



INSTITUTO FEDERAL DO ESPÍRITO SANTO

CURSO TÉCNICO DE INFORMÁTICA

LÓGICA DE PROGRAMAÇÃO

SUMÁRIO

1. PLANO DE CURSO	1
1.1. <i>Objetivo</i>	1
1.2. <i>Ementa</i>	1
1.3. <i>Metodologia</i>	2
1.4. <i>Avaliação</i>	2
1.5. <i>Referências</i>	2
1.6. <i>Recursos/Material de Apoio</i>	3
2. INTRODUÇÃO	3
2.1. <i>Conceitos</i>	3
2.2. <i>Métodos de Desenvolvimento de Algoritmos</i>	4
2.2.1. <i>Fluxogramas</i>	4
2.2.2. <i>Pseudocódigos Portugal</i>	5
3. COMPILADOR DE ALGORITMOS VISUALG	6
3.1. <i>Exercícios</i>	7
3.2. <i>Linearização de Expressões</i>	8
3.2.1. <i>Operadores Aritméticos</i>	8
3.2.2. <i>Operadores Aritméticos não convencionais</i>	9
3.2.3. <i>Operadores Relacionais</i>	9
3.2.4. <i>Operadores Lógicos</i>	9
3.2.4.1. <i>Tabela Verdade</i>	9
3.2.5. <i>Prioridade dos operadores</i>	10
3.2.6. <i>Modularização de Expressões</i>	10
3.2.7. <i>Expressões Lógicas</i>	10
3.2.8. <i>Exercícios</i>	11
3.3. <i>Primeiros Elementos da Linguagem</i>	12
3.3.1. <i>Identificadores</i>	13
3.3.1.1. <i>Regras para a nomeação de identificadores</i>	13
3.3.2. <i>Variáveis</i>	13
3.3.2.1. <i>Declaração de Variáveis</i>	15
3.3.2.2. <i>Tipos Primitivos de Dados</i>	16
3.3.3. <i>Sinal de Atribuição</i>	16

3.3.4. Comandos de E/S (Entrada/Saída)	16
3.3.5. Linhas de Comentário	17
3.3.6. Estruturas Seqüenciais - Construindo os Primeiros Algoritmos	18
3.3.7. Exercícios	20
3.4. Estrutura Condicional	24
3.4.1. Tipos de comandos condicionais	26
3.5. Testando o Algoritmo	29
3.6. Regras Práticas para Construir Algoritmos Legíveis	30
3.7. Exercício resolvido	31
3.8. Exercícios – Resolver os algoritmos na forma de pseudocódigo	33
3.9. Estrutura de Repetição	36
3.9.1. Repita..Até	36
3.9.2. Enquanto..Faca	37
3.9.3. Para..Faca	38
3.10. Exercícios de Algoritmos com Repetição	40
3.11. Variáveis Compostas Homogêneas	44
3.11.1. Variáveis Indexadas Unidimensionais (Vetores)	44
3.11.1.1. Declaração de Vetores	44
1.1.1.1. Atribuindo Valores a Elementos do Vetor	44
3.11.1.2. Exercícios sobre Vetores	45
3.11.1.3. Ordenação de Vetores	46
3.11.2. Variáveis Indexadas Multidimensionais (Matrizes)	47
3.11.2.1. Atribuindo Valores a Elementos da Matriz bidimensional	47
3.11.2.2. Exercícios	48
Anexo I – Lista de exercícios resolvidos	50
3.12. Exercícios da Seção 3.3.7	50
3.13. Exercícios da Seção 3.8	50
3.14. Exercícios da Seção 3.11	53
3.15. Exercícios da Seção 3.11.1.2	57
3.16. Exercícios da Seção 3.12.2.2	59

1. PLANO DE CURSO

Nome da disciplina: Lógica de Programação

Carga Horária Total: 90 horas/aulas

1.1. Objetivo

Esta disciplina tem por objetivo fornecer subsídios técnicos e práticos na construção de algoritmos e desenvolvimento de programas, utilizando uma ferramenta de compilação de algoritmos (Visualg 2.0, Scratch), e uma linguagem de programação científica, nos moldes das necessidades do curso técnico de Informática (Python).

1.2. Ementa

- ✓ Os conceitos de Lógica de Programação;
- ✓ Ferramentas de desenvolvimento:
- ✓ Pseudocódigo – PORTUGOL;
- ✓ Fluxogramas;
- ✓ Compilador de Algoritmos: Visualg 3.0
- ✓ Elementos do *Scratch*
- ✓ Tipos básicos de variáveis;
- ✓ Declaração de variáveis;
- ✓ Comandos de atribuição;
- ✓ Operadores Aritméticos, Lógicos e Relacionais;
- ✓ Comandos de E/S;
- ✓ Comandos Alternativos;
- ✓ Estruturas de repetição;
- ✓ Vetor;
- ✓ Matriz;
- ✓ Completo estudo dos comandos e operações na linguagem Python.

1.3. Metodologia

Aulas expositivas, aulas práticas em laboratório, dinâmica de grupo, seminários, estudos de caso;

1.4. Avaliação

Testes diagnósticos;

Provas formativas;

Trabalhos em grupo (teóricos);

Trabalhos em grupo (práticos em laboratório);

1.5. Referências

Bibliografia Básica (títulos, periódicos, etc.)					
Título/Periódico	Autor	Ed.	Local	Editora	Ano
Algoritmos e Lógica de Programação	Souza, Marco Antonio Furlan de/ Gomes, Marcelo Marques/ Concilio, Marcio Vieira Soares e Ricardo	2ª	Não Informado	Cengage	2012
Introdução à programação em Python	Menezes, N. N. C.	1ª	São Paulo	Novatec	2010
Aprendendo Python	Lutz M. & David Ascher	2ª	Porto Alegre	Bookman	2007
Python para desenvolvedores	Borges, L. E.	2ª	Rio de Janeiro	Edição do Autor	2010
Algoritmos: teoria e prática.	CORMEN, Thomas H.	2ª	Rio de Janeiro	Elsevier	2002
Bibliografia Complementar (títulos; periódicos etc.)					
Título/Periódico	Autor	Ed.	Local	Editora	Ano
Fundamentos da programação de computadores.	F. ASCENCIO, A. F. G. & CAMPOS, E. A. V.	2.ª	São Paulo	Pearson Education	2007
Algoritmos – Programação para iniciantes	VILARIM, G.	ni	Rio de Janeiro	Ciência Moderna	2004

1.6. Recursos/Material de Apoio

Quadro e pincel. Retroprojektor e transparências. Fontes bibliográficas e Apostilas. Laboratório contendo softwares necessários (Visualg 3.0, Scratch e Python).

2. INTRODUÇÃO

O COMPUTADOR é uma máquina capaz de fornecer a solução de um problema executando um roteiro com grande velocidade. Esse roteiro deve ser o mais detalhado possível, cercado todas as condições possíveis de acontecer, pois o computador não tem a capacidade de decidir por si só. Esse roteiro deve ser escrito em uma linguagem aceita pela máquina e que em sua fase final é chamado de PROGRAMA. A fase inicial, de definição dos passos a serem seguidos e dos desvios a serem considerados na solução de um problema, independe da linguagem de programação que será utilizada. Nesta fase o roteiro é chamado de ALGORITMO, e muitos métodos podem ser utilizados para defini-los.

Então a utilização de um computador para resolver um problema exige, antes de mais nada, que se desenvolva um algoritmo, isto é, que se faça descrição de um conjunto de comandos que, obedecidos, provocarão uma sucessão finita de ações que resultarão na resolução do problema proposto. Este algoritmo deve ser transmitido ao computador e armazenado em sua memória, para em seguida, ser posto em execução e conduzir o computador para a solução desejada.

A tarefa de programação envolve basicamente um processo de abstração de fatos e informações do mundo real. As informações devem ser modeladas em estruturas de dados enquanto as ações são transformadas em algoritmos. Deste modo devemos ter em mente que:

PROGRAMA = ALGORITMO + ESTRUTURA DE DADOS

2.1. Conceitos

O processamento de dados é baseado num número finito de operações com os dados, organizados em certas seqüências lógicas, para obtenção dos resultados desejáveis.

Segundo o Aurélio, Algoritmo é “um processo de cálculo, ou de resolução de um grupo de problemas semelhantes, em que se estipulam, com generalidade e sem restrições, as regras formais para a obtenção do resultado ou da solução do problema”.

Um algoritmo é uma norma executável para estabelecer um efeito desejável.

Na nossa vida cotidiana encontramos constantemente algoritmos, por exemplo: instruções de uso, receitas de cozinha, indicações de montagens, partituras musicais etc., que em sua grande maioria, apresentam o mesmo padrão de comportamento.

Num algoritmo (e um programa) devemos distinguir claramente dois aspectos: um estático e um dinâmico.

A formulação de um algoritmo geralmente consiste em um texto contendo comandos (instruções) que devem ser executadas numa ordem prescrita. Esse texto é uma representação concreta do algoritmo e tem um caráter evidentemente estático e atemporal.

Por outro lado esse texto não nos interessa em si, mas pelos efeitos que pode evocar sua execução no tempo, dado um conjunto de valores iniciais.

A grande dificuldade na concepção e no entendimento de algoritmos é o problema do relacionamento desses dois aspectos, ou seja, como entender as estruturas dinâmicas das possíveis execuções do algoritmo a partir da estrutura estática do texto do algoritmo.

Um algoritmo deve nos permitir que dado um estado inicial, sua execução nos leve sempre a um mesmo resultado final, através dos mesmos caminhos.

2.2. Métodos de Desenvolvimento de Algoritmos

O uso desses métodos surgiu com a necessidade de se obter algoritmos de uma forma estruturada, concisa e padronizada, para que possa ser de fácil entendimento por outras pessoas sem gerar ambiguidades. Esses métodos devem possuir certa estruturação para que o trabalho de transformação de algoritmos para programas seja fácil e eficiente.

Dentre os vários existentes podemos citar: FLUXOGRAMAS, Scratch, PSEUDOCÓDIGO, ETC...

2.2.1. Fluxogramas



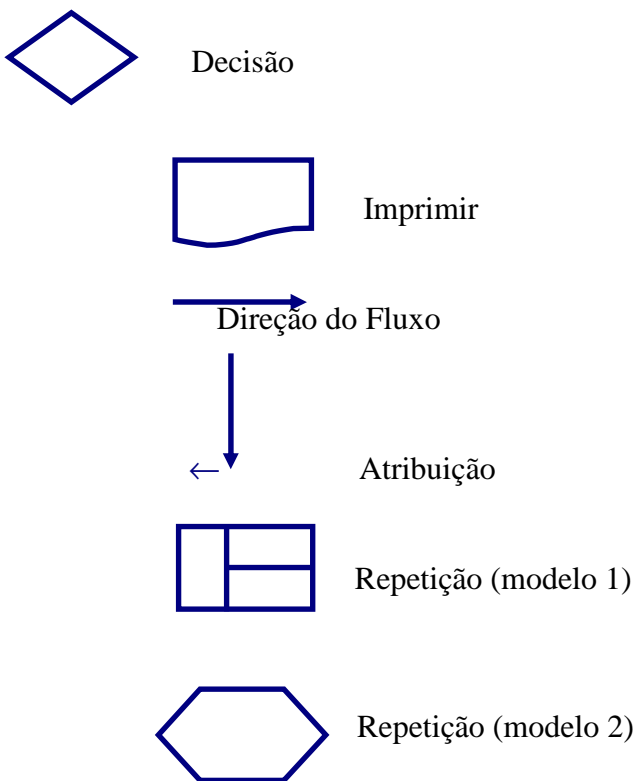
Início/Fim



Entrada de Dados (Cartão)



Atribuição, Operação, Processamento



2.2.2. Pseudocódigos Portugol

Escreve a lógica da solução de forma narrativa, de acordo com certas regras.

Pode ser modificado facilmente e facilmente traduzido em linguagem humana ou em instruções formais para computadores. Dentre os existentes podemos citar o Portugol, que é uma simbiose do português, Algol e Pascal. É uma pseudo-linguagem cujo objetivo principal é fazer com que o projetista passe a pensar no problema e não na máquina, porém não se afastando dela.

3. COMPILADOR DE ALGORITMOS VISUALG

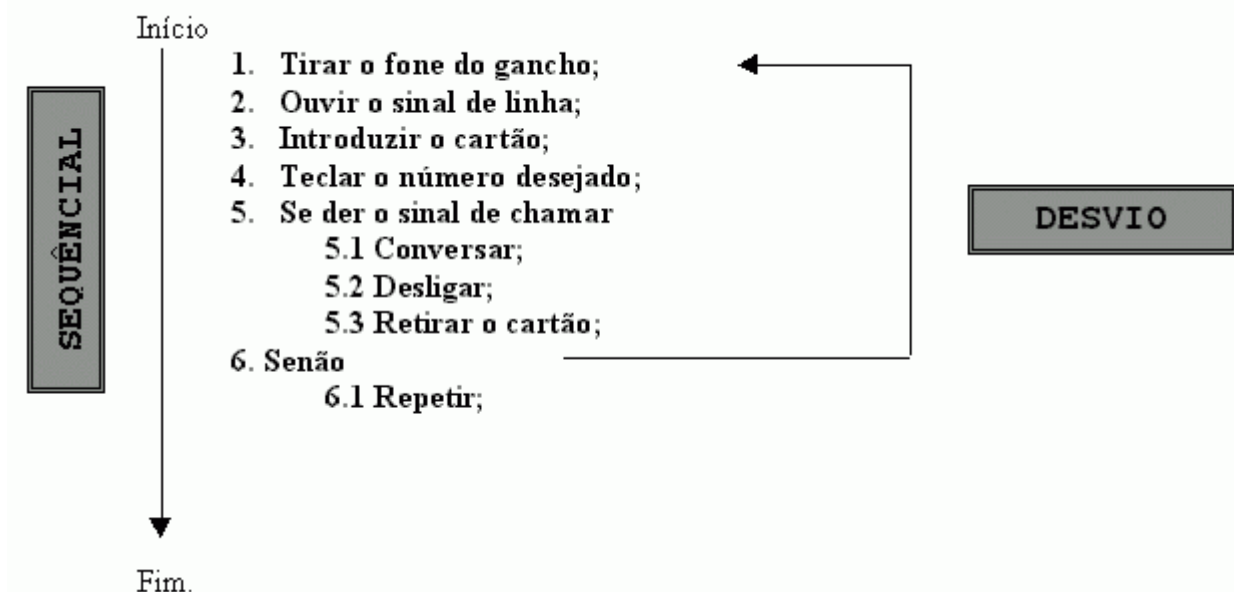
O objetivo principal do sistema VisuAlg (a versão na época da confecção da apostila é a 2.0) é fornecer uma ferramenta de apoio programação baseado no Português Estruturado, sem reduzir o estudo teórico. Com estas ferramentas pretende-se proporcionar uma forma de estimular os alunos a praticar e exercitar o desenvolvimento de algoritmos em uma pseudo-linguagem conhecida como Portugol ou Português Estruturado.

Portugol é derivado da aglutinação de Português + Algol + Pascal. Algol é o nome de uma linguagem de programação estruturada usada no final da década de 50.

Embora o português estruturado seja uma linguagem bastante simplificada, possui todos os elementos básicos e uma estrutura semelhante à de uma linguagem típica para programação de computadores. Além disso, resolver problemas com português estruturado, pode ser uma tarefa tão complexa quanto escrever um programa em uma linguagem de programação qualquer. Portanto, neste manual, estaremos na verdade procurando desenvolver as habilidades básicas que serão necessárias para adquirir-se competência na programação de computadores.

Exemplo de um Algoritmo não Computacional

Abaixo é apresentado um algoritmo não computacional cujo objetivo é usar um telefone público.



Algoritmo1: Trocar uma lâmpada

- Pegar uma escada
- Posicionar a escada embaixo da lâmpada
- Buscar uma lâmpada nova
- Subir na escada

- Retirar a lâmpada velha
- Colocar a lâmpada nova

Determinar uma sequência é importante para reger o fluxo de execução do algoritmo. Ao analisar novamente o algoritmo anterior, perceber-se que tem um objetivo definido de trocar a lâmpada. Porém, e se a lâmpada não estiver queimada? Pode-se efetuar um teste para verificar essa possibilidade. Uma nova solução seria:

Algoritmo2: Trocar uma lâmpada verificando se a mesma está queimada

- Acionar o interruptor
- Se a lâmpada não acender, então
 - Pegar uma escada
 - Posicionar a escada embaixo da lâmpada
 - Buscar uma lâmpada nova
 - Subir na escada
 - Retirar a lâmpada velha
 - Colocar a lâmpada nova

Percebe-se que sempre é possível melhorar um algoritmo prevendo situações que são pertinentes ao problema. Muitas outras instruções poderão ser acrescentadas a esse algoritmo.

Todo mundo que tem contato com computadores sabe que eles precisam ser programados para executar tarefas. Um programa é um conjunto de milhares de instruções que indicam ao computador, passo a passo, o que ele tem que fazer. Estes programas são construídos com ferramentas chamadas "linguagens de programação". Estas linguagens contêm os comandos que fazem o computador escreverem algo na tela realizar cálculos aritméticos, receber uma entrada de dados via teclado, e milhares de outras coisas, mas estes comandos precisam estar em uma ordem lógica e contribuir, cada um, para a tarefa em questão.

3.1. Exercícios

Escreva um algoritmo para resolver os seguintes problemas:

A - Descrever como você faz para trocar o pneu de um carro.

B - Três homens desejam atravessar um rio. O barco que possuem tem a capacidade máxima de 150 quilos. Eles pesam 50, 75 e 120 quilos. Como podem atravessar sem afundar o barco?

C - Um homem precisa atravessar um rio com um barco que possui capacidade para carregar apenas ele mesma, e mais uma de suas três cargas, que são: um lobo, um bode e um fardo de alfafa, mais o lobo não pode ficar sozinho com o bode e o bode com a alfafa.

D – Três homens participam de uma brincadeira, em que dois deles enxergam normalmente, e um terceiro é cego. Existem 5 chapéus, sendo 3 vermelhos e 2 brancos. São colocados ao acaso um desses chapéus na cabeça de cada um deles. A idéia é a seguinte: O primeiro consegue enxergar o chapéu do segundo e do terceiro (cego), mas não consegue enxergar o próprio chapéu. O segundo consegue enxergar o chapéu do primeiro e do terceiro (cego), mas não consegue enxergar o seu próprio. E o terceiro, obviamente, não enxerga. Foi perguntado ao primeiro homem, se ele, ao ver o chapéu do segundo e do terceiro, sabe com certeza a cor do próprio chapéu. Ele responde que não. O segundo ouve a resposta do primeiro, e também responde que também não sabe a cor do próprio chapéu. O cego, por sua vez, ouve a resposta do primeiro e do segundo, e responde que sabe com certeza qual a cor do próprio chapéu. A questão é: Qual a cor do chapéu do cego? Porque ele afirma com certeza que sabe a resposta certa?

Desafio - Dois monges estão perdidos numa mata e estão passando fome. Só existe uma planta que podem comer, mas para comê-la deverá esquentá-la 30 segundos exatos, senão os matará. Para marcar o tempo, eles só têm 2 ampulhetas: uma que marca 22 segundos e outra que marca 14 segundos. Como eles devem fazer para conseguir marcar o tempo de 30 segundos?

3.2. Linearização de Expressões

Para a construção de algoritmos, todas as expressões aritméticas devem ser linearizadas, ou seja, colocadas em linhas. É importante também ressaltar o uso dos operadores correspondentes da aritmética tradicional para a computacional.

Exemplo:

$\left[\frac{2}{3} + (5 - 3) \right] + 1$	$(2/3 + (5 - 3)) + 1$
Tradicional	Computacional

3.2.1. Operadores Aritméticos

Operador	Sintaxe Visualg
Adição	+
Subtração	-
Multiplicação	*

Divisão	/
Divisão Inteira	\
Potenciação	^ ou exp (base, expoente)
Módulo (resto da divisão)	%
Raiz quadrada	Raizq(expressão)
Quadrado	Quad(expressão)

3.2.2. Operadores Aritméticos não convencionais

7 % 3: obtém o resto da divisão de 7 por 3. Normalmente utilizado para agrupar conjunto de valores. Nesse exemplo, o resultado é 1.

Ex.: Par, Impar, Bissexto, etc.

18 \ 7: obtém o quociente da divisão de 18 por 7. Nesse exemplo, o resultado é 2.

3.2.3. Operadores Relacionais

Operadores relacionais atuam sobre operandos numéricos e resultam em valores lógicos (operadores de comparação entre dois operandos).

Operador	VisuAlg
Maior	>
Menor	<
Maior ou igual	>=
Menor ou igual	<=
Igual	=
Diferente	<>

3.2.4. Operadores Lógicos

Atuam sobre expressões retornando sempre os valores lógicos: VERDADEIRO ou FALSO.

VisuAlg	Função
E	Retorna verdadeiras se ambas as partes forem verdadeiras.
OU	Basta que uma das partes seja verdadeira para retornar verdadeiro.
NAO	Nega uma afirmação, invertendo o seu valor lógico. Se for verdadeiro vira falso, se for falso vira verdadeiro.
XOU	Operador que resulta em VERDADEIRO se os operandos forem diferentes, e em FALSO se forem iguais.

3.2.4.1. Tabela Verdade

A	B	A E B	A OU B	NÃO (A)	NÃO (B)	A XOU B
V	V	V	V	F	F	F
V	F	F	V	F	V	V
F	V	F	V	V	F	V
F	F	F	F	V	V	F

3.2.5. Prioridade dos operadores

Assim como nas operações aritméticas, também existe uma relação de prioridade entre os operadores. A lista abaixo mostra as prioridades, da maior prioridade para a menor.

Parênteses e Funções

Operadores Aritméticos na seguinte ordem:

Potenciação: ^;

Multiplicação e Divisão: *, /;

Soma e Subtração: +, -;

Operadores Relacionais: =, <>, >, <, >=, <=;

Operadores Lógicos na seguinte ordem: NAO, E, OU.

3.2.6. Modularização de Expressões

Modularização é a divisão da expressão em partes, proporcionando maior compreensão e definindo prioridades para a resolução da mesma.

$((2 + 2) * 4 + 8) / 2 = 12$ é diferente de $2 + 2 * 4 + 8 / 2 = 14$

Como pôde ser observado no exemplo anterior, é importante definir a prioridade das operações através dos parênteses, pois na informática podemos ter parênteses dentro de parênteses, como seriam os colchetes e as chaves na matemática. Além disso, em expressões computacionais utilizamos somente parênteses para modularização.

3.2.7. Expressões Lógicas

São as expressões compostas de relações que sempre retornam um valor lógico.

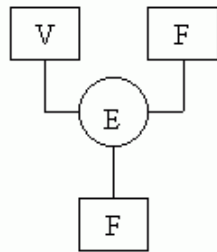
Exemplos:

$2+5>4$? Verdadeiro	$3<>3$? Falso
----------------------	----------------

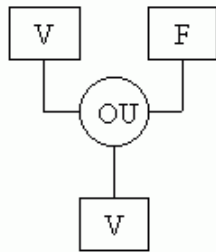
De acordo com a necessidade, as expressões podem ser unidas pelos operadores lógicos.

Exemplos:

$2+5>4 \text{ E } 3<3 \rightarrow \text{Falso}$



$2+5>4 \text{ OU } 3<3 \rightarrow \text{Verdadeiro}$



$\text{NÃO}(3<3) \rightarrow \text{Verdadeiro}$



3.2.8. Exercícios

1 - Escreva as expressões na forma computacional:

a) $ae + \frac{b^{3x}}{c^2} =$

b) $\frac{2x^2 - (3x)^{x+1}}{2} + \frac{\sqrt{x+1}}{x^2} =$

c) $2h - \left(\frac{45}{3x} - 4h(3-h) \right)^{22k} =$

d) $\frac{\sqrt{2b-4a^2} + 2f^{-3}}{3-2a} =$

e) $\frac{\sqrt{-6^x + (2y)^{\frac{1}{3}}}}{3^9} =$

$$f) \sqrt{\frac{2x+u^{\frac{2}{3}}}{a+bc}} =$$

2 - Escreva as expressões da forma tradicional

$$\begin{aligned} &a+b+(34+\exp(e,9))/u-89^{(1/2)}= \\ &23+5/((7*a)/47)^{(2/x)}= \\ &((a+x)^{(2+w)}-\text{quad}(5b))/2= \\ &(12*x)/(36-9^x)^2= \end{aligned}$$

3 - Resolva as expressões lógicas

- a) não $(2^3 < \text{raizq}(16))$ ou $15 \setminus 2 < 10$
- b) $(6 < 8)$ ou $(3 > 7) =$
- c) Não $(2 < 3) =$
- d) $(5 > 6)$ ou $(6 < 7)$ ou não $(a+5-6=8)$ {sabendo-se que $A=5$ }
- e) $(34 > 9 \text{ e } 5+u = 34)$ ou $(5=15/3 \text{ e } 8 > 12)$ {sabendo-se que $u=29$ }
- f) $10 \% 4 < 16 \% 3$
- g) $2+8 \% 7 \geq 6-\text{raizq}(8^{(2/3)})$
- h) $15 \setminus 7 \geq 27 \% 5$
- i) $2 < 5$ ou $15/3=5 < 4$ e $17 \% 5 = 2$

3.3. Primeiros Elementos da Linguagem

No curso de algoritmos vamos escrever soluções para problemas utilizando o português estruturado. Nessa seção vamos conhecer os primeiros elementos que compõem a linguagem e escrever alguns algoritmos.

Estrutura Geral

```
Algoritmo "NomedoAlgoritmo"
var
  <declaração de variáveis>
Início
  <lista de Comandos>
Fimalgoritmo
```

Os algoritmos que vamos escrever têm a seguinte forma geral:

A palavra “Algoritmo” e a expressão “Fimalgoritmo” fazem parte da sintaxe da linguagem e sempre delimitam o início e fim de um algoritmo.

Em < declaração de Variáveis> descrevemos os tipos de dados que serão usados na lista de comandos. Por exemplo, poderíamos definir que fruta é um tipo de dado que pode assumir apenas os valores maçã, pêra, banana, abacaxi e outras frutas, sobre os quais podemos efetuar as operações de comparar, comprar, comer e servir.

“Início” indica o fim das declarações e o início dos comandos.

< listas-de-comando> é apenas uma indicação de que entre a palavra “Início” e a expressão “Fimalgoritmo” podemos escrever uma lista com uma ou mais instruções ou comandos.

É importante salientar que, quando um algoritmo é “executado” as instruções ou comandos de um algoritmo são sempre executados na ordem em que aparecem no mesmo.

As palavras que fazem parte da sintaxe da linguagem são palavras reservadas, ou seja, não podem ser usadas para outro propósito em um algoritmo que não seja aquele previsto nas regras de sintaxe. A palavra “Algoritmo”, por exemplo, é uma palavra reservada. As palavras reservadas sempre aparecerão em **negrito** e *itálico*, neste manual.

3.3.1. Identificadores

Tudo que usamos tem um nome. Este nome ou rótulo é utilizado para que possamos reconhecer uma determinada variável ou constante. O objetivo desse rótulo ou nome é estabelecer a necessidade de identificarmos o espaço que iremos armazenar determinado conteúdo. Esta identificação deve ser clara, precisa e deve identificar o conteúdo guardado para o usuário.

3.3.1.1. Regras para a nomeação de identificadores

Não podem ter nomes de palavras reservadas;

Devem possuir como primeiro caractere uma letra ou sublinhado '_';

Outros caracteres podem ser letras, números e sublinhado;

Ter no máximo 127 caracteres;

Não possuir espaços em branco;

O VisuAlg não difere letras maiúsculas de minúsculas (NOME é o mesmo que noMe).

Identificadores válidos: NOME, TELEFONE, IDADE_FILHO, NOTA1, Est_Civil

Identificadores inválidos: 3Endereco, Estado Civil, PARA, fim, numero/complemento

3.3.2. Variáveis

O computador possui uma área de armazenamento conhecida como memória. Todas as informações existentes no computador estão ou na memória primária (memória RAM), ou na memória secundária (discos, fitas, CD-ROM, etc.). Nós iremos trabalhar, neste curso, somente com a memória primária, especificamente com as informações armazenadas na RAM (memória de acesso aleatório).

A memória do computador pode ser entendida como uma sequência finita de caixas, que, num dado momento, guarda algum tipo de informação, como número, uma letra, uma palavra, uma frase, etc, não importa, basta saber que lá sempre existe alguma informação.

O computador, para poder trabalhar com alguma destas informações precisa saber onde, na memória, o dado está localizado. Fisicamente, cada caixa, ou cada posição de memória, possui um endereço, ou seja, um número que indica onde cada informação está localizada. Este número é representado através da notação hexadecimal, tendo o tamanho de quatro, ou mais bytes. Abaixo segue alguns exemplos:

Endereço Físico Informação

3000: B712 é o endereço físico de "João"

2000: 12EC é o endereço físico de 12345

3000: 0004 é o endereço físico de "H"

Como pode ser observado, o endereçamento das posições de memória através de números hexadecimais é perfeitamente compreendido pela máquina, mas para nós humanos torna-se uma tarefa complicada. Pensando nisto, as linguagens de computador facilitaram o manuseio, por parte dos usuários, das posições de memória da máquina, permitindo que, ao invés de trabalhar diretamente com o número hexadecimal, fosse possível dar nomes diferentes a cada posição da memória. Tais nomes seriam de livre escolha do usuário. Com este recurso, os usuários ficaram livres dos endereços físicos (números hexadecimais) e passaram a trabalhar com endereços lógicos (nomes dados pelos próprios usuários). Desta forma, o exemplo acima poderia ser alterado para ter o seguinte aspecto:

Endereço Físico Informação

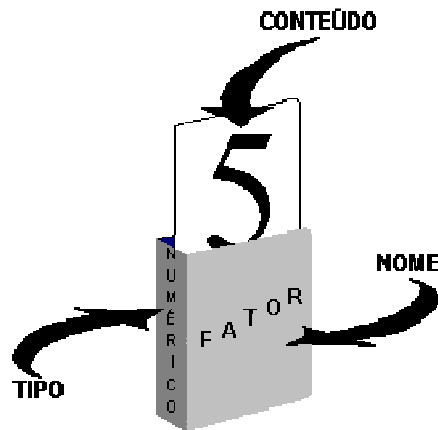
Nome : "João"

número : 12345

letra : "H"

Como tínhamos falado, os endereços lógicos são como caixas, que num dado instante guardam algum tipo de informação. Mas é importante saber que o conteúdo desta caixa não é algo fixo, permanente, na verdade, uma caixa pode conter diversas informações, ou seja, como no Exemplo acima, a caixa (Endereço Lógico) rotulada de "Nome" num dado momento contém a informação "João", mas em um outro momento, poderá conter uma outra informação, por Exemplo, "Pedro". Com isto queremos dizer que o conteúdo de uma destas caixas (endereço lógico) pode variar, isto é, podem sofrer alterações em seu conteúdo. Tendo este conceito em mente, a partir de agora iremos chamar de forma genérica, as caixas ou endereços lógicos, de variáveis.

Desta forma podemos dizer que uma variável é uma posição de memória, representada por um nome simbólico (atribuído pelo usuário), a qual contém, num dado instante, uma informação, conforme pode ser observado na figura abaixo:



Uma variável pode ser vista como uma caixa identificada por um nome colocado na tampa desta e por um valor de uma constante que poderá estar colocado no interior desta caixa. Enquanto o nome sempre permanece o mesmo, o conteúdo da caixa, isto é, a constante nela armazenada pode ser alterada a qualquer momento.

3.3.2.1. Declaração de Variáveis

Variáveis são palavras que tem um significado bem específico em um algoritmo. Para que o computador possa executar comandos que envolvem variáveis da maneira correta, ele deve conhecer os detalhes das variáveis que pretendemos usar. Esses detalhes são: o identificador desta variável e o tipo de valores que essa variável irá conter. Precisamos assim, de uma maneira de especificar esses detalhes e comunicá-los ao computador. Para isso utilizamos o comando de declaração de variáveis que faz parte da nossa linguagem que tem a seguinte forma:

<identificador> [, <lista_identificadores>] : <tipo_das_variáveis>

onde:

<identificador> - é uma lista de palavras que pretendemos empregar como variáveis em nosso algoritmo;

<tipo_das_variáveis> - determina que tipo de valor as variáveis poderão receber.

Todas as Variáveis devem assumir um determinado tipo de informação

O tipo de dado pode ser:

Primitivo: Pré-definido pela linguagem;

Estáticos: É uma parte de um tipo já existente (vetores, como será visto adiante);

Dinâmicos - Definidos pelo programador(alocação dinâmica).

3.3.2.2. Tipos Primitivos de Dados

Esses tipos de dados são os mais frequentes nas linguagens de programação, pois oferecem alguns tipos básicos pré-definidos, para criação de outros tipos. Cada um desses tipos primitivos tem um conjunto de valores restrito, veja abaixo:

	Descrição
Inteiro	Representa valores inteiros. Exemplo: 10, 33, -56
Real	Representa valores inteiros ou reais (com ponto separador da parte decimal). Exemplos: 10, 15.5
Caracter	Um número, uma letra, qualquer símbolo especial ou sequência de caracteres entre aspas duplas. Exemplos: "a", "=", "#\$%@", "999", "678.90"
Logico	Só podem assumir dois valores: verdadeiro ou falso.

3.3.3. Sinal de Atribuição

Quando declaramos uma variável, criamos uma área de armazenamento para os dados, mas ela ainda está sem valor. Para que a variável seja útil, deve ter valores colocados por nós. Isso se chama "atribuir valores". A atribuição de valores é feita pelo operador de atribuição "<-" (uma seta apontando para a esquerda).

Por exemplo:

```
Peso <- 78 // Este comando atribui à variável Peso o valor 78.
Nome <- "João da Silva" // Este comando atribui à variável Nome o valor "João da Silva".
```

É importante lembrar que só se pode atribuir às variáveis valores do mesmo tipo da variável. Nos exemplos acima, o variável Salário é do tipo real. Então, o seguinte comando seria inválido:

```
Salário <- "Insuficiente"
```

Uma variável, como o próprio nome está dizendo, pode ter seu conteúdo (seu valor) mudado quantas vezes forem necessárias durante um programa.

3.3.4. Comandos de E/S (Entrada/Saída)

Um programa que faça o seu processamento e não tenha como mostrar seus resultados é praticamente inútil. Portanto, em algum ponto deve haver a exibição de valores, e todas as

linguagens de programação têm comandos para este fim. Nos algoritmos usamos o comando **Escreva** para isto.

Escreva – Comando de saída que exhibe uma informação no monitor.

Escreval – Comando de saída que exhibe uma informação no monitor, mas executa o comando seguinte na linha seguinte.

Sintaxe:

Escreva (<expressão | variável>[,<lista_de_variáveis>])

Exemplo: **Escreva** ("Este é o valor de X:", **X**, "e este de Y:", **Y**)

Nem todos os dados que um programa manipula são gerados por ele. Um programa de caixa automático, por exemplo, têm que obter do usuário o número da conta, a senha, a opção de serviço desejada, etc. Assim, deve haver um meio para que sejam digitados (ou fornecidos de outra maneira) dados para uso do programa. Mais uma vez, todas as linguagens de programação permitem isto, e nos algoritmos usamos o comando **Leia**. A sintaxe deste comando é:

Leia – Comando de entrada que permite a leitura de variáveis de entrada utilizando o teclado.

Sintaxe:

Leia (<variável>[,<lista_de_variáveis>])

Assim, temos:

Comando de Saída	Comando de Entrada
Escreva (variável) ou Escreval (variável)	Leia (variável)

3.3.5. Linhas de Comentário

Os comentários são declarações não compiladas que podem conter qualquer informação textual que você queira adicionar ao código-fonte para referência e documentação de seu programa.

Uma Linha

São representados por duas barras normais (//). Todo o texto que você digitar após as duas barras será comentário.

Exemplo:

```
// Este método calcula o fatorial de n.
```

```
.
```

```
x <- y    // Inicializa a variável x com o valor de y.
```

Múltiplas Linhas

Para tal, basta colocar o comentário entre chaves { }

Exemplo:

```
ALGORITMO "Exemplo"
{algoritmo que pega o nome do aluno, seu código e as suas notas, e verifica se ele passou de ano }
var
    codigo, i, j, nota1, nota2, nota3: inteiro { declaração das variáveis do tipo inteiro }
    nome: caracter // declaração das variáveis de cadeia de caracteres
FIMALGORITMO
```

Uma Linha	Múltiplas Linhas
// TEXTO	{ TEXTO }

3.3.6. Estruturas Sequenciais - Construindo os Primeiros Algoritmos

Basicamente a construção de um algoritmo se resume às seguintes etapas:

Entendimento do problema;

Definição dos dados que serão necessários para resolvê-lo (as entradas); aí já deveremos ter idéia dos tipos de dados que usaremos;

Obtenção destes dados; alguns vêm do "exterior" do programa, e outros são calculados no próprio programa;

Processamento em si;

Exibição dos resultados.

A primeira fase é a mais difícil de se "pegar", pois depende um pouco da experiência do programador. Entretanto, mesmo nesta fase há técnicas que podem ser aprendidas, e modelos que podem ser aplicados. Programação é arte, ciência e técnica, tudo ao mesmo tempo...

Problema 1 - Cálculo de Média Aritmética

Enunciado:

Faça um programa que leia dois valores inteiros, e calcule e exiba a sua média aritmética.

Etapa 1

Simples, hein? Dos tempos de escola lembramos que a média aritmética de dois valores é calculada como $(a+b) / 2$, e sendo assim a primeira etapa já está pronta.

Etapa 2

Os dados necessários serão os dois valores, que colocaremos em duas variáveis A e B, do tipo real, e uma terceira variável, que chamaremos Media, que armazenará a média aritmética calculada.

Etapa 3

A obtenção dos dados neste programa é simples e direta. Basta pedir ao usuário que digite os valores.

Etapa 4

O processamento aqui é o cálculo da média, usando o método citado acima, na etapa 1. O resultado do cálculo será armazenado na variável Media.

Etapa 5

Basta exibir o conteúdo da variável Media.

Solução:

Algoritmo “CalculaMedia”

```
var
    A,B: Inteiro
    Media : Real
Inicio
Escreval("Programa que calcula a média aritmética de dois
valores.")
Escreval("Digite um valor : ")
Leia(A)
Escreva("Digite outro valor : ")
Leia(B)
Media <- (A+B)/2
Escreva("A média dos dois valores é : ", Media)
FimAlgoritmo
```

Comentários

Você deve ter notado que colocamos na tela instruções para o usuário usando o comando Escreva. Esta é uma boa técnica de programação, mesmo hoje em dia, com o ambiente gráfico do Windows. Da mesma forma, ao imprimir o resultado, não mostramos simplesmente a média, mas explicamos ao usuário o que aquele valor significa.

Como pode ser analisado no tópico anterior todo programa possui uma estrutura sequencial determinada por um Inicio e Fim. Em um algoritmo, estes limites são definidos com as palavras.

Algoritmo "teste"

```
var
  prod: character
  valor_prod, qtd, tot: real
Inicio
  escreva("digite o nome do produto:")
  leia(prod)
  escreva("digite o valor unitário do produto:")
  leia(valor_prod)
  escreva("digite a quantidade:")
  leia(qtd)
  tot <- valor_prod*qtd
  escreva("total:",qtd, " unidades de ",prod," = ", tot)
finalgoritmo
```

Lembrete:

Os comandos devem ser inseridos linha por linha.

Não são diferenciadas as letras maiúsculas de minúsculas.

As palavras não devem ser acentuadas.

E não deve ser utilizada a letra 'ç'.

Se executássemos esse algoritmo, teríamos o seguinte resultado:

1º Passo: "Digite o produto:" Uma janela irá aparecer, pedindo para inserir o dado desejado. Vamos supor que você digite **Abacaxi**

2º Passo: "Digite o valor unitário do produto:" Você digita **5.50**

3º Passo: "Digite a quantidade:" Você digita o valor **10**

4º Passo: O computador irá executar a operação da antepenúltima linha do algoritmo: atribuir o resultado da multiplicação entre valor_prod e qtd à variável tot.

5º Passo: Total: 10 unidades de Abacaxi = 55.0

3.3.7. Exercícios

A. Classifique os conteúdos das variáveis abaixo de acordo com seu tipo, assinalando com I os dados Inteiros, com R os dados Reais, com L os lógicos, e com C os caracteres.

() 0.1	() "abc"	() "João"
() 5.7	() 1012	() "F"
() -49	() +342	() 569
() "Lucas"	() "V"	() 0.00001
() falso	() -545	() "444"

B. Assinale com um X os nomes de variáveis válidos.

- | | | |
|---------------------------------|---------------------------------|---------------------------------------|
| <input type="checkbox"/> abc | <input type="checkbox"/> 3abc | <input type="checkbox"/> a |
| <input type="checkbox"/> 123a | <input type="checkbox"/> -a | <input type="checkbox"/> acd1 |
| <input type="checkbox"/> -_ad | <input type="checkbox"/> A&a | <input type="checkbox"/> 1 |
| <input type="checkbox"/> A123 | <input type="checkbox"/> Aa | <input type="checkbox"/> guarda-chuva |
| <input type="checkbox"/> AB CDE | <input type="checkbox"/> etc... | <input type="checkbox"/> b316 |

C. Para as variáveis declaradas a seguir são dados os valores informados. Determine o resultado da avaliação das expressões abaixo. X, Y e Z são variáveis do tipo inteiro, sendo que: $X = 2$, $Y = 3$ e $Z = 5$.

- | | |
|------------------|-----------------------|
| a) $X * Y - Z$ | e) $(X + Y) * Z$ |
| b) $X * (Y - Z)$ | f) $\exp(X, Y) - 1$ |
| c) $X + Y * Z$ | g) $(X \wedge Y) - 1$ |
| d) $X + (Y * Z)$ | h) $X \wedge (Y - 1)$ |

D. Assinalar os comandos de atribuição considerados inválidos.

Nome, cor, Teste, Dia: Character

Soma, Num: Real

X: Logico

- a) ☐ NOME <- 5
- b) ☐ SOMA <- NUM + 2 * X
- c) ☐ TESTE <- SOMA
- d) ☐ NUM <- SOMA
- e) ☐ COR <- "PRETO"
- f) ☐ X <- X + 1
- g) ☐ NUM <- "*ABC*"
- h) ☐ DIA <- "SEGUNDA"
- i) ☐ SOMA + 2 <- X
- j) ☐ X <- (NOME = COR)

E. Quais os valores armazenados em SOMA, NOME e TUDO, supondo-se que NUM, X, COR, DIA, TESTE e COD valem, respectivamente, 5; 2; "AZUL"; "TERÇA"; Falso e Verdadeiro?

- a) NOME <- DIA
- b) SOMA <- (NUM^2/X) + (X + 1)
- c) TUDO <- NÃO ((TESTE OU COD) E (SOMA < X))

F. Observar o seguinte algoritmo e descreva o que ele faz.

```

Algoritmo "CalculaMedia"
var
    nota1, nota2, nota3, nota4, media: real
    nome: caracter
inicio
    leia(nome)
    leia(nota1, nota2, nota3, nota4)
    media <- (nota1 + nota2 + nota3 + nota4)/4
    escreva (nome, media)
FimAlgoritmo
  
```

G. Resolva os exercícios na forma de pseudocódigos Portugol

- 1) Faça um programa que leia três valores inteiros, e calcule e exiba a sua média ponderada. A primeira nota tem peso 2, a segunda tem peso 3 e a terceira tem peso 5.
- 2) Realizarei uma viagem de vários dias em meu automóvel, e gostaria de saber a quilometragem média por litro de gasolina. Para isto, anotarei a quilometragem no velocímetro ao sair de viagem, e depois ao chegar. Também vou somar toda a gasolina que comprar para o carro. Você poderia fazer um programa que me desse, com estes dados, quantos km fiz, em média, por litro de gasolina?
- 3) Faça um programa que leia o nome de um piloto, uma distância percorrida em km e o tempo que o piloto levou para percorrê-la (em horas). O programa deve calcular a velocidade média em km/h, e exibir a seguinte frase: A velocidade média de XX foi YY km/h, onde XX é o nome do piloto, e YY é a velocidade média.
- 4) Em uma pizzaria, cada tulipa de chope custa R\$3,80 e uma pizza mista grande custa R\$55,00 mais R\$ 4,50 por tipo de cobertura pedida (queijo, presunto, banana, etc.). Uma turma vai à pizzaria e pede uma determinada quantidade de "chopes" e uma pizza grande com uma determinada quantidade de coberturas. Faça um programa que calcula a conta e, sabendo que a será informada a quantidade de pessoas, quanto que cada um deve pagar. Lembre-se dos 10% do garçom.
- 5) A conversão de graus Fahrenheit para Celsius é obtida pela fórmula abaixo. Faça um algoritmo que leia um valor em graus Centígrados e imprima seu correspondente em graus Fahrenheit.

$$C = \frac{5}{9} * (F - 32)$$

- 6) Fazer um programa que leia um número inteiro e mostre o seu triplo, sua metade, a sua raiz cúbica, e por fim, o número elevado a potência fracionária 2/3.
- 7) Entrar com os valores dos catetos de um triângulo retângulo e imprimir a hipotenusa.
- 8) Construa um algoritmo que, tendo como dados de entrada dois pontos quaisquer no plano, P(x1,y1) e P(x2,y2), escreva a distância entre eles. A fórmula que efetua tal cálculo é:
(Exercício resolvido no Anexo I)

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

9) Determine o que será impresso após a execução do algoritmo abaixo.

```
Algoritmo "misterio"
var
  num, a, b, c, d : inteiro
inicio
  escreva("Informe um número inteiro de 4 dígitos diferentes:")
  leia(num)
  d <- num\1000
  a <- (num\100)%10
  b <- (num%100)\10
  c <- (num%10)%10
  escreva("Resultado:",d,a,b,c)
Fimalgoritmo
```

3.4. Estrutura Condicional

Na vida real tomamos decisões a todo o momento baseado em uma situação existente. Em programação chamamos esta situação de condição, e as alternativas possíveis de ações.

Por exemplo: "Se tiver R\$ 10,00 sobrando, irei ao cinema hoje à noite, mas se não tiver ficarei vendo TV em casa".

Qual é a condição nesta frase? Fácil, "tiver R\$ 10,00 sobrando". Ela é uma expressão lógica, pois a pergunta "Tenho R\$ 10,00 sobrando?" pode (tem que) ser respondida com "Sim" ou "Não". Lembre-se, então: em um algoritmo, *toda condição tem que ser uma expressão lógica*. Quais são as ações possíveis? Fácil, mais uma vez; "irei ao cinema" e "ficarei vendo TV em casa". A primeira só será realizada se a resposta à pergunta "Tenho dinheiro suficiente?" for "Sim", enquanto que a segunda será realizada caso a resposta seja "Não".

Então, em um algoritmo, as ações são um ou mais comandos que serão realizados, em alguns casos a avaliação da condição resulte em Verdadeiro, outros caso ela resulta em Falso.

Vamos colocar agora a frase do parágrafo anterior em outra forma, mais parecida com o que é um programa de computador:

Se "tiver R\$ 10,00 sobrando" **então**

 "Irei ao cinema"

Senão

 "ficarei vendo TV em casa".

Veja que grifamos três palavras: **Se**, **então**, **senão**. Elas são muito importantes na estrutura dos comandos de decisão. Como próximo passo, vamos generalizar a estrutura que criamos acima:

Se <condição> **então**

<ações (uma ou mais) a serem realizadas se a condição for verdadeira>

senão

<ações (uma ou mais) a serem realizadas se a condição for falsa>

Para terminar a nossa comparação, devemos lembrar que os comandos do algoritmo são sempre imperativos, e que o computador só lida com quantidades definidas (ou seja, ele não sabe o que é "ter R\$ 10,00 sobrando"). Para aproximar mais nossa frase de um algoritmo, poderemos ter a seguinte forma:

Se Dinheiro >= 10 então

Ir ao Cinema

senão

Ver TV em Casa

Entendeu a transformação? "Dinheiro" faz o papel de uma variável que contém o que eu tenho sobrando no momento, e se valor é maior ou igual a 10, então "tenho R\$ 10,00 sobrando". Por falar nisso, fique sabendo que a técnica (ou arte) de se transformar perguntas do dia-a-dia em quantidades definidas que possam ser colocadas em um programa é a chave de se fazer algoritmos. Não se preocupe, no entanto: é algo fácil e que pode ser aprendido e desenvolvido. Bom, agora já podemos fazer um programa que ajude nosso amigo...

O que faço esta noite?

Faça um programa que peça ao usuário a quantia em dinheiro que tem sobrando e sugira, caso ele tenha 10 ou mais reais, que vá ao cinema, e se não tiver, fique em casa vendo TV.

Algoritmo "Tem dinheiro sobrando?"

```
var
    Dinheiro: Real
Inicio
    Escreval("Serviço Informatizado de Sugestões")
    Escreva("Quanto dinheiro você tem sobrando?")
    Leia (Dinheiro)
    Se Dinheiro >= 10 entao
        Escreva("Vá ao cinema hoje à noite.")
    senao
        Escreva("Fique em casa vendo TV.")
    Fimse
    Escreval("Obrigado e volte sempre.")
```

Em relação ao que vimos até agora, apenas uma novidade: a expressão *fimse* ao final do comando de decisão. Ela delimita o comando, isto é, mostra onde as ações da parte *senão* do comando terminam. O algoritmo abaixo está incorreto:

Se Dinheiro >= 10 entao

Escreva ("Vá ao cinema hoje à noite.")

Senao

Escreva ("Fique em casa vendo TV.")

Escreva ("Obrigado e volte sempre.").

Neste caso, o computador não saberia se o comando Escreva "Obrigado e volte sempre." faria ou

não parte do comando de decisão (a endentação, ou seja, o fato de algumas linhas estarem mais distantes da margem esquerda que as outras não quer dizer nada para o computador. Fazemos isto apenas - *e esta é uma dica importante para você* - para que o algoritmo fique mais fácil de ler). Assim o fimse é fundamental, e todas as linguagens de programação têm algo que cumpra esta função.

A estrutura condicional permite a mudança de caminho em um determinado fluxo devido à veracidade de um determinado teste feito no decorrer do desenvolvimento do algoritmo.

A estrutura "Se" deve ser aplicada nos momentos em que, de acordo com uma determinada condição, um processamento deva ser feito.

Os comandos internos a essa estrutura só serão executados após o teste da condição ter sido feito e seu resultado for verdadeiro. Após serem executados os comandos internos a essa estrutura, o primeiro comando após o "fimse" será executado e o algoritmo seguirá a sequência lógica normalmente.

3.4.1. Tipos de comandos condicionais

- **Seleção Simples:** testa uma condição, se esta for verdadeira executa uma ação ou um conjunto de ações.

Sintaxe	Exemplo
<pre> Algoritmo "Verifica condicional" var //Declaração de variáveis Inicio //Comandos Se <condição> entao C1 ... Cn Fimse FimAlgoritmo </pre>	<pre> Algoritmo "Teste" var A,B: inteiro Inicio A <- 1 B <- 1 Se A = B entao A <- A+1 Fimse FimAlgoritmo </pre>

- **Seleção Composta:** testa uma condição, se esta for verdadeira executa uma ação ou um conjunto de ações. Caso contrário executa outro conjunto de ações:

Sintaxe	Exemplo
<pre> Algoritmo "Verifica condicional" var //Declaração de variáveis Inicio //Comandos Se <condição> entao C1 ... Cn senao D1 ... Dn Fimse FimAlgoritmo </pre>	<pre> Algoritmo "Teste" Var A,B: inteiro Inicio A <- 1 B <- 2 Se A > B entao A <- 5 senao A <- B+2 Fimse FimAlgoritmo </pre>

- **Seleção Encadeada:** agrupa várias alternativas a fim de inspecionar mais de duas condições.

Sintaxe	Exemplo
<pre> Algoritmo "Verifica condicional" var //Declaração de variáveis Inicio //Comandos Se <condição1> entao C1 ... Cn Senao Se <condição2> entao D1 ... Dn Senao Se <condição3> entao X1 ... Xn Fimse Fimse Fimse FimAlgoritmo </pre>	<pre> Algoritmo "Teste" var A,B: inteiro Inicio A <- 1 B <- 3 Se A > B então A <- A+1 B <- B*2 Senao Se B % 2 = 0 entao A <- 0 B <- B+2 Senao Se B % 3 = 0 entao A <- A*2 B <- 5 Fimse Fimse Fimse FimAlgoritmo </pre>

- **Seleção de Múltipla Escolha:** usada quando um conjunto de valores precisa ser testado e ações diferentes são associadas a esses valores.

Sintaxe	Exemplo
<pre> Algoritmo "Verifica Escolha" var //Declaração de variáveis Inicio //Comandos escolha<expressão-de-seleção > caso <exp1>,<exp2>,...,<expn> <lista-de-comandos-1> caso <exp1>,<exp2>,...,<expn> <lista-de-comandos-2> outrocaso <lista-de-comandos-3> fimescolha Fimalgoritmo </pre>	<pre> Algoritmo "Verifica trimestre" var mes: inteiro inicio escreva("Informe o número do mês: (1 a 12): ") leia(mes) escolha mes caso 1,2,3 escreva("Primeiro trimestre") caso 4,5,6 escreva("Segundo trimestre") caso 7,8,9 escreva("Terceiro trimestre") caso 10,11,12 escreva("Quarto trimestre") outrocaso escreva("Mes informado errado") fimescolha Fimalgoritmo </pre>

3.5. Testando o Algoritmo

Um algoritmo depois de ser elaborado pode e deve ser testado. Utilizamos um método conhecido como teste de mesa. O teste de mesa é como uma simulação de todos os passos, ou seja, entradas, comandos e instruções do algoritmo, a fim de saber se ele chega ao resultado a que se propõe e se a lógica está correta. Preenchendo uma tabela com valores para as variáveis e seguindo-se o fluxo de execução do algoritmo. A cada comando o valor das variáveis deve ser atualizado, concluído o teste de mesa podemos analisar os resultados, obtivemos a resposta correta? Se não, onde estão os erros? Sempre que é realizado um teste de mesa, utiliza-se de tabelas para armazenar os valores e estados das variáveis do algoritmo. Monitorando as variáveis é fácil determinar a eficiência do algoritmo. Uma vez entendido o funcionamento de um teste de mesa, e sabendo aplicá-lo nos algoritmos elaborados o aprendizado da lógica computacional se torna muito mais fácil, visto que o teste de mesa faz uma simulação do raciocínio estruturado, utilizando a entrada de variáveis e seguindo passo a passo a lógica. Caso em algum ponto a lógica esteja falha, o erro é facilmente encontrado e facilmente resolvido.

Exemplo:

Para cada variável você deve fazer uma coluna e uma coluna para saída de dados.

Algoritmo	Teste de Mesa			
Algoritmo "Teste"				
var				
a, b, c: Real	A	B	C	Saída
Início	?	?	?	
a ← 5	5	?	?	
b ← 15	5	15	?	
c ← a+b	5	15	20	
Escreva (c)	5	15	20	20
a ← 10	10	15	20	
b ← 25	10	25	20	
c ← a+b	10	25	35	
Escreva (c)	10	25	35	35
a ← a-b	-15	25	35	
Escreva (a)	-15	25	35	-15
a ← 0	0	25	35	
b ← 0	0	0	35	
c ← 0	0	0	0	
FimAlgoritmo				

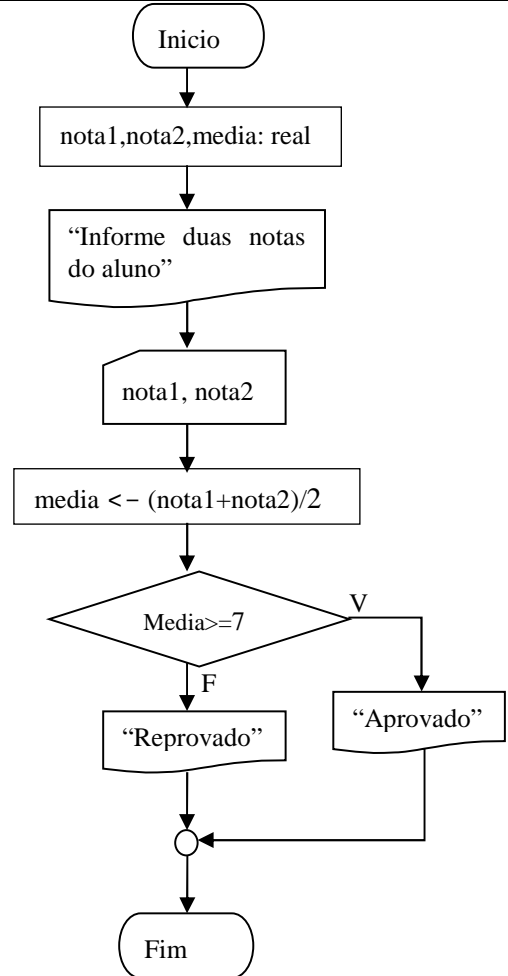
Quando estamos escrevendo um algoritmo devemos ter em mente que eles devem ser facilmente entendidos, testados e modificados por outras pessoas (e por nos mesmos após algum tempo).

3.6. Regras Práticas para Construir Algoritmos Legíveis

1. Use nomes significativos para suas variáveis;
2. Use endentação, isto é, os comandos devem ser alinhados de acordo com o nível a que pertença;
3. Use parênteses para aumentar a legibilidade e prevenir-se contra erros
4. Use apenas um comando por linha
5. Utilize espaços e linhas em branco para melhorar a legibilidade
6. Comentários:
 - . Use comentários para descrever o significado de algumas variáveis;
 - . Coloque-os no momento em que estiver escrevendo o algoritmo;
 - . Coloque comentários no início do algoritmo para explicar o que faz, como utilizá-lo, autor, etc...
 - . Coloque comentários nos principais comandos (os mais importantes), para explicar o objetivo dos mesmos;
 - . Altere os comentários quando alterar o algoritmo;

3.7. Exercício resolvido

Em uma escola, o aluno faz duas provas por período, com as notas variando de 0 a 10. Caso a média aritmética das duas notas seja 7 ou mais, ele passa de ano; senão, ele é reprovado. Faça um programa que receba as duas notas de um aluno e escreva se ele passou ou não de ano.

Visualg 3.0	Fluxograma
<pre> Algoritmo "calcula média" var nota1, nota2, media: real inicio escreva("Informe as duas notas do aluno:") leia(nota1, nota2) media <- (nota1+nota2)/2 se media >= 7 entao escreva("Aluno aprovado") senao escreva("Aluno reprovado") fimse Fimalgoritmo </pre>	 <pre> graph TD Inicio([Inicio]) --> Declara[nota1, nota2, media: real] Declara --> Entrada[/Informe duas notas do aluno/] Entrada --> Dados[nota1, nota2] Dados --> Processa[media <- (nota1+nota2)/2] Processa --> Decida{Media >= 7} Decida -- V --> SaidaA[/Aprovado/] Decida -- F --> SaidaR[/Reprovado/] SaidaA --> Conexao(()) SaidaR --> Conexao Conexao --> Fim([Fim]) </pre>

Dizer os valores a serem impressos após a execução dos algoritmos abaixo:

<pre> algoritmo "exerc 1" var A,B:inteiro inicio A <- 10 B <- 15 se A<= B entao A <- B - 1 senao B <- B +1; A <- A -1; fimse A <- A + 2 escreva(A,B) fimalgoritmo </pre>	<pre> Algoritmo "exerc 2" var A,B,C,D,X,Y: inteiro inicio A <- 62 B <- 30 C <- 44 D <- 75 X <- 4 Y <- 3 se (A> B) e (X > Y) entao X <-Y + A D <-9 + B*2 Y <- QUAD(Y) senao B <- A + D fimse A <- A+B escreva (A, B, C) escreva (D, X, Y) fimalgoritmo </pre>
---	---

Dizer os valores a serem impressos após a execução dos algoritmos abaixo, e convertê-los para fluxogramas.

<pre> Algoritmo "Exerc 3" var X,Y,Z: Inteiro W: Real Inicio X <- 5 Y <- 12 Z <- 4 W <- 0 Se X > Z entao Se Z > Y entao Escreva(Y) W <- (X + Y + Z)/3 Senao escreva(Z) W <- Z + 12 fimse senao escreva(X) fimse escreva(W) fimalgoritmo </pre>	<pre> Algoritmo "Exerc 4" var N1,N2: inteiro SOMA: real inicio N1 <- 8 N2 <- 11 SOMA <- 4 Se N1 > 0 entao SOMA <-SOMA + (N2 - N1) Se N1 % 2 = 0 entao escreval(N1) escreval(soma) fimse escreva(N2) Senao SOMA <- SOMA + N1 + N2 escreva(SOMA) fimse fimalgoritmo </pre>
---	--

3.8. Exercícios – Resolver os algoritmos na forma de pseudocódigo

1. Faça um programa que receba o valor do salário de uma pessoa e o valor de um financiamento pretendido. Caso o financiamento seja menor ou igual a 5 vezes o salário da pessoa, o programa deverá escrever "Financiamento Concedido"; senão, escreverá "Financiamento Negado". Independente de conceder ou não o financiamento, o programa escreverá depois a frase "Obrigado por nos consultar."
2. Dois carros percorreram diferentes distâncias em diferentes tempos. Sabendo que a velocidade média é a razão entre a distância percorrida e o tempo levado para percorrê-la, faça um programa para ler as distâncias que cada carro percorreu e o tempo que cada um levou, e indique o carro que teve maior velocidade média.
3. Faça um programa que leia o nome e idade de duas pessoas e imprima o nome da pessoa mais nova, e seu ano de nascimento (o programa deve funcionar corretamente para qualquer que seja o ano atual).
4. Leia três números e imprima o maior deles. Não é necessário verificar se os números são iguais.
5. Leia um nome e duas notas dadas a um aluno e imprima: nome, média e “aprovado”, “reprovado” ou “recuperação”, de acordo com as regras abaixo:
 Reprovado, se média maior igual a zero e menor que 5;
 Recuperação, se média maior igual a 5 e menor que 7;
 Aprovado, se média maior igual a 7 e menor igual a 10.
6. Leia uma média e um número de faltas dadas a um aluno e imprima aprovado, reprovado por média e por falta, reprovado por média ou reprovado por falta. Sendo média maior igual a 7 indica aprovado, e faltas maior igual a 32 indica reprovado.(Exercício resolvido no Anexo I)
7. Leia três valores que representam os lados de um triângulo, e verificar se compõe um triângulo equilátero (três lados iguais), isósceles (dois lados iguais) ou escaleno (três lados diferentes).
 (Dica: Verificar primeiro se os três lados formam um triângulo: cada lado é menor do que a soma dos outros dois).
8. Construa um algoritmo para calcular as raízes de uma equação do 2º. grau, sendo que os valores de A , B e C são fornecidos pelo usuário. Não esquecer de verificar se o valor de Delta é maior ou igual a zero, ou seja, se existem raízes reais. (Exercício resolvido no Anexo I)

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

Para homens: $(72.7 * \text{Altura}) - 58$

Para mulheres: $(62.1 * \text{Altura}) - 44.7$

10. Leia um salário bruto e imprima o salário bruto, o desconto e salário líquido, após os descontos do imposto de renda, que será calculado da seguinte maneira:

Salário	Imposto	Dedução
Até 1.372,81	I S E N T O	
Acima de 1.372,81 até 2.743,25	15%	205,92
Acima de 2.743,25	27.5%	548,82

11. Leia um mês e ano e imprima a quantidade de dias que este mês possui. Lembre-se dos anos bissextos.
12. Leia sexo e idade de uma pessoa e imprima maior idade ou menor idade, considerando que sexo masculino e maior idade se maior igual a 21 anos e sexo feminino e maior idade se maior igual a 18 anos.
13. A loja Constrói em Partes produz dois tipos de hastes: cobre e alumínio. Cada haste de cobre é vendida por R\$ 2,00, e cada haste de alumínio é vendida por R\$ 4,00. O desconto dado dependerá da quantidade de hastes compradas, conforme tabela abaixo. Fazer um algoritmo para ler a quantidade comprada de cada tipo de haste e imprima o total pago.

Quantidade comprada (duas hastes juntas)	Percentual de Desconto
Abaixo de 5	0
Entre 5 e 15	10
Entre 16 e 20	15
Acima de 20	20

14. Uma empresa de energia elétrica trabalha com 3 tipos de consumidores: I – Industrial; C – Comercial; R – Residencial. Fazer um algoritmo para ler o tipo de consumidor ('I', 'C' ou 'R'), a quantidade de energia consumida, e calcular e imprimir qual será o valor pago. Para calcular o valor pago, verificar a tabela abaixo.

Tipo de Consumidor	Cálculo
Industrial	$0.68 * \text{Consumo} + 34$
Comercial	$0.37 * \text{Consumo} + 45$
Residencial	$0.77 * \text{Consumo} - 22$

15. Escrever um algoritmo para ler a hora de início e hora de término de um jogo, ambas expressas em horas e minutos. Calcular e escrever a duração do jogo, também em horas e minutos, considerando que o tempo máximo de duração de um jogo é de 24 horas e que o jogo pode iniciar em um dia e terminar no dia seguinte. Se o usuário informar horas maiores que 24 ou minutos maiores que 59, informar mensagem de erro: “Entrada de dados não é válida”.

Exemplo: Hora inicial do jogo: 23 Minuto inicial do jogo: 50
 Hora final do jogo: 01 Minuto final do jogo: 45
 Duração do jogo: 1 hora e 55 minutos

(Exercício resolvido no Anexo I)

16. Escreva um algoritmo que leia um número inteiro de três dígitos e calcule a soma dos seus dígitos. Verificar se o novo número formado é divisor de 16 e múltiplo de 4 ao mesmo tempo.

Exemplo: Número informado: 125

Soma dos dígitos do número 125 é igual a 8

O número informado é divisor de 16 e múltiplo de 4 ao mesmo tempo.

17. Fazer um algoritmo para ler os valores de z e w , e calcular e imprimir:

$$y = (7x^2 - 3x - 8t) / 5(x + 1)$$

sabendo-se que os valores de x são assim definidos:

$$\text{se } w > 0 \text{ ou } z < 7$$

$$x = (5w + 1) / 3;$$

$$t = (5z + 2);$$

caso contrário

$x = (5z + 2);$

$t = (5w + 1) / 3;$

3.9. Estrutura de Repetição

Nos exemplos e exercícios que vimos até agora sempre foi possível resolver os problemas com uma seqüência de instruções onde todas eram necessariamente executadas uma única vez. Os algoritmos que escrevemos eram, portanto, apenas uma seqüência linear de operações. Nesta seção veremos um conjunto de estruturas sintáticas que nos dão mais flexibilidade para determinar a seqüência de execução dos comandos da linguagem.

Quando queremos que um trecho de um algoritmo seja repetido num determinado número de vezes, utilizamos os comandos de repetição. A nossa linguagem possui três estruturas de repetição: Repita..Até, Para..Faça e Enquanto..Faça.

3.9.1. Repita..Até

Nessa estrutura todos os comandos da lista são executados e a expressão é avaliada. Isto se repete até que a avaliação da condição resulte em verdadeiro, quando então o próximo comando a ser executado é o comando imediatamente após o **até**. Cada repetição da lista de comandos também é chamada de iteração. Essa estrutura também é chamada de laço de repetição. Eis a sua forma geral:

Repita

<lista de comandos>

até <expressão lógica ou relacional>

ex.: escrever os números de 1 a 10.

```
algoritmo "repetição"
var
i:inteiro
inicio
i <- 1
repita
    escreva (i)
    i <- i + 1
ate i > 10
finalgoritmo
```

Obs.: A variável "i" controla o número de repetições do laço. Normalmente a variável de controle do laço recebe um valor inicial, é incrementada de um valor constante no laço e tem seu valor testado em algum ponto do laço. Ao chegar a um determinado valor o laço é interrompido. A inicialização da variável contadora deve acontecer fora do laço, antes do seu início.

Existem diversas maneiras de implementar o mesmo laço, mas todo laço com variável de controle deve conter:

- a) inicialização
- b) incremento($i=i+1$) ou decremento($i=i-1$)
- c) teste de valor final

Veja um outro exemplo: Escreva os números de 10 a 1.

```

algoritmo "repetição"
var
  i:inteiro
inicio
i <- 10
repita
  escreva (i)
  i <- i - 1
ate i = 0
fimalgoritmo

```

3.9.2. Enquanto..Faca

Na estrutura Enquanto..faca, a expressão lógica é avaliada e, se ela for verdadeira, a lista de comandos é executada. Isso se repete até que a condição seja falsa. Veja a sua forma geral:

Enquanto <expressão lógica ou relacional> **Faca**

<lista de comandos>

Fim Enquanto

A estrutura **enquanto...faca** também é uma estrutura de repetição, semelhante a repita. A diferença básica entre as duas estruturas é a posição onde é testada a expressão. No repita, a condição é avaliada após a execução dos comandos, o que garante que os comandos serão executados pelo menos uma vez. No enquanto, a expressão é avaliada no início e se o resultado for falso no primeiro teste a lista de comandos não é executada nenhuma vez. Essa diferença faz com que em determinadas situações o uso de uma estrutura seja mais vantajoso que o uso da outra. O exemplo a seguir, onde são mostradas soluções para um problema, utilizando as duas estruturas, ilustram essa diferença:

Problema: Faça um algoritmo que leia diversos números positivos e escreva, para cada um, o número recebido. Terminar o algoritmo quando for informado um número negativo.

```
algoritmo "teste 1"
var
  i:inteiro
inicio
  escreva("Informe um número inteiro e positivo:")
  leia(i)
  enquanto i >=0 faca
    escreval(i)
    escreva("Informe um número inteiro e positivo.
  Para sair informe -1:")
    leia(i)
  fimenquanto
finalgoritmo
```

```
algoritmo "teste 2"
var
  i:inteiro
inicio
  repita
    escreva("Informe um número inteiro e positivo:")
    leia(i)
    se i>=0 entao
      escreval(i)
    fimse
  ate i < 0
finalgoritmo
```

Neste algoritmo, se o primeiro valor for negativo, o algoritmo não deve escrever nada. Para que isso ocorra um teste do valor deve ser feito antes da escrita. Como no **Repita** o teste é feito ao final, outro teste deve ser colocado no início do laço, o que faz com que, a cada iteração, dois testes sejam feitos. Isto não ocorre no **Enquanto**, onde o teste é feito no início, não sendo, portanto, necessário um teste adicional.

3.9.3. Para..Faca

Forma geral:

Para <variável de controle> **De** <valor inicial> **Ate**<valor final> [**Passo**<incremento>] **Faca**
 <lista de comandos>
FimPara

Na estrutura Para, a variável de controle é inicializada com <valor inicial> e no início de cada iteração o valor da variável de controle é comparado com <valor final>. Se o valor da variável for

menor ou igual a <valor final>, a lista de comandos é executada e após ser executado o último comando da lista, a variável de controle é incrementada. Isto repete-se até que o valor da variável de controle seja maior que <valor final>, quando então é executado o comando imediatamente após a palavra Fim. A condição Passo não é obrigatória, mas se precisar incrementar a variável de controle você deve utilizar, por exemplo, você quer que a variável de controle pule em Dois em Dois a sintaxe ficaria assim:

```
para i de 1 ate 1000 passo 2 faca
    <lista de comandos>
fimpara
```

A estrutura Para é uma estrutura de repetição mais completa que as anteriores, pois ela incorpora a inicialização, incremento e teste de valor final da variável de controle. É preferencialmente utilizada em situações em que sabe-se previamente o número de repetições a serem feitas. Este número de repetições pode ser uma constante ou estar em uma variável.

A seguir serão apresentados alguns problemas utilizando estruturas de repetição e desenvolvidas algumas soluções para os mesmos.

Problema 1:

Faça um algoritmo que leia 5 números e escreva somente os que forem positivos.

Solução:

Neste problema, a mesma ação é repetida 5 vezes. Em cada uma delas um número é lido e, se for positivo, é escrito. Como o número de repetições é definido (5), pode-se utilizar a estrutura para.

Uma possível solução para o algoritmo é a seguinte:

```
algoritmo "teste 3"
var
numero :real
i: inteiro
inicio
para i de 1 ate 5 passo 1 faca
    escreva("Informe um
número:")
    leia (numero)
    se numero>0 entao
        escreval (numero)
    fimse
fimpara
fimalgoritmo
```

Neste algoritmo são utilizadas duas variáveis, cada uma com uma função bem definida. A variável i é usada para controlar o número de repetições e a variável número é utilizada para armazenar cada

um dos valores lidos. Ao escrever um algoritmo é importante ter bem clara a função de cada variável. Como serão lidos 5 números diferentes, a leitura do número deve ser feita dentro do laço.

Problema 2:

Faça um algoritmo que leia um número N e escreva todos os números de 1 a N.

Solução:

Neste problema, é lido um número N, e são escritos todos os números de 1 a N. Para isso deve ser utilizado uma estrutura de repetição. Como o número de repetições é conhecido (está na variável N) pode-se utilizar a estrutura Para. Uma possível solução para o problema é a seguinte:

```
algoritmo  "teste 4"
var
i,numero :inteiro
inicio
escreva("Informe um número:")
leia (numero)
para i de 1 ate numero faca
    escreval(i)
fimpara
fimalgoritmo
```

Vale observar que, como nesse algoritmo é lido apenas um número, a leitura do mesmo deve ser feita fora da estrutura de repetição.

3.10. Exercícios de Algoritmos com Repetição

- 1) Fazer um algoritmo para calcular a soma de todos os múltiplos de 7 entre 100 e 500. (Exercício resolvido no Anexo I)
- 2) Faça um algoritmo que imprima os números inteiros decrescentes compreendidos entre 100 e 10.
- 3) Fazer um algoritmo para ler 20 números inteiros e imprimir somente os lidos que ao dividir por 13 dêem resto 2.
- 4) Ler dois números inteiros N e M ($N \leq M$) e imprimir todos os números pares entre N e M.
- 5) Fazer um algoritmo para ler dois números inteiros N e M ($N < M$) e calcular e imprimir a soma de todos os números do intervalo entre N e M que não são múltiplos de 3 nem de 7. Ao final imprimir também a soma desses números. (Exercício resolvido no Anexo I)
- 6) Faça um algoritmo que imprima a soma dos números ímpares compreendidos entre 10 e 100.
- 7) Faça um algoritmo que imprima a tabuada do número 5 da seguinte maneira:

```
5 x 0=0
5 x 1=5
5 x 2=10
...
```

$$5 \times 10 = 50$$

- 8) Faça um algoritmo que imprima o valor de S da seguinte série: (Exercício resolvido no Anexo I)

$$S = \frac{1}{60} - \frac{2}{58} + \frac{3}{56} - \frac{4}{54} + \dots - \frac{30}{2}$$

- 9) Faça um algoritmo que imprima o valor de H da seguinte série:

$$H = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + \dots + \frac{99}{50}$$

- 10) Faça um algoritmo que leia um número inteiro e positivo N e imprima o valor de S da seguinte série:

$$S = \frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{N}$$

- 11) Faça um algoritmo que imprima a soma dos 20 primeiros termos da série abaixo, onde o denominador representa o fatorial do número.

$$\frac{100}{0!} + \frac{99}{1!} + \frac{98}{2!} + \frac{97}{3!} + \dots$$

- 12) Um jequitibá tem 1.50 e cresce 1 centímetro por ano. Uma jaqueira tem 1.10 e cresce 2 centímetros por ano. Uma jaboticabeira tem 2.98 e não cresce mais. Construa um programa que calcule e imprima qual árvore terá a maior altura após 35 anos.

- 13) Repita os dados do exercício anterior, e imprima após quantos anos a jaqueira será a maior das três árvores.

- 14) Fazer um algoritmo para ler o nome, sexo e altura de 50 pessoas e calcular e imprimir:

- a) A maior altura;
- b) Quantas pessoas são do sexo feminino;
- c) O nome da pessoa de menor altura (Exercício resolvido no Anexo I);

- 15) Ler dois números inteiros N e M (N < M). Calcular e imprimir a soma de todos os números do intervalo que divididos por 7 dão resto 3.

- 16) Faça um algoritmo que leia a idade e o sexo ("M", "F") de 100 pessoas. Calcular e imprimir:

- a) A maior idade;
- b) Quantos homens estão com idade maior ou igual a 45 anos;
- c) Média de altura das mulheres com mais de 30 anos;

- 17) A conversão de graus Fahrenheit para centígrados é obtida pela fórmula $C = 5/9 (F - 32)$. Escreva um algoritmo que calcule e escreva uma tabela de graus centígrados em função de graus Fahrenheit que variem de 32 a 212 de 1 em 1. A tabela deverá ser semelhante à mostrada abaixo:

Graus Fahrenheit	Graus Celsius
32	0
33	0.6
34	1.1
...	...
212	100

- 18) Implementar um algoritmo capaz de encontrar o maior e o menor dentre 20 números inteiros quaisquer. Desconsiderar números iguais.
- 19) Escrever um algoritmo que, dada uma sequência de valores inteiros e positivos, determine qual é o menor valor desta sequência e a média aritmética dos valores pares. O valor 0 (zero) indica o término dos dados de entrada, ou seja, o programa termina quando for informado o valor 0(zero).
- 20) Em uma cidade do interior, sabe-se que, de janeiro a abril de 1976(121 dias), não ocorreu temperatura inferior a 15°C nem superior a 40°C. Fazer um algoritmo leia a temperatura ocorrida dia-a-dia, calcular e imprimir:
- A menor temperatura ocorrida;
 - A maior temperatura ocorrida;
 - A temperatura média; (Exercício resolvido no Anexo I)
- 21) Num frigorífico existem diversos bois. Cada boi traz preso no seu pescoço um cartão contendo um número de identificação e seu peso. Implementar um algoritmo que escreva o número de identificação e o peso do boi mais gordo e do boi mais magro. Sair do algoritmo quando for informado o valor -1 no número de identificação do boi.
- 22) Escrever um algoritmo que leia o nome, sexo, altura e idade de um grupo de 50 pessoas e informe:
- Média das alturas dos homens;
 - Média de idades das mulheres;
 - Nome da pessoa de maior altura;
- 23) Escrever um algoritmo que lê vários valores reais, um de cada vez, e calcular o percentual de números negativos em relação ao total de números informados. Imprimir o resultado final. Sair do algoritmo quando for informado o valor 0(zero) no valor lido. O percentual de números negativos é calculado dividindo-se a quantidade de números negativos informados pela quantidade total de números. (Exercício resolvido no Anexo I)
- 24) A série de Fibonacci tem como valores iniciais os dois primeiros termos da série que são respectivamente 0 e 1. A partir deles os demais termos são construídos pela regra abaixo. Escrever um algoritmo que gera os 15 primeiros termos da série de Fibonacci e calcula e escreve a soma destes termos. $t_n = t_{n-1} + t_{n-2}$
- 25) Escrever um algoritmo que lê um valor n e outro valor m, e calcula a tabuada de n de 1 até m. Exemplo: Supor que o usuário informe os seguintes valores: n=8 e m=12.

$$1 \times 8 = 8$$

$$2 \times 8 = 16$$

...

...

$$12 \times 8 = 96$$

- 26) Um prédio comercial tem três elevadores, denominados A, B e C. Para otimizar o sistema de controle dos elevadores, foi realizado um levantamento estatístico com 100 usuários, onde cada usuário respondia: O elevador que utilizava com maior frequência; o andar ao qual se dirigia; o período que utilizava o elevador, entre M = matutino, V = vespertino e N = noturno;

Construa um algoritmo que calcule e imprima:

- ✓ Quantos usuários que utilizaram o elevador A foram para o décimo andar;
 - ✓ Qual é o elevador mais frequentado e em que horário se encontra seu maior fluxo;
 - ✓ Quantos usuários utilizam os elevadores que estão do primeiro ao quinto andar.
- 27) Faça um algoritmo que leia um conjunto de valores que representam a idade de pessoas e imprima a maior e a menor idade. O último valor informado não é válido e será o finalizador do programa.
- 28) Faça um algoritmo que leia a altura de moças inscritas em um concurso de beleza. Para finalizar será digitado zero na altura. Imprima as duas maiores alturas.
- 29) Faça um algoritmo que leia a altura de 100 pessoas e imprima a maior, a menor, a média destas alturas. Imprima também quantas alturas maiores que 1.60 e quantas menores que 1.30.
- 30) Faça um algoritmo que leia os dados de cada empregado: nome, salário por dia e número de dias trabalhados de uma firma de 200 empregados. Imprima o salário a ser pago a cada empregado, o salário total a ser pago aos empregados e a média dos salários.
- 31) Faça um algoritmo que leia a altura e o sexo de 100 pessoas e imprima:
- Quantos homens e mulheres foram medidas;
 - Quantos homens acima de 1,70;
 - Percentual entre as mulheres que possuem altura entre 1.50 e 1.90.
 - A média das alturas das mulheres.
- 32) Deseja-se fazer uma pesquisa a respeito do consumo mensal de energia elétrica em uma determinada cidade. Para isso, são fornecidos os seguintes dados: Preço do KWH consumido, Número do consumidor, Quantidade de KWH consumidos durante o mês e o Código do tipo de consumidor (R para residencial , C para Comercial e I para industrial). O Número do consumidor igual a zero deve ser usado para término do algoritmo. Ler os dados acima e imprimir:
- Para cada consumidor, o seu número e o total a pagar;
 - Maior e o menor consumo verificado;
 - Total do consumo para cada um dos três tipos de consumidores;
 - A média de consumo.

3.11. Variáveis Compostas Homogêneas

Vimos, no início deste curso, ser possível dar um Nome para uma posição de memória, sendo que a esta memória será associada a um valor qualquer. Pois bem, acontece que, muitas vezes, esta forma de definição, ou, melhor dizendo, de alocação de memória, não é suficiente para resolver certos problemas computacionais. Imagine por Exemplo, como faríamos para construir um algoritmo, para ler 1000 números e que imprimisse um relatório destes mesmos números, mas ordenados alfabeticamente? Não seria uma tarefa simples, como é mostrado abaixo:

```
algoritmo "loucura"
var
    num1, num2, num3, num4, num5, num6, num7, num8,... num1000: inteiro
inicio
    leia(num1,num2,num3,num4,..,num1000)
:
:
fimalgoritmo
```

Considere o tamanho do algoritmo, e o trabalho braçal necessário para construí-lo. Isto só com 1.000 Números, imagine agora 1.000.000 de números. A construção deste algoritmo começaria a ficar inviável na prática. Para resolver problemas como este, e outros, foi criado um novo conceito para alocação de memória sendo, desta forma, também criado uma nova maneira de definirem variáveis, a qual foi denominada de variável indexada.

Uma variável indexada corresponde a uma sequência de posições de memória, a qual será dada um único nome, sendo que cada uma destas pode ser acessada através do que conhecemos por índice. O índice corresponde a um valor inteiro, ou a um valor caractere (exceto CADEIA). Cada uma das posições de memória de uma variável indexada pode receber valores no decorrer do algoritmo como se fosse uma variável comum, a única diferença reside na Sintaxe de utilização desta variável.

3.11.1. Variáveis Indexadas Unidimensionais (Vetores)

Também conhecida por "Vetor". Uma variável unidimensional, como o próprio nome já indica, possui apenas uma dimensão, sendo possível definir variáveis com quaisquer tipos de dados.

3.11.1.1. Declaração de Vetores

Sintaxe:

<identificador>: **VETOR** [<tamanho>] <tipo (**numérico, lógico, caracter ou cadeia**)>

Ex.: idades: vetor [5] de inteiro

1.1.1.1 Atribuindo Valores a Elementos do Vetor

Para se atribuir um valor a um elemento do vetor devemos utilizar o seguinte padrão:

Sintaxe:

```
< identificador>[<posição>] <- <valor>
```

Exemplo :

```
idades[3] <- x
idades[1] <- 2
idades[i] <- i^2
```

Problema 1 - Escrever um algoritmo que lê um vetor V de 6 posições e o escreve. Conte, a seguir, quantos valores de V são negativos e escreva esta informação.

Solução:

Neste problema, a mesma ação é repetida 6 vezes e mostra os valores lidos, logo a seguir pede para dizer quantos desses números são negativos. Para isso deve ser utilizada uma estrutura de repetição para receber os valores lidos e uma condição para contar quantos números são negativos. Uma possível solução para o algoritmo é a seguinte:

```
Algoritmo conta_vetor
  vet: vetor[1..6] de inteiro
  i, conta_neg: inteiro
Início
  conta_neg <- 0
  Para i de 1 ate 6 faça
    leia(vet[i])
    se vet[i]<0 entao
      conta_neg <- conta_neg + 1
    fimse
  fimpara
  para i de 1 ate 6 faça
    escreval(vet[i])
  fimpara
  escreva("total de números negativos: ", conta_neg)
Fimalgoritmo
```

3.11.1.2. Exercícios sobre Vetores

- 1) Declarar um vetor de 100 posições e preenchê-lo com 0 nas posições pares e 1 nas posições ímpares. Imprimir o vetor gerado. Não é necessário leitura de dados.
- 2) Dados 2 vetores V1 de 5 elementos e V2 de 10 elementos. Ler os dois vetores e gerar um terceiro vetor V3 de 15 elementos, cujas 5 primeiras posições devem ser preenchidas com os elementos de V1 e as outras 10 devem ser preenchidas com os elementos de V2.
- 3) Implementar um algoritmo que leia 50 notas de provas e as respectivas matrículas dos indivíduos, e calcule e imprima:
 - a) A média das notas;
 - b) Quantas notas estão acima e abaixo da média calculada no item a (não considerar notas iguais à média);

- c) Quantas pessoas possuem a maior e a menor das notas, e quais são elas (isto é, suas matrículas).
- 4) Leia um vetor de 10 elementos e em seguida informe um valor m. Ache a posição do valor m (dado pelo usuário) no vetor. Caso o valor informado não exista no vetor, informe ao usuário.
- 5) Faça um algoritmo que leia 20 números inteiros e apresente-os em ordem crescente.
- 6) Leia um vetor com 20 elementos. A seguir, troque o primeiro elemento com o último, o segundo com o penúltimo, etc..., até o décimo com o décimo primeiro.
- 7) Leia um vetor de 12 posições e em seguida ler também dois valores X e Y quaisquer correspondentes a duas posições no vetor. Ao final seu programa deverá escrever a soma dos valores encontrados nas respectivas posições X e Y.
- 8) Leia um vetor de 40 posições de inteiros e acumule os valores do primeiro elemento no segundo, deste no terceiro e assim por diante. Ao final, escreva o vetor lido e o resultante. (Exercício resolvido no Anexo I)

Vetor lido:

5	-1	3	9	17	6	1	8
---	----	---	---	----	---	---	---	-----	-----

Vetor resultante:

5	4	7	16	33	39	40	48
---	---	---	----	----	----	----	----	-----	-----

- 9) Fazer um algoritmo para ler 80 valores inteiros entre 0 e 10, e responder qual é a frequência absoluta e a frequência relativa. Frequência absoluta é a quantidade de cada valor. Frequência relativa é a quantidade de cada valor dividido pela quantidade total de valores. (Exercício resolvido no Anexo I)
- 10) Dada uma sequência numérica de 50 elementos armazenados num vetor, determinar o maior elemento do conjunto e o seu índice (posição). Suponha que não existam elementos repetidos no vetor.

3.11.1.3. Ordenação de Vetores

Ordenar um vetor é colocar seus componentes sob algum critério de ordenação (ascendente ou descendente). Existem vários algoritmos de ordenação. Um método simples de ordenação é comparar cada elemento com todos os outros elementos a partir dele. Por exemplo, para a ordenação de um vetor de N elementos:

```

Para I de 1 ate (N-1) faça
  Para J de (I+1) ate N faça
    Se V[J] > V[I] entao
      AUX <- V[I]
      V[I] <- V[J]
      V[J] <- AUX
    Fimse
  Fimpara
Fimpara

```

Existem vários algoritmos de ordenação, cada um apresentando maior ou menor eficiência, por exemplo Bubble Sort, Quick Sort, HeapSort, Shell Sort etc...

3.11.2. Variáveis Indexadas Multidimensionais (Matrizes)

Também conhecida por "Matriz". Uma variável Multidimensional, como o próprio nome já indica, possui duas ou mais dimensões, sendo ser possível definir variáveis com quaisquer tipo de dados válidos.

3.11.2.1. Atribuindo Valores a Elementos da Matriz bidimensional

Problema 1 - Escreva um algoritmo que lê uma matriz $M(3,3)$ e calcula as somas:

- a) da linha 3 de M.
- b) da coluna 2 de M.
- c) da diagonal principal.
- d) da diagonal secundária.
- e) de todos os elementos da matriz.

Solução:

Neste problema, temos uma matriz quadrada de ordem três por três, sendo $3 \times 3 = 9$ ações que são repetidas, logo a seguir o problema solicita a soma da linha três da matriz $M[3, \text{quantidade de colunas}]$, soma da coluna dois $M[\text{quantidade de linhas}, 2]$, diagonal principal onde os índices da coluna e linha sempre são iguais, soma da diagonal secundária onde o índice da coluna vai ser a ordem da matriz quadrada mais um e menos o índice da linha e a soma de todos os elementos da matriz. Para isso deve ser utilizada uma estrutura de repetição para receber os valores lidos e algumas condições para verificar as questões pedidas. Uma possível solução para o algoritmo é a seguinte:

```
ALGORITMO "Matrizes"
var
  mat: VETOR[1..3,1..3] de Inteiro
  i, j, somaLinha3, somaColuna2: Inteiro somaDiagPrinc, somaDiagSecu, somaTudo:
  Inteiro
Inicio
  somaLinha3 <- 0
  somaColuna2 <- 0
  somaDiagPrinc <- 0
  somaDiagSecu <- 0
  somaTudo <- 0
```

```

para i de 1 ate 3 faca
  para j de 1 ate 3 faca
    leia(mat[i,j])
    somatudo <- mat[i,j] + somatudo
    se i=3 entao
      somalinha3 <- mat[i,j]+ somaLinha3
    fimse
    se j=2 entao
      somaColuna2 <- mat[i,j]+ somaColuna2
    fimse
    se i=j entao
      somaDiagPrinc <- mat[i,j]+ somaDiagPrinc
    fimse
    se j=4-i entao
      somaDiagsecu <- mat[i,j]+ somaDiagsecu
    fimse
  fimpara
fimpara
para i de 1 ate 3 faca
  para j de 1 ate 3 faca
    escreva (mat[i,j])
  fimpara
fimpara
escreval("soma de todos os elementos é ", somatudo)
escreval("soma dos elementos da linha 3 é ", somalinha3)
escreval("soma dos elementos da coluna 2 é ", somacoluna2)
escreval("soma dos elementos da diagonal principal é ", somadiagprinc)
escreval("soma dos elementos da diagonal secundária é ", somadiagsecu)
finalgoritmo

```

3.11.2.2. Exercícios

- 1) Faça um algoritmo que leia uma matriz 4 X 4 por linha e imprima a soma de todos os elementos da matriz superior.
- 2) Faça um algoritmo que leia uma matriz 7 X 7 por linha e imprima somente a matriz triangular inferior.(Exercício resolvido no Anexo I)
- 3) Faça um algoritmo que leia 5 notas dadas a cada aluno durante o período letivo em uma turma de 30 alunos. Calcule a média de cada aluno, a média geral da turma, e imprima de acordo com o seguinte layout:

Núm. Aluno	Notas						Média Aluno
1	9.9	9.9	9.9	9.9	9.9	9.9	9.9
....
....
....

Média Geral da Turma: 9.9

- 4) Na Teoria de Sistemas define-se como elemento minimax de uma matriz o menor elemento da linha em que se encontra o maior elemento da matriz. Escreva um programa que preencha uma matriz M(15,15) por leitura e determine o seu elemento minimax.

5) Implementar um algoritmo para multiplicar duas matrizes de números inteiros. A multiplicação só é possível se o número de colunas da matriz A for igual ao número de linhas da matriz B. Supor a Matriz A sendo 3×4 e a matriz B é 4×3 . (Exercício resolvido no Anexo I)

Anexo I – Lista de exercícios resolvidos

3.12. Exercícios da Seção 3.3.7

Exercício 8

```

algoritmo "Calcula distância"
// Função : Calcula a distância entre dois pontos P1(x1,y1) e P2(x2,y2)
// Autor :
// Data : 4/2/2010
// Seção de Declarações
var
x1,y1,x2,y2,d:real
inicio
Escreva("Informe as coordenadas do primeiro ponto:")
leia(x1,y1)
Escreva("Informe as coordenadas do segundo ponto:")
leia(x2,y2)
d<-raizq((x2-x1)^2+(y2-y1)^2) //Calcula a distância entre os pontos
Escreva("A distância entre o ponto (",x1,",",y1,) e o ponto
(",x2,",",y2,) é ",d:4:2)
Fimalgoritmo

```

3.13. Exercícios da Seção 3.8

Exercício 6

Leia uma média e um número de faltas dadas a um aluno e imprima aprovado, reprovado por média e por falta, reprovado por média ou reprovado por falta. Sendo média maior igual a 7 indica aprovado, e faltas maior igual a 32 indica reprovado.

```

algoritmo "Verifica aprovação do aluno"
// Função : Ler média e um número de faltas dadas a um aluno e verifica
//aprovação ou reprovação
var
media,nota1,nota2:real
faltas:inteiro
inicio
escreva("Informe as duas notas do aluno:")
leia(nota1,nota2)
escreva("Informe o número de faltas:")
leia(faltas)
media<-(nota1+nota2)/2
Escreva("A média do aluno foi ",media:6:1," e ",faltas," faltas,")
se (media >= 7) e (faltas < 32) entao
    Escreva(" e o aluno foi APROVADO.")
senao
    se (media < 7) e (faltas < 32) entao
        Escreva(" e o aluno foi Reprovado por Nota.")
    senao
        se (media >= 7) e (faltas > 32) entao
            Escreva(" e o aluno foi Reprovado por Nota.")
        senao
            se (media < 7) e (faltas > 32) entao
                Escreva(" e o aluno foi Reprovado por Nota e Faltas.")

```

```

        fimse
    fimse
fimse
fimalgoritmo

```

Exercício 8

Construa um algoritmo para calcular as raízes de uma equação do 2º grau, sendo que os valores de A, B e C são fornecidos pelo usuário. Não se esqueça de verificar se o valor de Delta é maior ou igual a zero, ou seja, se existem raízes reais.

```

algoritmo "Calcula raizes"
// Função : Calcular as raízes de uma equação de 2o. grau, sendo os
coeficientes
// da equação  $Ax^2 + Bx + C = 0$  fornecidos pelo usuário
// Autor : Edilson Luiz do Nascimento
// Data : 16/2/2008
// Seção de Declarações
// A, B, C - Coeficientes da equação de primeiro grau, fornecidos pelo
usuário
// Delta - Calcula o valor de Delta
// X1, X2 - Raízes da equação
var
A,B,C,Delta,X1,X2:Real    //Declaração das variáveis
inicio
Escreva("Informe os valores dos coeficientes da equação:")
Leia(A,B,C)              //Entrada de dados
Delta <- Quad(B) - 4*A*C  //Cálculo do valor de delta
Se Delta > 0 entao        //Serão duas raízes reais
    X1<- (-B + Raizq(Delta))/(2*A)
    X2<- (-B - Raizq(Delta))/(2*A)
    Escreva("As raízes serão:", X1:6:1, X2:6:1)
senao
    Se Delta = 0 entao    //Uma única raiz real
        X1<- (-B + Raizq(Delta))/(2*A)
        Escreva("A raiz será:", X1:6:1)
    senao
        Escreva("Não há raízes reais")
    fimse
fimse
fimalgoritmo

```

Exercício 15

Escrever um algoritmo para ler a hora de início e hora de término de um jogo, ambas expressas em horas e minutos. Calcular e escrever a duração do jogo, também em horas e minutos, considerando que o tempo máximo de duração de um jogo é de 24 horas e que o jogo pode iniciar em um dia e terminar no dia seguinte. Se o usuário informar horas maiores que 24 ou minutos maiores que 59, informar mensagem de erro: “Entrada de dados não é válida”.

Exemplo: Hora inicial do jogo: 23 Minuto inicial do jogo: 50
 Hora final do jogo: 01 Minuto final do jogo: 45
 Duração do jogo: 1 hora e 55 minutos

Observação: Neste problema temos que analisar vários casos:

1 - Hora início do jogo menor que Hora Fim do jogo

1.1 - Minuto Início do jogo maior que minuto fim do jogo

Exemplo 1: 8:40H até 10:30H

Nesse caso teremos que subtrair os minutos, mas "pegando emprestado" uma hora para que a subtração fique correta, ou seja, acrescentamos 60 aos minutos finais e subtraímos em um da hora final;

1.2 - Minuto Início do jogo menor que minuto fim do jogo;

Exemplo 2: 8:40H até 10:50H

Nesse caso fazemos a subtração dos minutos e das horas normalmente.

2 - Hora início do jogo maior que Hora Fim do jogo

2.1 - Minuto Início do jogo menor que minuto fim do jogo

Exemplo 3: 22:30 até 1:40

Neste caso o jogo passou para o dia seguinte. No caso dos minutos, a subtração ocorre normalmente e somamos 24 a hora final para podermos fazer a subtração correta.

2.2 - Minuto Início do jogo maior que minuto fim do jogo

Exemplo 4 - 22:50 até 1:40

Neste caso o jogo passou para o dia seguinte, e, além disso, os minutos do fim de jogo são menores que os minutos de início. Nesse caso teremos que subtrair os minutos, mas "pegando emprestado" uma hora para que a subtração fique correta, ou seja, acrescentamos 60 aos minutos finais e subtraímos em um da hora final. Além disso, Assim, somamos 24 na hora final para podermos fazer a subtração correta.

```
algoritmo "Calcula duração do jogo"
// Função : Lê hora início e término do jogo, e calcula a sua duração
// Autor : Edilson Luiz do Nascimento
// Data : 16/2/2008
//HoraIni - Hora de início do jogo
//HoraFim - Hora de fim do jogo
//MinIni - Minuto de início do jogo
//MinFim - Minuto de fim do jogo
//DurHoras - Duração do jogo em Horas
//DurMin - Duração do jogo em Minutos

var
HoraIni, HoraFim, MinIni, MinFim, DurHoras, DurMin: Inteiro
inicio
Escreva("Escreva o horário do início do jogo, em horas e minutos:")
Leia(HoraIni, MinIni)
Escreva("Escreva o horário de Fim do jogo, em horas e minutos:")
Leia(HoraFim, MinFim)
//Item 1 - Hora início do jogo menor que hora fim
Se HoraIni < HoraFim então
//Item 1.1 - Minuto início do menor maior que minuto fim
    Se MinIni > MinFim então
        DurMin <- MinFim - MinIni + 60 // "pega emprestado" uma hora
//Subtraindo de um, pois os minutos "pegaram emprestado" uma hora
        DurHoras <- HoraFim - HoraIni - 1
    Senão
//Item 1.2 - Minuto início menor ou igual ao minuto fim do jogo
    Se MinIni <= MinFim então
```

```

        DurMin <- MinFim - MinIni      // Calcula normalmente os minutos
        DurHoras <- HoraFim - HoraIni //Calcula normalmente as horas
    fimse
    fimse
senao
    //Item 2 - Hora início do jogo maior ou igual à hora fim
    Se HoraIni >= HoraFim entao
    //Item 2.1 - Minuto início do menor maior que minuto fim
        Se MinIni > MinFim entao
            DurMin <- MinFim - MinIni + 60 //"pega emprestado" uma hora
//subtrai de um, pois os minutos "pegaram emprestado" uma hora, além do
//jogo ter terminado no dia seguinte
            DurHoras <- HoraFim - HoraIni - 1 + 24
        Senao
//Item 2.2 - Minuto início do menor maior que minuto fim
        Se MinIni <= MinFim então
            DurMin <- MinFim - MinIni // Calcula normalmente os minutos
            DurHoras <- HoraFim - HoraIni + 24
        fimse
    fimse
    fimse
    escreval //Deixar uma linha em branco
    //Impressão da duração do jogo
    escreval("A duração do jogo foi de ",DurHoras," Horas e ",DurMin,"
    minutos.")
finalgoritmo

```

3.14. Exercícios da Seção 3.11

Exercício 1

```

algoritmo "Calcula Soma"
// Função : Calcula e imprime a soma dos múltiplos de 7 entre 100 e 500
// Autor : Edilson Luiz do Nascimento
// Seção de Declarações
//I - Variável de controle da repetição
//Soma - Acumula a soma dos múltiplos de 7 no intervalo especificado
var
I,Soma:Inteiro

inicio
Soma<-0
Para I de 100 ate 500 faca
    Se I%7=0 entao      //Captura os múltiplos de 7
        Soma<-Soma+I    //Acumula os múltiplos de 7
    fimse
Fimpara
Escreva("A soma de todos os múltiplos de 7 entre 100 e 500 é ",Soma)
finalgoritmo

// Outra solução
inicio
Soma<-0
//Inicia com o primeiro múltiplo de 7 após 100, e conta de 7 em 7
Para I de 105 ate 500 passo 7 faca
    Soma<-Soma+I

```



```
Fimpara
Escreva("A soma de todos os múltiplos de 7 entre 100 e 500 é ",Soma)
Fimalgoritmo
```

Exercício 5

```
algoritmo "Soma de Números"
var
N,M,Soma,I: Inteiro
inicio
Soma<-0
Escreva("Informe 2 números inteiros:")
Leia(N,M) // Entrada de dados
Se N<M entao // Verifica se os valores estão corretos
    Para I de N ate M faca
        Se (I%3<>0) e (I%7<>0) entao // Não é múltiplo de 3 nem de 7
            Soma<-Soma+I
        Fimse
    Fimpara
    Escreval("A Soma dos números do intervalo é",Soma)
senao
    Escreva("Erro. O primeiro valor deverá ser menor que o segundo.")
fimse
fimalgoritmo
```

Exercício 8

```
algoritmo "Calcula termos da série"
// Função : Calcular a soma da série:  $S = \frac{1}{60} - \frac{2}{58} + \frac{3}{56} - \frac{4}{54} + \dots - \frac{30}{2}$ 
var
//I - Variável de controle da repetição
//Num - Variável que assume os valores do Numerador
//Den - Variável que assume os valores do Denominador
//Soma - Acumula os valores das frações geradas
I,Num,Den, TAM: Inteiro
Soma: Real
inicio
//Inicialização das variáveis
Num<-1
Den<-60
Soma<-1/60
TAM<- 30
Para I de 2 ate TAM faca
    Se I%2=0 entao
        Soma<-Soma - (Num/Den) //Parcelas negativas das frações
    senao
        Soma<-Soma + (Num/Den) //Parcelas positivas das frações
    Fimse
    Num<-Num+1 //Avança o numerador
    Den<-Den-2 //Avança o denominador
Fimpara
Escreva("A soma da Série é ",Soma:4:2) //Duas casas após a vírgula
Fimalgoritmo
```

Exercício 14

```

algoritmo "Estatísticas de altura"
// Função : Ler o nome, sexo e altura de 50 pessoas e calcular e
imprimir:
//a) A maior altura;
//b) Quantas pessoas são do sexo feminino;
//c) O nome da pessoa de menor altura
// Seção de Declarações
// nome_malt - Nome da pessoa de menor altura
// maior - maior altura
// contf - Conta a quantidade de pessoas do sexo feminino

var
nome,nome_malt,sexo:caracter
maior,menor,altura:real
contf,i:inteiro
inicio
contf<-0
para i de 1 ate 5 faca
    Escreva("Informe a altura, o nome e o sexo('M','F'):")
    Leia(altura,nome,sexo)
    Se i=1 entao //Inicialização de variáveis
        maior<-altura
        nome_malt<-nome
        menor<-altura
    senao
        Se altura>maior entao //Maior altura
            maior<-altura
        senao
            se altura<menor entao //Nome da pessoa de menor altura
                menor<-altura
                nome_malt<-nome
            fimse
        fimse
    fimse
    Se (sexo="F") ou (sexo="f") entao //Número pessoas do sexo feminino
        contf<-contf+1
    fimse
fimpara
Escreval("Maior altura:",maior)
escreval(contf," pessoas do sexo feminino")
escreval("Nome da pessoa de menor altura;",nome_malt)
finalgoritmo

```

Exercício 20

Em uma cidade do interior, sabe-se que, de janeiro a abril de 1976(121 dias), não ocorreu temperatura inferior a 15°C nem superior a 40°C. Fazer um algoritmo que calcule e imprima:

- a) A menor temperatura ocorrida;
- b) A maior temperatura ocorrida;
- c) A temperatura média;

Algoritmo "Temperaturas"

//Função: Ler 121 temperaturas, e calcular a maior, menor e temperatura média

//Autor: Edilson Luiz do Nascimento

//Data: 16/02/2008

//Temp - Temperatura a ser lida

//Maior - Armazena a maior temperatura

//Menor - Armazena a maior temperatura

//Soma - Acumula a leitura das temperaturas, para posterior cálculo da Temperatura média

//TempMedia - Temperatura média do período

//primeiro - Auxilia na inicialização da maior e menor temperatura com a primeira temperatura lida

//I - índice da variável de controle da repetição

var

primeiro:logico

I,Tam:inteiro

Temp,Maior, Menor, Soma, TempMedia:Real

Inicio

Soma <- 0

TempMedia <- 0

Tam <-3

primeiro <- verdadeiro

Para I de 1 ate Tam faca

 Escreva("Informe a ",I,"ª temperatura:")

 Leia(Temp)

 Enquanto (Temp < 15) ou (Temp > 40) faca

 Escreval("Erro. Não ocorreu temperaturas menores que 15°C nem maiores que 40°C")

 Escreva("Informe a ",I,"ª temperatura correta: ")

 Leia(Temp)

 Fimenquanto

 //Acumula as temperaturas de todos os dias para o cálculo da média

 Soma <- Soma + Temp

 Se primeiro entao

 {Inicialização das variáveis Maior e Menor}

 Maior <- Temp

 Menor <- Temp

 primeiro <- falso

 fimse

 Se Temp > Maior entao {Verificação da Maior temperatura do período}

 Maior <- Temp

 Fimse

 Se Temp < Menor entao {Verificação da Menor temperatura do período}

 Menor <- Temp

 Fimse

Fimpara

TempMedia <- Soma/Tam

Escreval("A menor temperatura no período foi:", Menor:6:2)

```

Escreval("A maior temperatura no período foi:", Maior:6:2)
Escreva("A temperatura média do período foi de:",TempMedia:6:2)
Fimalgoritmo

```

Exercício 23

```

algoritmo "Calcula percentual de negativos"
// Função : Ler diversos números reais e imprimir o percentual de
// números negativos em relação ao total de números lidos. Sair quando
// for informado o valor zero.
// Autor : Edilson Luiz do Nascimento
// Data : 18/2/2008
// Seção de Declarações
var
    num,perc_neg:real
    cont_neg, cont_num: inteiro
inicio
//Inicialização das variáveis com zero
cont_num<-0
cont_neg<-0
Escreva("Informe um número real. Para sair informe 0(zero):")
leia(num)
enquanto num <> 0 faca // repetição até encontrar valor informado zero
    cont_num <- cont_num+1 //Conta a quantidade de números lidos
    se num<0 entao //Verifica se o número lido é igual a zero, contando-o
        cont_neg <- cont_neg +1
    fimse
    Escreva("Informe um número real. Para sair informe 0(zero):") //Lê
novo número
    leia(num)
fimenquanto
perc_neg <- (cont_neg/cont_num)*100 //Calcula o percentual de números
negativos
escreva("Existem ",perc_neg:4:1,"% de números negativos.") //Mostra
// resultados finais, alinhando 4 posições e 1 casa após a vírgula
fimalgoritmo

```

3.15. Exercícios da Seção 3.11.1.2

Exercício 8

```

algoritmo "Acumula valores em vetor"
// Função : Leia um vetor de 40 posições de inteiros e acumule os
// valores do primeiro elemento no segundo, deste no terceiro e assim
// por diante. Ao final, escreva o vetor lido e o resultante.
var
lido,result:vetor[1..40] de inteiro
i:inteiro
inicio
Escreva("informe os valores do vetor:")
para i de 1 ate 40 faca //Ler o vetor
    Leia(lido[i])
fimpara
result[1]<-lido[1] //Inicializa o vetor resultante
para i de 2 ate 40 faca
    result[i]<-result[i-1]+lido[i] //Calcula o vetor resultante
fimpara

```

```

Escreva("Vetor lido:")
para i de 1 ate 40 faca
    escreva(lido[i]) //Imprime o vetor lido
fimpara
escreval // pular uma linha
escreva("Vetor resultante:")
para i de 1 ate 40 faca
    escreva(result[i]) //Imprime o vetor resultante
fimpara
fimalgoritmo

```

Exercício 9

Fazer um algoritmo para ler 80 valores inteiros entre 0 e 10, e responder qual é a frequência absoluta e a frequência relativa. Frequência absoluta é a quantidade de cada valor. Frequência relativa é a quantidade de cada valor dividido pela quantidade total de valores.

```

Algoritmo "Frequências"
//Função: Ler 80 valores inteiros e calcular a frequência absoluta e
relativa.
var
//Declaração do vetor de Notas
Nota: Vetor[1..10] de Inteiro
//Declaração do vetor de Frequência absoluta
F: Vetor[0..10] de Inteiro
I, TAM : Inteiro
inicio
TAM<-10
Escreva("Informe as Notas:") {Leitura das notas}
Para I de 1 ate TAM faca
    Leia(Nota[I])
Fimpara
Para I de 0 ate TAM faca      {Inicialização da variável que vai
acumular a frequência absoluta}
    F[I] <- 0
Fimpara

{Cálculo da Frequência Absoluta das notas - Observar que o conteúdo do
vetor nota é o índice do vetor de frequência absoluta}
Para I de 1 ate TAM faca
    F[Nota[I]] <- F[Nota[I]] + 1
Fimpara

{Impressão da frequência absoluta e frequência relativa}
Para I de 0 ate 10 faca
    Se F[I]<>0 entao      //Imprime apenas as frequências diferentes de
zero
        Escreval("A frequência absoluta da nota:", I, " é ", F[I])
        Escreval("A frequência relativa da nota:", I, " é ",
(F[I]/TAM*100):0:1," %")
        Escreval
    fimse
Fimpara
Fimalgoritmo

```

3.16. Exercícios da Seção 3.12.2.2

Exercício 2

```

algoritmo "Matriz triangular inferior"
// Função : Faça um algoritmo que leia uma matriz 7 X 7 de inteiros por
// linha e imprima somente a matriz triangular inferior.
var
mat:vetor[1..7,1..7] de inteiro
lin,col:inteiro
inicio
para lin de 1 ate 7 faca // Leitura da matriz
    para col de 1 ate 7 faca
        leia(mat[lin,col])
    fimpara
fimpara

para lin de 1 ate 7 faca
    para col de 1 ate 7 faca
        se lin+col>7 entao //Pertence a matriz triangular inferior
            escreva(mat[lin,col]:5)
        senao
            escreva("      ") // Preenche a matriz com brancos
        fimse
    fimpara
    escreval //Pula uma linha a cada linha impressa da matriz
fimpara
fimalgoritmo

```

Exercício 5

```

algoritmo "Multiplicação de matrizes"
// Função : Multiplicação de uma matriz A(3x4) por uma matriz B(4x3)
var
i,j,x,linA,linC,colB,colC:inteiro
A:vetor[1..3,1..4] de inteiro
B:vetor[1..4,1..3] de inteiro
C:vetor[1..3,1..3] de inteiro
inicio
Escreval("Informe os dados da matriz A:")
para i de 1 ate 3 faca //Leitura da matriz A
    para j de 1 ate 4 faca
        Escreva("Linha ",i," coluna ",j,":")
        leia(A[i,j])
    fimpara
fimpara
Escreval
Escreval("Informe os dados da matriz B:")
para i de 1 ate 4 faca //Leitura da matriz B
    para j de 1 ate 3 faca
        Escreva("Linha ",i," coluna ",j,":")
        leia(B[i,j])
    fimpara
fimpara
linA<-3
colB<-3
Para i de 1 ate linA faca

```

```
para j de 1 ate colB faca
  C[i,j]<-0
  Para x de 1 ate colB faca
    C[i,j]<-C[i,j]+A[i,x]*B[x,j]
  fimpara
fimpara
fimpara
linC<-linA
colC<-colB
//Impressão da matriz final
para i de 1 ate linC faca //Leitura da matriz B
  para j de 1 ate colC faca
    Escreva(C[i,j]:5)
  fimpara
  escreval
fimpara
finalgoritmo
```