor

9.1. O VETOR COMO UMA ESTRUTURA DE DADOS

a) A escolha de itens de dados é um passo fundamental na definição e solução do problema;
b) Estrutura de dados é uma maneira particular de organização dos dados;
c) Estruturas primitivas: o computador tem conjunto de instruções para manipulá-las:
    i) inteiro, real, cadeia, lógico, ponteiro;
d) Estruturas não-primitivas: poucos computadores têm estruturas para manipulá-las:
    i) números complexos: U = x + yi ; V = a + bi onde i = $\sqrt{-1}$; x, y, a, b são números reais:

$$U \oplus V = (x + a) + (y + b)i$$
$$U \ominus V = (x - a) + (y - b)i$$
$$U \circledast V = (x * a - y * b) + (x * b + y * a)i$$
$$U \oslash V = \frac{(x * a + y * b) + (y * a - x * b)i}{(a * a + b * b)}$$

Onde os operandos indicam as operações complexas

    ii) Operações com números complexos podem ser efetuadas com operadores aritméticos ordinários (soma, subtração, etc.) em uma sequência de operações.
e) Estrutura de notas de uma classe: lista com nome do aluno respectiva nota (matriz n x 2)
    i) Exemplo: algoritmo para imprimir nome e nota dos alunos com média acima da média de uma classe com cinco alunos:

```
1. [Ler a relação de nomes e notas]
     Leia(NOME1, NOTA1, NOME2, NOTA2, NOME3, NOTA3, NOME4, NOTA4,NOME5, NOTA5)

2. [Calcular a média da turma]
     MEDIA ← (NOTA1 + NOTA2 + NOTA3 + NOTA4 + NOTA5) / 5.0

3.  [Verificar se a nota do estudante é maior que a média da turma e imprimir]
     Se NOTA1 > MEDIA Então Escreva(NOME1)
     Se NOTA2 > MEDIA Então Escreva(NOME2)
     Se NOTA3 > MEDIA Então Escreva(NOME3)
     Se NOTA4 > MEDIA Então Escreva(NOME4)
     Se NOTA5 > MEDIA Então Escreva(NOME5)

4. [Terminar]
     Saída
```

    ii) Como ficaria o algoritmo acima para uma classe de 100 alunos? E para 10.000?
f) Uma variável indexada é composta pelo nome da variável mais um índice:

NOME[i] NOTA[i]

g) Esse índice varia de 1 a N, onde N é o número total de variáveis em questão;
    i) NOME[5] é a variável que está na posição 5 da lista de variáveis;
h) Um Vetor é composto por um número determinado de variáveis indexadas, que podem ser manipuladas por laços controlados por variável

i) O algoritmo anterior passa a tomar a seguinte forma, utilizando-se variáveis indexadas NOME[i] e NOTA[i] - considera classe de 100 alunos:

```
1. [Inicializar variáveis]
     N ← 100
2. [Ler a relação de nomes e notas]
     Repita para i – 1, 2, ..., N
       Leia(NOME[i], NOTA[i])
3. [Calcular a média da turma]
     SOMA ← 0.0
     Repita para i = 1, 2, ..., N
       SOMA ← SOMA + NOTA[i]
     MEDIA ← SOMA / N
4.  [Verificar se a nota do estudante é maior que a média da turma e imprimir]
     Repita para i = 1, 2, ..., N
       Se NOTA[i] > MEDIA Então Escreva(NOME[i])
5. [Terminar]
     Saída
```

Variáveis		
N	inteiro	número de alunos
NOME[i]	cadeia	nome do aluno i
NOTA[I]	real	nota do aluno i
SOMA	real	soma das notas
MEDIA	real	média da turma

A única mudança para uma classe de 1000 alunos é no valor da variável N

9.2. OPERANDO SOBRE VETORES

a) Vetor: conjunto de elementos monodimensional X[1], X[2], ..., X[n]
   i) Em algumas linguagens, o limite inferior do índice pode ser diferente de 1: 0, -5, etc.;
   ii) O valor do índice pode ser calculado durante a execução do programa;

b) Cada elemento do vetor deve ser inicializado individualmente (quando assim exigido):

```
Repita para i = 0, 1, ..., 9
  A[i] ← 0
```

Utilizaremos a seguinte formulação: A ← 0

c) Para leitura de valores e atribuição ao vetor A:

```
Repita para i = 0, 1, ..., 9
  Leia(A[i])
```

Utilizaremos a seguinte formulação: Leia(A)

d) Para a saída de um vetor A:

```
Repita para i = 0, 1, ..., 9
  Escreva(A[i])
```

Utilizaremos a seguinte formulação: Escreva(A)

Muitas linguagens de programação permitem essa generalização

   i) Assume-se que o primeiro valor (inicializado / lido / impresso) é referente ao primeiro elemento do vetor e assim sucessivamente.

e) Exemplo: cálculo do desvio padrão D dos elementos de um vetor x:

$$D = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{N-1}}$$

Algoritmo Desvio Padrão

```
1. [Ler o tamanho do vetor]
      Leia(N)
2. [Ler o vetor]
      Leia(X)
3. [Calcular a média]
      SOMA ← 0.0
      Repita para i = 1, 2, ..., N
          SOMA ← SOMA + X[i]
      MEDIA ← SOMA / N
4.  [Calcular o desvio padrão]
      S ← 0.0
      Repita para i = 1, 2, ..., N
          S ← S + (X[i] – MEDIA)↑2
      D ← SQRT(S/(N - 1)
5. [Sair resultados]
      Escreva(D)
6. [Terminar]
      Saída
```

Variáveis

N	inteiro	nº de elementos do vetor
X[i]	real	Vetor X
SOMA	real	soma dos elementos de X
MEDIA	real	média dos elementos de X
S	real	somatória $(x_i - \bar{x})^2$
D	real	desvio padrão

f) Um conceito importante é que o índice pode ser calculado no decorrer do programa. Esse conceito será ilustrado no seguinte exemplo:

O Centro Técnico Aeroespacial mantém registros pluviométricos anuais (em centímetros) de 86 estações da região de secas do Nordeste. O nome da estação e a quantidade total de chuva registrada nesta estação no último ano estão armazenados em registros de banco de dados. Preparar algoritmo para determinar o número de estações cujo registro de quantidade de chuva esteja nos grupos especificados na Tabela 4.1.

Grupo	Intervalo
1	0 a 9
2	10 a 19
3	20 a 29
4	30 a 39
5	40 a 49
6	50 a 59
7	60 a 69
8	70 a 79
9	80 a 89
10	90 a 99

Assumimos que nenhuma estação recebe mais que 99 centímetros de chuva em 1 ano.

Serão utilizados dois métodos: o da “força bruta” e o do “cálculo do índice”.

Variáveis:

CLASSE	vetor	intervalo de classe da chuva – 10 posições
NOME_ESTACAO	cadeia	nome da estação
CHUVA	inteiro	quantidade de chuva em cm
POSICAO	inteiro	posição do índice referente a uma quantidade de chuva

Considerar método de fim de arquivo.

```
1. [inicializar contador de intervalos de classe]
     CLASSE ← 0                                   (inicializa todas as posições com o valor 0)
2. [Processar os dados]
     Repita até o passo 4 enquanto existirem registros
3. [Ler estatística da estação]
     Leia(NOME_ESTACAO, CHUVA)                    (o nome da estação é lido mas não é utilizado)
     Se não existirem mais dados de entrada
     Então  Escreva('GRUPO', 'NUMERO DE ESTAÇÕES')
            Repita para i = 1, 2, ..., 10
              Escreva( i, CLASSE(i))
     Saída
4. [Atualizar contador de classe apropriado]
     Se 0 ≤ CHUVA  E  CHUVA ≤ 9  Então CLASSE[1] ← CLASSE[1] + 1
     Se 10 ≤ CHUVA  E  CHUVA ≤ 19  Então CLASSE[2] ← CLASSE[2] + 1
     ...
     Se 90 ≤ CHUVA  E  CHUVA ≤ 99  Então CLASSE[10] ← CLASSE[10] + 1
```

Como alternativa para o passo 4 podemos elaborar uma equação para a posição do índice:

```
4. [Atualizar contador de classe apropriado]
     POSICAO ← TRUNC(CHUVA/10) + 1
     CLASSE(POSICAO) ← CLASSE(POSICAO) + 1
```

A questão passa a ser como montar uma equação que determina a posição do índice a partir do volume de chuva observado.

EXERCÍCIOS

1. Para um vetor A de *n* números reais, formular um algoritmo que determine o maior e o segundo maior elemento deste vetor. Assumir que os valores são distintos.

2. Dado um vetor A de *n* números reais, obter a maior diferença entre dois elementos consecutivos desse vetor.

3. Repetir o exercício 2 e obter a menor diferença entre dois elementos consecutivos.

4. Formular um algoritmo para obter as seguintes estatísticas para um vetor X de *n* elementos:

   Desvio médio (DM) $DM = \frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x}|$ onde $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

   Raiz média quadrática (RMQ) $RQM = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$

   Média Harmônica (MH) $MH = \frac{n}{\sum_{i=1}^{n}(\frac{1}{x_i})}$

   Amplitude total (AT) $AT = max.\{x_1, x_2, \dots, x_n\} - min.\{x_1, x_2, \dots, x_n\}$

   Média geométrica (MG) $MG = \sqrt[n]{x_1 * x_2 * \dots * x_n}$

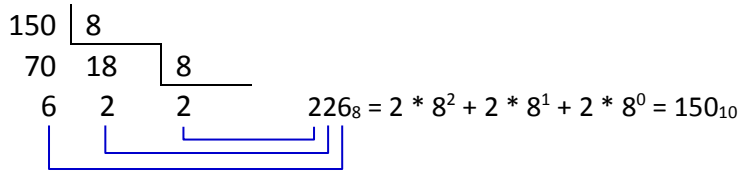
5. Preparar um algoritmo que leia um vetor A não ordenado de *n* inteiros e imprima o vetor na mesma sequência, ignorando valores em duplicata encontrados no vetor lido. O número de elementos restantes (*m*) é também solicitado. Por exemplo, dado o vetor de 10 inteiros:

A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
15	31	23	15	75	23	41	15	31	85

   o vetor resultante teria *m*=6:

A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
15	31	23	75	41	23	85			

6. Formular um algoritmo para converter representações de números inteiros decimais (base 10) em seus correspondentes octais (base 8), utilizando divisões sucessivas. Seja NUMERO o inteiro a ser convertido e BASE a base para a qual o inteiro será convertido (8, no caso). Por exemplo, para calcular a representação octal de 150, basta dividi-lo repetidamente por 8 e guardar o resto em ordem:



7. Dado um polinômio *p(x)* da forma:

   $p(x) = a_0x^n + a_1x^{n-1} + \dots + a_{n-1}x + a_n$

   onde $a_0, a_1, \dots, a_n$ são números reais, designando os coeficientes do polinômio, construir um algoritmo que leia *n* seguido dos coeficientes $a_0, a_1, \dots, a_n$ e de uma sequência de valores de *x*. Para cada valor de *x*, tabular o valor de *p(x)*.

8. Problema "distância-de-caracter" Examinar uma cadeia de n caracteres alfabéticos (entrada) e formar uma cadeia com valores numéricos, um número para cada caráter alfabético. Cada posição da cadeia resultante será ocupada por um número representando uma contagem dos caracteres que separam o caráter na posição correspondente do caráter similar mais próximo para a esquerda na cadeia de entrada. Nenhuma distância maior do que q será registrada. Qualquer caráter, sem correspondente à esquerda dentro de nove posições, terá zero na cadeia numérica resultante. Por exemplo, dada a cadeia de caracteres AABCDBEFFEABGBWB, a mesma daria como resultado a cadeia numérica 0100030013960202. Formular um algoritmo para esse problema. Para entrada, suponha que cada caráter esteja entre aspas. Por exemplo, a entrada da cadeia de caracteres citada acima teria a seguinte forma

   16 ← número de caracteres
   'A' 'A' 'B' 'C' 'D' 'B' 'E' 'F' 'F' 'E' 'A' 'B' 'G' 'B' 'W' 'B'

9. Durante a aplicação de revestimento de concreto em uma escavação numa mina de potássio, amostras de concreto eram recolhidas e testadas para resistência máxima. Os dados de profundidade do poço versus resistência do concreto foram perfurados em registros especificados abaixo:

   O primeiro registro contém a profundidade inicial de medida do poço e o número total de resultados de testes para pontos consecutivos e para baixo, com incrementos de um pé.

   Todos os registros a seguir contêm 10 resultados de teste por registro, exceto o último que pode conter menos de 10 resultados, dependendo do número total de medidas.

   O exemplo abaixo ilustra os registros de entrada:

   Profundidade inicial do poço ← 2318 → 28 → Número total de resultados de testes (nunca maior do que 100)

   4560 3920 4350 ...

   Cada registro contém 10 resultados consecutivos

   O último registro conteria 8 resultados se o número total de medida fosse 28

   Uma média móvel de oito resultados é usada como uma indicação da média da resistência do concreto e indicaria qualquer seção extremamente fraca. A média móvel é a média das leituras naquela profundidade e os próximos 7 pés abaixo. A média móvel para cada um dos últimos 7 pés é a média das leituras restantes. Obter um algoritmo para gerar uma tabela com as médias móveis, como indicado a seguir:

PROFUNDIDADE	RESULTADO TESTE	MÉDIAS MÓVEIS
2318	4560	4341
2319	3920	4256
...	...	...
2339	4820	4515
...	...	...
2345	4500	4500

   A ordem da média móvel, no caso 8, pode variar de aplicação para aplicação. Após obter a solução, generalizar o algoritmo de modo que somente um registro de dado deva ser mudado para alterar a ordem da média.

10. A seguinte fórmula fornece a variância dos valores num vetor x (x designa a média dos n valores do vetor):

$$\sigma_x^2 = \frac{1}{n-1}\left[\sum_{i=1}^{n}(x_i^2) \; - n(\bar{x})^2\right]$$

Preparar um algoritmo para ler os elementos de um vetor x e calcular a média e variância dos valores utilizando esta fórmula para variância. Você pode ver qualquer vantagem computacional na utilização desta fórmula ao invés da fórmula mais familiar dada abaixo?

$$\sigma_x^2 = \frac{1}{n-1}\left[\sum_{i=1}^{n}(x_i - \bar{x})^2\right]$$

11. Um importante problema na estatística é o da estimativa do valor de uma variável a partir do valor de outra variável. Duas variáveis que apresentam boas possibilidades de estimativa são ditas fortemente correlacionadas. O grau de correlação é determinado pelo coeficiente de correlação. Desejamos conduzir uma experiência para determinar o grau de correlação entre a média final de um estudante no 2º grau e seu desempenho no primeiro ano da Universidade. Para cada aluno do 1º ano é preparado um registro contendo dois valores reais: média do 2º grau (H) e média do primeiro ano universitário (F). Assumir que existem N estudantes envolvidos no estudo. Projetar um algoritmo para ler estes dados em dois vetores, H[i] e F[i], i = 1, 2, ..., N. Calcular então o coeficiente de correlação r utilizando a fórmula:

$$\mathrm{r} = \frac{\mathrm{N}\,\Sigma\mathrm{H[i]F[i]} \; - \; \Sigma\mathrm{H[i]}\;\Sigma\mathrm{F[i]}}{\sqrt{(\mathrm{N}\,\Sigma\mathrm{H[i]}^2 \; - \; (\Sigma\mathrm{H[i]})^2)\;(\mathrm{N}\,\Sigma\mathrm{F[i]}^2 \; - \; (\Sigma\mathrm{F[i]})^2)}}$$

Se o coeficiente de correlação exceder 0.85, uma mensagem deve ser impressa, informando que estas variáveis parecem ser fortemente correlacionadas.

12. Em qualquer experiência, existe um certo erro associado às medidas. Uma técnica conhecida como suavização pode ser utilizada para reduzir o efeito deste erro na análise dos resultados. Suponha que uma série de valores reais tenha sido registrada de N repetições de uma experiência. Estes valores são armazenados num vetor V. Antes da análise destes resultados experimentais, a seguinte operação simples de suavização é aplicada aos valores de V. Para cada valor (exceto o primeiro e o último, que não mudam), V_i é substituído por

$$\frac{\mathrm{V_{i-1}} \; + \; \mathrm{V_i} \; + \; \mathrm{V_{i+1}}}{3}$$

Preparar um algoritmo para ler as medidas iniciais e, então, imprimir os valores observados e os suavizados. Estes valores suavizados são armazenados num vetor separado SUAVE.

13. As companhias cinematográficas de Hollywood não têm imaginação para produzir filmes de monstros. Seguindo o sucesso de "Frankenstein", foram produzidas seqüências como "O filho de Frankenstein" e "A noiva de Frankenstein". Quando há pares de monstros em competição, tais como Frankenstein e Drácula, temos títulos adicionais, como "Frankenstein Encontra Drácula".

a) Desenvolver um algoritmo para gerar possíveis títulos de filmes de monstros a partir de uma lista de monstros populares. O algoritmo utilizará como "título-padrão" básico os seguintes quatro casos:

1. *MONSTRO1*
2. O FILHO DE *MONSTRO1*
3. A NOIVA DE *MONSTRO1*
4. *MONSTRO1* ENCONTRA *MONSTRO2*

Com este padrão particular, o algoritmo gerará todos os possíveis títulos a partir de uma lista de entrada de nomes de monstros.

b) Testar manualmente o algoritmo com o seguinte conjunto de dados. Mostrar claramente a ação do algoritmo e a saída que resulta dele.

'KING KONG'
'MOTHRA'
'GODZILLA'

Utilizar o método fim-de-arquivo para detectar o fim dos dados.

14. Uma imobiliária tem 25 vendedores entre seus empregados. Cada venda feita por um vendedor é armazenada num registro. Este registro contém

número vendedor, nome vendedor, valor de venda

O número de registros de entrada é desconhecido. O número de vendas pode variar de vendedor para vendedor. Por exemplo, um vendedor pode fazer 12 vendas e outro, 10. Os registros de entrada não estão em seqüência. Formular um algoritmo que gere o valor total de vendas por vendedor. O relatório deve ter a forma:

NUMERO VENDEDOR	NOME VENDEDOR	TOTAL
1	JOAO	50240.75
2	SUELY	71326.50
...	...	...
25	INES	21375.00

o método fim-de-arquivo deve ser utilizado para detectar o fim dos dados. Menções especiais devem ser dadas aos vendedores com os dois totais mais altos. Projetar o algoritmo de modo que os nomes destas duas pessoas sejam realçados.

15. Construir um algoritmo que gere um relatório anual de vendas. O relatório deve dar subtotais mensais de vendas e o total anual. Cada transação de venda é registrada em registros como segue:

valor da venda, número do mês

onde cada mês do ano é numerado de 1 a 12. O número de registros de entrada é desconhecido e estes registros não estão em seqüência. Utilizar o método fim-de-arquivo para detectar o fim dos dados.

10. CLASSIFICAÇÃO E PESQUISA COM VETORES

a) Classificação e Pesquisa: duas operações muito importantes em computação:
   i) Muito do tempo dos computadores é utilizado nessas operações;
   ii) Freqüentemente essa operação é realizada com vetores;
b) Classificação é arranjar os elementos do vetor em uma ordem seqüencial específica;
   i) A ordem do arranjo é determinada por um critério bem definido;
   ii) Elementos numéricos: ordem crescente ou decrescente;
   iii) Elementos não numéricos: ordem ascendente ou descendente;
c) O elemento sobre o qual a classificação é feita é denominado "Chave";
   i) Exemplo: ordenar a sequência 73, 65, 52, 24, 83, 17, 35, 96, 41;
   ii) Crescente: 9, 17, 24, 35, 41, 52, 65, 73, 83, 96;
   iii) Decrescente: 96, 83, 73, 65, 52, 41, 35, 24, 17, 9;
d) Pesquisa envolve a exploração de um conjunto de itens a fim de localizar um item desejado;
   i) Exemplo: em uma relação de nomes, encontrar um nome determinado;
e) Existem diversos métodos de realizar essa pesquisa – alguns mais eficientes que outros;
   i) Os métodos melhores exigem que os dados estejam arranjados de forma particular;

10.1. Classificação por Seleção

a) Um dos processos mais simples de classificação:
   i) Inicia-se com o primeiro elemento, buscando o menor de todos;
      (1) O elemento encontrado é copiado para a primeira posição de um outro vetor;
      (2) Um valor fictício é copiado para essa posição do vetor original para, na próxima interação, ser ignorado pela busca;
   ii) Prossegue-se a partir do segundo elemento, repetindo o procedimento anterior (i)
      (1) Quando esse elemento é encontrado, ele é copiado para a segunda posição e o mesmo valor fictício é copiado para essa posição no vetor original;
      (2) Quando a busca encontra o valor fictício, ele é ignorado;
   iii) Repete-se a busca até esgotar o número de elementos do vetor a ser classificado;
   iv) Ao final do processo, tem-se um segundo vetor organizado em ordem crescente;
   v) O processo de busca de um valor é chamado de "passagem"
      (1) Um vetor K de *n* elementos requer *n* passagens para ser ordenado;
b) Um algoritmo para representar esse procedimento é o seguinte:

1. Repita o passo 2 um total de *n* vezes

2. examine cada elemento no vetor não classificado e coloque o menor deles na posição seguinte de um vetor parcialmente classificado. Ignore o valor fictício escolhido para substituir os valores já classificados no vetor original.

Variáveis

N	inteiro	n. de elementos vetor K
K	real	vetor a ser classificado
SAIDA	real	vetor classificado
INDICE_MIN	inteira	posição do menor elemento em um passo particular
PASSAGEM	inteira	nº da passagem em execução

Obs: 9999 é o valor fictício que substituirá o valor já classificado. Supõe-se que nenhum valor do vetor K assuma esse valor.

Algoritmo Classificação: dado um vetor K, contendo N elementos, classificar esse vetor em ordem crescente copiando os valores classificados em um vetor SAIDA:

```
1. [Ler o vetor K e o número de elementos N desse vetor]
      Leia(K, N)
2. [Executar as N passagens]
      Repita até o passo 5 para PAS = 1, 2, ..., N
3. [Inicializar o índice mínimo]
      INDICE_MIN ← 1
4. [Executar uma passagem e obter o elemento com o menor valor]
      Repita para I = 2, 3, ..., N
            Se K[I] < K[INDICE_MIN] E K[I] ≠ 9999          (valor fictício)
            Então INDICE_MIN ← I
5. [Copiar o menor elemento atual e mudar seu valor em K para 9999]
      SAIDA[PASS] ← K[INDICE_MIN]
      K[INDICE_MIN] ← 9999
6. [Terminar]
      Saída
```

c) Rastreamento do algoritmo Classificação:

	Número de Passagens (PAS)									
	1	2	3	4	5	6	7	8	9	10
1	73	73	73	73	73	73	73	(73)	9999	9999
2	65	65	65	65	65	65	(65)	9999	9999	9999
3	52	52	52	52	52	(52)	9999	9999	9999	9999
4	24	24	(24)	9999	9999	9999	9999	9999	9999	9999
5	83	83	83	83	83	83	83	83	(83)	9999
6	17	(17)	9999	9999	9999	9999	9999	9999	9999	9999
7	35	35	35	(35)	9999	9999	9999	9999	9999	9999
8	96	96	96	96	96	96	96	96	96	(96)
9	41	41	41	41	(41)	9999	9999	9999	9999	9999
10	(9)	9999	9999	9999	9999	9999	9999	9999	9999	9999

d) Deficiências desse processo:
   i) Utilização de vetor adicional para guardar os dados classificados;
   ii) A cada passo todos os elementos do vetor têm que ser examinados;
   iii) O valor fictício "9999" poderia criar problemas em algumas situações:
      (1) Em um caso geral, esse valor poderia ser um valor válido.

e) Processo alternativo que elimina todas essas deficiências (obs: vetor com *n* elementos);
   i) Inicia-se com o primeiro elemento até encontrar o menor valor;
   ii) Esse menor valor é permutado com o elemento inicial (primeiro);
   iii) Repete-se sucessivamente o procedimento iniciando-se com o próximo elemento;
   iv) Na passagem *n*-1 o último elemento já será o maior elemento do vetor;
      (1) Serão *n*-1 passagens com número de testes progressivamente menor.

f) Algoritmo Classificação otimizado:

```
1. [Ler o vetor K e o número de elementos N desse vetor]
      Leia(K, N)
2. [Executar as N -1 passagens]
      Repita até o passo 5 para PAS = 1, 2, ..., N-1
3. [Inicializar o índice mínimo]
      INDICE_MIN ← PAS
4. [Executar uma passagem e obter o elemento com o menor valor]
      Repita para I = PAS + 1, PAS + 2, ..., N
            Se K[I] < K[INDICE_MIN]
            Então INDICE_MIN ← I
5. [Permutar os elementos]
      Se INDICE_MIN ≠ PAS
      Então K[PAS] ⇔ K[INDICE_MIN]
6. [Terminar]
      Saída
```

Obs: o operador ⇔ indica:

```
TEMP ← K[INDICE_MIN]
K[INDICE_MIN] ← K[PAS]
K[PAS] ← TEMP
```

g) Rastreamento do algoritmo Classificação otimizado:

	Não classificado	Número de Passagens (PAS)							Classificado	
j	*K*[*j*]	1	2	3	4	5	6	7	8	9
1	73	9	9	9	9	9	9	9	9	9
2	65	65	17	17	17	17	17	17	17	17
3	52	52	52	24	24	24	24	24	24	24
4	24	24	24	52	35	35	35	35	35	35
5	83	83	83	83	83	41	41	41	41	41
6	17	17	65	65	65	65	52	52	52	52
7	35	35	35	35	52	52	65	65	65	65
8	96	96	96	96	96	96	96	96	73	73
9	41	41	41	41	41	83	83	83	83	83
10	9	73	73	73	73	73	73	73	96	96

i) O número de comparações necessárias é proporcional a n^2;

ii) O número máximo de permutações necessárias é n-1;

(1) Depende de o quanto os elementos do vetor inicial estão fora de ordem;

10.2. Pesquisa Básica

a) Pesquisa Linear: método/técnica mais direto(a);
   i) Consiste na verificação seqüencial de cada elemento do vetor até que o elemento desejado seja encontrado;

b) Algoritmo PESQ_LINEAR: dado um vetor K não classificado consistindo de n (n ≥ elementos, este algoritmo procura no vetor um elemento particular cujo valor é x. Suponha que todas as variáveis sejam do tipo real.

```
1. [Ler o vetor K e o número de elementos N desse vetor]
     Leia(K, N, X)

2. [Pesquisar o vetor]
     Repita para I = 1, 2, ..., N
       Se K[I] = X
       Então Escreva('Valor', X, 'está na posição, 'I')
       Saída

3. [Escrever mensagem de elemento não encontrado]
     Escreva('Valor', X, 'não encontrado no vetor K')

4. [Terminar]
     Saída
```

   i) Este método requer de 1 a N passagens (em média, N/2) para encontrar o elemento X;

c) Algoritmo PESQ_LINEAR modificado: uma outra forma de implementar o algoritmo

```
1. [Ler o vetor K e o número de elementos N desse vetor]
     Leia(K, N, X)

2. [Inicializar pesquisa]
     I ← 1
     K[N + 1] ← X

3. [Pesquisar o valor X no vetor K]
     Repita enquanto K[I] ≠ X
       I ← I + 1

4. [Testar o sucesso da pesquisa]
     Se I = N + 1
     Então Escreva('Valor', X, 'não encontrado no vetor K')
     Senão Escreva('Valor', X, 'está na posição, 'I')
     Saída

4. [Terminar]
     Saída
```

d) Pesquisa Binária: utilizada quando o vetor está classificado em ordem ascendente:

```
1. Repetir até o passo 3 enquanto o intervalo de pesquisa não ficar vazio;
2. Obter a posição do elemento central no intervalo atual de pesquisa;
3. Se o valor do elemento desejado é igual ao do elemento central
   Então a pesquisa é bem sucedida - sair
   Senão, reduza o intervalo de pesquisa à segunda metade do intervalo anterior
4. Pesquisa mal sucedida
```

e) Implementação do algoritmo Pesquisa Binária

```
1. [Ler o vetor K e o nº de elementos N de K]
      Leia(K, N, X)
2. [Inicializar pesquisa]
      BAIXO ← 1
      ALTO ← N
3. [Executar a pesquisa]
      Repita até passo 5 enquanto BAIXO ≤ ALTO
4. [Obter índice central do intervalo
      MEDIO ← (BAIXO + ALTO) / 2          (divisão de inteiros: resultado inteiro)
      MEDIO ← TRUNC( (BAIXO + ALTO) / 2 )  (caso geral para algumas linguagens)
5. [Comparar X com elementos de K]
         Se X < K[MEDIO]
         Então ALTO ← MEDIO - 1
         Senão  Se X > K[MEDIO]
                Então BAIXO ← MEDIO + 1
                Senão  Escreva('Valor', X, 'está na posição', MEDIO)
                       Saída
4. [Pesquisa mal sucedida
      Escreva('Valor', X, 'não encontrado no vetor X']
      Saída
```

Variáveis		
K	real	vetor K classificado
N	inteiro	nº de elementos do vetor
X	real	valor a ser encontrado
BAIXO	inteiro	índice inferior
ALTO	inteiro	índice superior

i) Rastreamento com o vetor ordenado 61, 147, 197, 217, 309, 448, 503

Pesquisa para 197			
Iteração	BAIXO	MEDIO	ALTO
1	1	7	4
2	1	3	4
3	3	3	3

Pesquisa para 503			
Iteração	BAIXO	MEDIO	ALTO
1	1	7	4
2	5	7	6
3	7	7	7

ii) Serão, em média, TRUNC($\log_2 n$) -1 e no máximo, TRUNC($\log_2 n$) +1 comparações

f) Exemplo de pesquisa binária envolvendo expedição de pacotes num sistema postal:

i) Para ser aceita, uma encomenda deve observar as seguintes limitações:

(1) Peso limite 18 kg

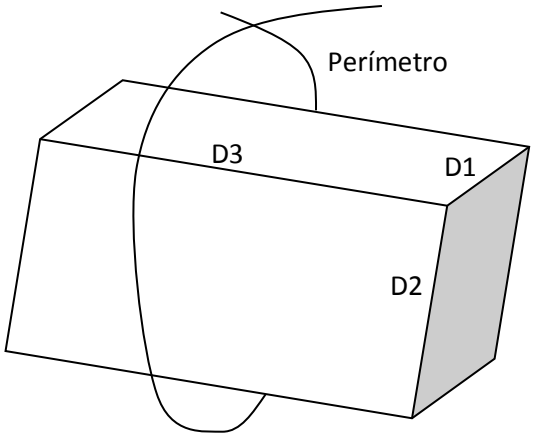
(2) Taxas da tabela abaixo aplicam-se a encomendas com comprimento, largura ou altura menores que 100 cm e comprimento e perímetros combinados não excedendo 200 cm

(3) Encomendas que excedem as dimensões acima são aceitas, exceto se uma dimensão exceder 200 cm e a combinação de comprimento e perímetro exceder 300 cm. Neste caso há uma sobretaxa de R$ 250,00

	Pacotes até e incluindo								
Peso (kg)	1	2	3	4	5	6	7	8	9
Custo (R$)	150	200	250	300	350	400	450	500	550

Peso (Kg)	10	11	12	13	14	15	16	17	18
Custo (R$)	600	650	700	750	800	850	900	950	1000

ii) O perímetro da encomenda é a circunferência do pacote ao redor de seus dois lados menores. Matematicamente:
P = 2 * (D1 + D2 + D3 – MAIOR)
onde MAIOR é a maior das três dimensões D1, D2 e D3:



iii) O algoritmo deve processar um arquivo de registros dos pacotes, um registro por pacote, expedidos durante a semana. Cada registro contém um número de transação seguido do peso do pacote e de suas três dimensões em uma ordem qualquer. O fim de arquivo é indicado por um registro com transação numero zero.

iv) O algoritmo deve imprimir:
(1) o número da transação, o peso e a tarifa postal para todos os pacotes aceitos e
(2) o número da transação, o peso e a dimensão para todos os pacotes rejeitados.

v) No final do relatório, deve imprimir:
(1) o número de pacotes aceitos e
(2) o número de pacotes rejeitados.

g) Iniciamos definindo dois vetores para armazenar os dados de PESO e CUSTO.
i) Inicialmente determinamos se o pacote pode ser aceito e em que condições.
ii) O custo total de cada encomenda pode ser determinado pesquisando-se o vetor PESO para localizar a faixa de peso da encomenda e o elemento correspondente no vetor CUSTO determinando o custo postal básico.
iii) Admitimos que o peso da encomenda, W, já tenha sido arredondado para cima, para o cálculo das taxas, por exemplo, um peso de 2,4 kg seria arredondado para 3 kg.
iv) Admitimos que as dimensões das encomendas sejam dadas em centímetros.

h) Estabelecemos, então, um algoritmo simplificado inicial em linguagem natural:

```
1. Ler uma descrição da encomenda
2. Repetir até passo 4 enquanto o último registro não for encontrado
3. Se o pacote é muito grande
   Então  rejeitar a encomenda e atualizar a estatística de rejeição
   Senão  Se a caixa é grande
          Então  lançar uma penalidade de R$ 250
          Senão  não lançar penalidade
          Pesquisar tabela postal para o custo postal básico
          Obter o custo postal total adicionando-se o custo da penalidade ao custo básico
          Aceitar a encomenda e atualizar a estatística de aceitação
4. Ler outra descrição de encomenda
5. Imprimir estatística de aceitação e de rejeição
```

i) Os passos mais simples são os indicados abaixo:
i) Passo 1: Entrada de dados: Leia(NUMERO, W, D1, D2, D3) (ver dicionário de dados)
ii) Passo 2: Laço de cálculo das encomendas: Repita até passo ..*n*.. enquanto NUMERO ≠ 0
iii) Passo 4: Entrada de dados dentro do laço: Leia(NUMERO, W, D1, D2, D3)
iv) Passo 5: Saída das estatísticas: Escreva('Pacotes aceitos:', NA, 'Pacotes rejeitados:', NR)

j) A seguir o desenvolvimento dos passos 3:

k) Definimos então as variáveis, seu tipo e sua descrição

Variáveis		
PESO	inteiro	vetor com a tabela de pesos de encomendas
CUSTO	real	vetor correspondente a PESO com a tabela de custos postais
NUMERO	inteiro	número de identificação do pacote
W	inteiro	peso da encomenda
D1	inteiro	primeira dimensão do pacote
D2	inteiro	segunda dimensão do pacote
D3	inteiro	terceira dimensão do pacote
MAIOR	inteiro	maior dimensão do pacote
PERIMETRO	inteiro	perímetro do pacote
PENALIDADE	real	penalidade associada às dimensões do pacote
CUSTO_POSTAL	real	custo de expedição da encomenda
NR	inteiro	número de pacotes rejeitados
NA	inteiro	número de pacotes aceitos
BAIXO	inteiro	limite inferior do segmento da tabela para pesquisa binária
MEDIO	inteiro	posição média do segmento da tabela sendo pesquisada
ALTO	inteiro	limite superior da tabela para pesquisa binária

l) Iniciamos pela determinação do lado maior e do perímetro do pacote:

```
MAIOR ← D1
Se MAIOR < D2 Então MAIOR ← D2
Se MAIOR < D3 então MAIOR ← D3
PERIMETRO ← 2 * (D1 + D2 + D3 – MAIOR)
```

m) Com o perímetro determinado, podemos verificar se a encomenda é aceita ou rejeitada.

i) Se é aceita,

(1) verifica-se se há penalidade;

(2) pesquisa-se a tabela de taxas postais para determinar o custo de expedição;

(3) atualiza-se a estatística de encomendas aceitas;

ii) Se é rejeitada, apenas atualiza-se o contador de encomendas rejeitadas.

```
Se W > 18 Ou (MAIOR + PERIMETRO) > 300 Ou MAIOR > 200
Então   Escreva ('Pacote rejeitado', NUMERO, W, D1, D2, D3)
        NR ← NR +1
Senão   Se MAIOR + PERIMETRO > 200 Ou MAIOR > 100
        Então   PENALIDADE ← 250.00
        Senão PENALIDADE ← 0.0
        Pesquise a tabela de taxas postais para o custo básico e imprima mensagem aceitação
```

n) Agora o algoritmo de pesquisa binária para determinar o custo básico e o custo total:

```
BAIXO ← 1
ALTO ← 18
Repita enquanto BAIXO ≤ ALTO
        MEDIO ← TRUNC((BAIXO + ALTO) / 2)
        Se W < PESO[MEDIO]
        Então   ALTO ← MEDIO – 1
        Senão   Se W > PESO[MEDIO]
                Então   BAIXO ← MEDIO + 1
                Senão   CUSTO_POSTAL ← PENALIDADE + CUSTO[MEDIO]      W = PESO[MEDIO]
                        Escreva(´Pacote Aceito', W, CUSTO_POSTAL)
                        NA ← NA + 1
                        BAIXO ← ALTO + 1
```

o) O algoritmo completo e detalhado: Dadas uma tabela de taxas postais, representadas pelos vetores PESO, CUSTO e as especificações de uma encomenda, que consistem em número de identificação (NUMERO), peso (W) e dimensões D1, D2, D3, este algoritmo gera o relatório já descrito. As variáveis MAIOR e PERIMETRO designam a maior dimensão e o perímetro da encomenda. PENALIDADE representa o custo de penalização para um pacote maior, e CUSTO_POSTAL significa o custo total de expedição da encomenda. NA e NR são usados para contar o número de encomendas aceitas e rejeitadas.

```
1. [Inicializar variáveis de contagem dos pacotes aceitos e rejeitados]
     NA ← 0
     NR ← 0

2. [Ler tabela de taxas postais]
     Repita para I = 1, 2, ..., 18
         Leia(PESO[I], CUSTO[I]

3. [Ler descrição da primeira encomenda]
     Leia(NUMERO, W, D1, D2, D3)

4. [Processar as encomendas]
     Repita até passo 8 enquanto NUMERO ≠ 0          (registro sentinela no fim de arquivo)

5. [Encontrar a maior dimensão da encomenda]
     MAIOR ← D1
     Se MAIOR < D2 Então MAIOR ← D2
     Se MAIOR < D3 Então MAIOR ← D3

6. [Calcular o perímetro da encomenda]
     PERIMETRO ← 2 * (D1 + D2 + D3 – MAIOR)

7. [Processar a encomenda]
     Se W > 18 Ou MAIOR + PERIMETRO > 300 Ou MAIOR > 200
     Então   Escreva('Pacote rejeitado', NUMERO, W, D1, D2, D3)
             NR ← NR + 1
     Senão   Se MAIOR + PERIMETRO > 200 Ou MAIOR > 100
             Então PENALIDADE ← 250.00
             Senão PENALIDADE ← 0.0
             [pesquisar tabela de taxas postais para o custo básico]
             BAIXO ← 1
             ALTO ← 18
             Repita enquanto BAIXO ≤ ALTO
                 MEDIO ← TRUNC((BAIXO + ALTO) / 2)
                 Se W < PESO[MEDIO]
                 Então   ALTO ← MEDIO -1
                 Senão   Se W > PESO[MEDIO]
                         Então   BAIXO ← MEDIO + 1
                         Senão   CUSTO_POSTAL ← PENALIDADE + CUSTO[MEDIO]
                                 Escreva('Pacote Aceito', W, CUSTO_POSTAL)
                                 NA ← NA + 1
                                 BAIXO ← ALTO +1
8. [Ler outra descrição de encomenda]
     Leia(NUMERO, W, D1, D2, D3)

9. [Imprimir sumário[
     Escreva('Pacotes aceitos:', NA, 'Pacotes rejeitados', NR)
     Saída
```

10.3. Intercalação e intercalação com classificação

a) Dois vetores de entrada classificado em ordem crescente, um vetor de saída classificado;

i) VETOR1: 13, 21, 29 VETOR2: 7, 28 VETORFINAL: 7, 13, 21, 28, 39

VETOR	1	2	3	4	5
1	13	21	39		
2	7	28			
3	7				
1	13	21	39		
2	7	28			
3	7	13			
1	13	21	39		
2	7	28			
3	7	13	21		
1	13	21	39		
2	7	28			
3	7	13	21	28	
1	13	21	39		
2	7	28			
3	7	13	21	28	39

b) Algoritmo Intercalação simples de dois vetores:

```
1. [Ler vetores A e B]
      Leia(A)
      Leia(B)

2. [Inicializar variáveis]
      I ← J ← K ← 1

3. [Determinar menor elemento correspondente]
   Repita enquanto I ≤ N  E  J ≤ M
      Se A[i] ≤ B[J]
      Então  C[K] ← A[I]
             I ← I + 1
             K ← K + 1
      Senão  C[K] ← B[J]
             J ← J + 1
             K ← K + 1

3. [Copiar os elementos não processados no vetor de saída]
   Se I > N
   Então  Repita para R = J, J+1, ..., M
          C[K] ← B[R]
          K ← K + 1
   Senão  Repita para R = I, I+1, ..., N
          C[K] ← A[R]
          K ← K + 1

4. [Terminar]
      Saída
```

Variáveis		
A, B, C	inteiro	vetores A, B e C
N	inteiro	nº elementos vetor A
M	inteiro	nº elementos vetor B
I, J, K, R	inteiro	índices dos vetores

c) Algoritmo Mala Direta: duas listas de mala direta, já classificadas em ordem alfabética, devem ser intercaladas e os nomes repetidos devem ser eliminados.

```
1. [Ler vetores A e B]
      Leia(LISTAA)
      Leia(LISTAB)
2. [Inicializar variáveis]
      I ← J ← K ← 1
3. [Comparar nomes e atualizar LISTAC]
   Repita enquanto I ≤ N   E   J ≤ M
      Se LISTAA[i] < LISTAB[J]
      Então   LISTAC[K] ← LISTAA[I]
              I ← I + 1
              K ← K + 1
              Se LISTAA[I] > LISTAB[J]
              Então   LISTAC[K] ← LISTAB[J]
                      J ← J + 1
                      K ← K + 1
              Senão   LISTAC[K] ← LISTAA[I]
                      I ← I + 1
                      J ← J + 1
                      K ← K + 1
3. [Copiar nomes restantes na lista nova]
   Se I > N
   Então  Repita para R = J, J+1, ..., M
          LISTAC[K] ← LISTAB[R]
          K ← K + 1
   Senão  Repita para R = I, I+1, ..., N
          LISTAC[K] ← LISTAA[R]
          K ← K + 1
4. [Terminar]
      Saída
```

Variáveis

LISTAA	cadeia	vetor com nomes A
LISTAB	cadeia	vetor com nomes B
N	inteiro	nº elementos vetor A
M	inteiro	nº elementos vetor B
LISTAC	cadeia	vetor mesclado A e B
I, J, K, R	inteiro	índices dos vetores

d) Para efetuar a intercalação de mais de dois vetores, procedemos à intercalação em pares sucessivamente até que reste apenas um vetor de saída.