Introdução à Lógica
e
Técnicas de Programação

agosto de 13

Sumário

1. Teoria dos Conjuntos

A teoria dos conjuntos foi estabelecida por Georg Ferdinand Ludwig Philip Cantor (1845–1918), nascido na Rússia, onde estudou física e matemática e doutorou-se na Alemanha em 1867, sobre a Teoria dos Números.

1.1 Conjunto

Intuitivamente um conjunto é uma coleção de objetos(não repetidos). Os objetos são os elementos do conjunto e podem ser expressos sob a forma de números, cores, pessoas etc.

Em geral, os conjuntos são expressos por letras maiúsculas do alfabeto(A, B, C...) e os elementos de um conjunto são representados por letras minúsculas(a, b, c,...)

Exemplos

1) Conjunto das vogais — Elementos: a, e, i, o, u
2) Conjunto dos pares compreendidos entre 9 e 17. — Elementos: 10, 12, 14, 16

Para indicarmos se um certo elemento pertence ou não pertence a um determinado conjunto, utilizamos, respectivamente, os símbolos $\in$ e $\notin$.

Importante: **Os símbolos $\in$ e $\notin$ relacionam elementos com conjuntos**.

1.2 Determinação de um conjunto

Um conjunto está bem determinado quando conhecemos todos os seus elementos, ou quando conhecemos a regra que determina a composição de seus elementos. Um conjunto pode ser representado de dois modos:

1.2.1 Por Extensão

Quando é possível escrever todos os seus elementos.

Exemplos

Conjunto das vogais do alfabeto $\rightarrow$ A = {a, e, i, o, u}
Conjunto dos pares maiores que 3 $\rightarrow$ P = {4, 6, 8,...}

1.2.2 Por Compreensão

Quando conhecemos uma propriedade característica comum a todos os elementos do conjunto.

Exemplos

Conjunto das vogais do alfabeto $\rightarrow$ A = {x | x é vogal}
Conjunto dos pares $\rightarrow$ P = {x | x é par}

> Se todos os elementos de A satisfazem a propriedade **p**, podemos escrever A = {x | x tem a propriedade p}

1.3 Conjunto Unitário e Conjunto Vazio

Conjunto Unitário é o que tem um único elemento.

Exemplo

$B = \{ x \mid x \text{ é um inteiro e } 3x = 6\} \Rightarrow B = \{ 2 \}$

Conjunto Vazio é aquele que não possui elemento.

Exemplos

C = { x | x + 1 = x + 2 }
D = { x | x é par compreendido entre 3 e 3,56 }

Notação de Conjunto Vazio: { } ou $\varnothing$

1.4 Conjuntos Finitos e Infinitos

Intuitivamente um conjunto é finito se contém um número específico (contável) de elementos. Caso contrário, é infinito.

Exemplos

1) Conjunto Finito

E = Múltiplos de 5 entre 10 e 25 $\Rightarrow$ E = {15 , 20 }
G = Vogais de nosso alfabeto $\Rightarrow$ G = {a, e, i, o, u}

2) Conjunto Infinito

H = Múltiplos de 5 $\Rightarrow$ H = {5, 10, 15, 20, ... }
L = Números Impares $\Rightarrow$ L = { 1, 3, 5, 7,... }

1.5 Subconjuntos

Um Conjunto C qualquer é subconjunto de J **se e somente se** todo elemento de C for também elemento de J.

Exemplos

Seja A = { a, b, c, d } ; B = { a, c } e C = { a, c, d }. É fácil verificar que B é subconjunto de C e C é subconjunto de A.

Notação:

$B \subset A \Rightarrow$ B é subconjunto de A ou B está contido em A.
$C \not\subset B \Rightarrow$ C não é subconjunto de B ou C não está contido em B.
$B \not\supset A \Rightarrow$ B não contém A.
$A \supset B \Rightarrow$ A contém B.

Obs: Os símbolos $\subset$, $\not\subset$, $\not\supset$ e $\supset$ relacionam **Conjunto** com **Conjunto.**

Observações Importantes:

$\varnothing \subset$ A, qualquer que seja A.
$A \subset$ A, qualquer que seja A.

1.6 Igualdade de Conjuntos

Diz-se que os Conjuntos A e B são iguais quando ambos têm os mesmos elementos, isto é, qualquer elemento de A é elemento de B e qualquer elemento de B é também elemento de A.

Exemplos

Se A = {a, b, c, d} e B = { a, c, d, b } então A=B $\rightarrow$ a ordem dos elementos não importa para a igualdade.

Se X = {a, b, c} e Y = {a, a, b, c, c} então X=Y $\rightarrow$ a repetição de elementos não importa para a igualdade.

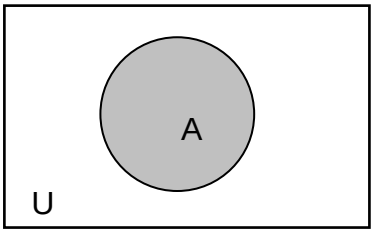
Importante

SE $A = B \Leftrightarrow A \supset B$ e $B \supset A$

1.7 Conjunto Universo

No estudo e nas aplicações da Teoria dos Conjuntos, admite-se que todos os conjuntos devem ser pensados como subconjuntos de um conjunto previamente fixado e ao qual denomina-se CONJUNTO UNIVERSO que é representado por U ou Ω.

Diagrama de Euler -Venn



É possível pensar nos elementos do Conjunto Universo representados pelos pontos interiores de um retângulo e os elementos de qualquer de seus subconjuntos pelos pontos interiores de um círculo.

1.8 Conjuntos Numéricos Importantes

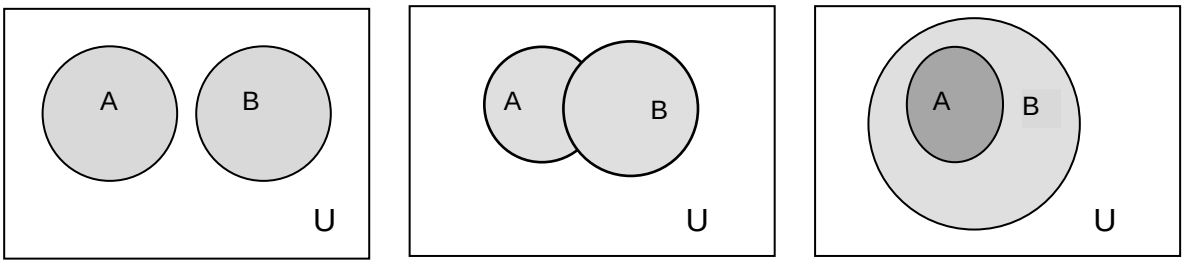
Alguns conjuntos numéricos são muito importantes e de enorme utilização no estudo da Matemática. Os principais deles são:

a) **N** = Conjunto dos números naturais = { 0, 1, 2, 3, 4, 5,.... }
b) $\mathbf{N}^*$ = Conjunto dos Naturais exclusive o zero = { 1, 2, 3,...}
c) **Z** = Conjunto dos números inteiros = { ...,-2, -1, 0, 1, 2,... }
d) $\mathbf{Z}_+$ = Conjunto dos Inteiros não Negativos = { 0, 1, 2, 3,...}
e) $\mathbf{Z}_-$ = Conjunto dos Inteiros não Positivos = {...,-2, -1, 0 }
f) $\mathbf{Z}_+^*$ = Conjunto dos números Inteiros Positivos = {1, 2, 3,..}
g) $\mathbf{Z}_-^*$ = Conjunto dos números Inteiros Negativos = {...,-2, -1}
h) $\mathbf{Z}^*$ = Conjunto dos inteiros não zero = {...,-2, -1, 1, 2,...}
i) **Q** = Conjunto dos números Racionais = $\{ a/b \mid a \in Z \quad e \quad b \in Z^* \}$
j) **Q'** = Conjunto dos números Irracionais.
k) **R** = Conjunto dos números Reais = **Q** U **Q'**

1.9 Operações com Conjuntos

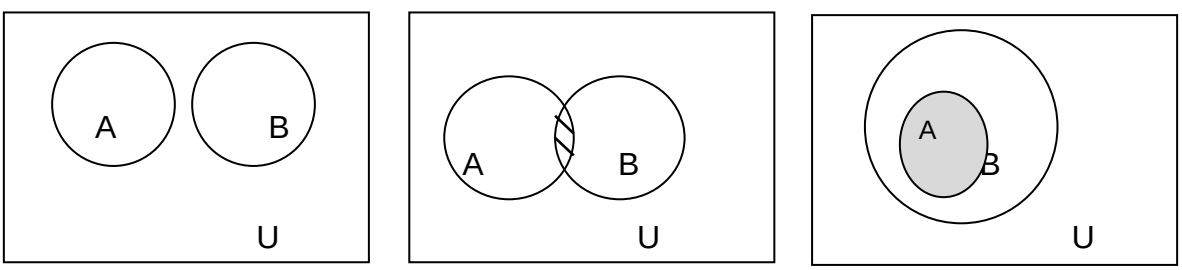
1.9.1 União de Conjuntos (∪).

$A \cup B = \{ x \in U \mid x \in A \textbf{ ou } x \in B \}$



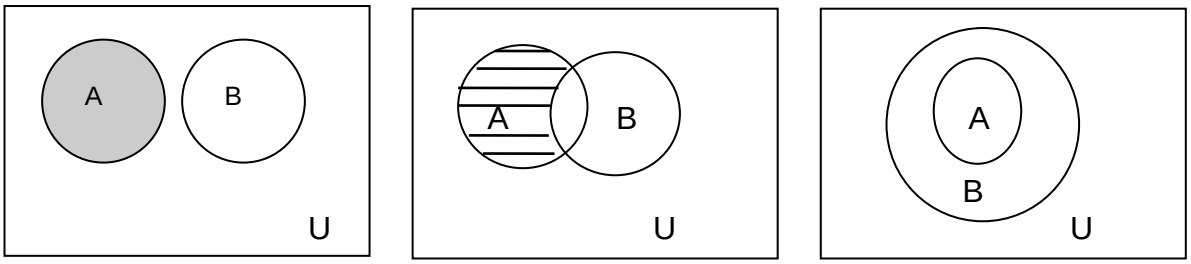
1.9.2 Interseção de Conjuntos (∩).

$A \cap B = \{ x \in U \mid x \in A \textbf{ e } x \in B \}$



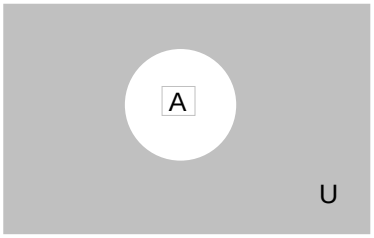
1.9.3 Diferença de Conjuntos (–).

$$A - B = \{ x \in U \mid x \in A \text{ e } x \notin B \}$$

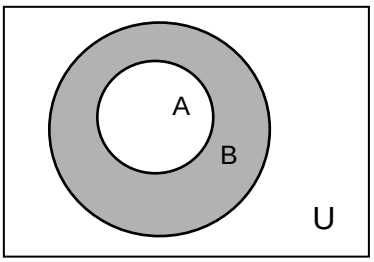


1.9.4 Complementar de A em Relação ao Universo

$$C_U^A = A' = \{ x \in U \mid x \notin A \}$$



1.9.5 Complementar de A em relação a B

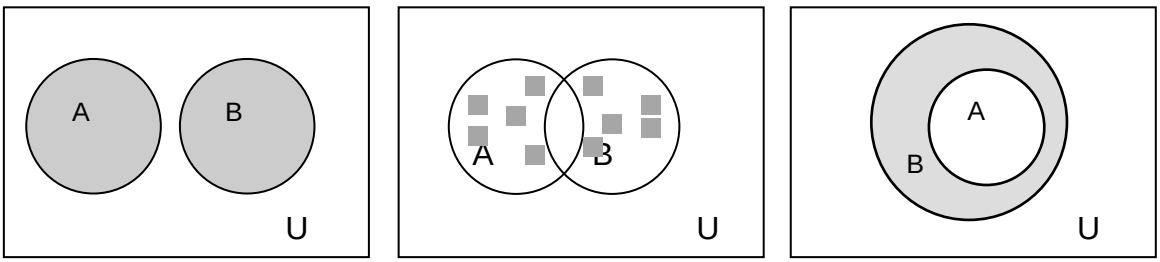


Sejam A e B dois subconjuntos do Universo U tais que $A \subset B$. A diferença B - A é denominada "Complementar de A em relação a B" e é indicada por C_B^A.

$$C_B^A = B - A = \{x \in U \mid x \in B \text{ e } x \notin A\}$$

1.9.6 Diferença Simétrica

(Δ) : $A \Delta B = (A - B) \cup (B - A)$



1.10 Propriedades das Operações com Conjuntos

a) $A \cup A = A$
b) $A \cup \varnothing = A$
h) $A \cap A = A$
i) $A \cap \varnothing = \varnothing$

c) $A \subset B \Leftrightarrow A \cup B = B$
d) $A - A = \varnothing$
e) $\varnothing - A = \varnothing$
f) $\varnothing' = U$
g) $A' \cup A = U$
j) $A \subset B \Leftrightarrow A \cap B = A$
k) $A - \varnothing = A$
l) $(A')' = A$
m) $U' = \varnothing$
n) $A \cap A' = \varnothing$

1.10.1 Comutativas

$$A \cup B = B \cup A$$
$$A \cap B = B \cap A$$

1.10.2 Associativas

$$(A \cup B) \cup C = A \cup (B \cup C)$$
$$(A \cap B) \cap C = A \cap (B \cap C)$$

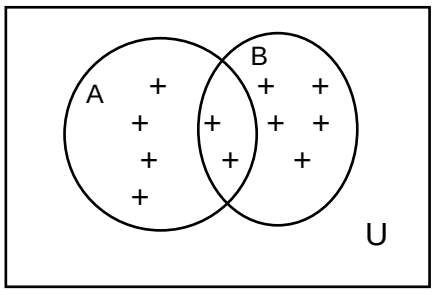
1.10.3 Distributivas

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$
$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

1.10.4 Leis de "DE MORGAN"

$$(A \cup B)' = A' \cap B'$$
$$(A \cap B)' = A' \cup B'$$

1.11 Número de Elementos de um Conjunto



$n (A \cup B) = n (A) + n (B) – n (A \cap B)$

$n (A - B) = n (A) – n (A \cap B)$

$n (A') = n (U) – n (A)$

Observe que: n (A) = 6; n (B) = 7, mas $n (A \cup B) = 11$
Diferente, portanto de n (A) + n (B) = 13

Se fosse a união de 3 conjuntos seria:
$n (A \cup B \cup C) = n (A) + n (B) + n (C) – n (A \cap B) – n (A \cap C) – n (B \cap C) + n (A \cap B \cap C)$

1.12 Conjunto das partes de um conjunto A

É o conjunto cujos elementos são todos os subconjuntos de A.

Notação: $P(A) = \{ x \mid x \in A \}$
Exemplo

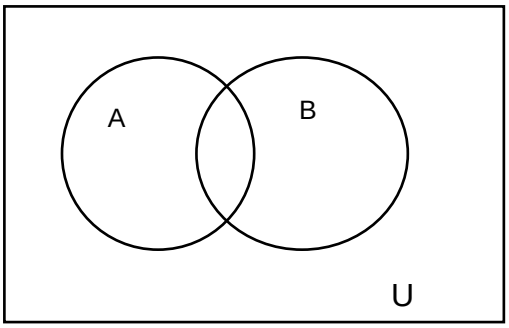
$A = \{a , b\}$ $P(A) = \{ \{a\} , \{b\} , \{a,b\} , \varnothing\}$

Se A têm n elementos então P(A) terá 2^n elementos

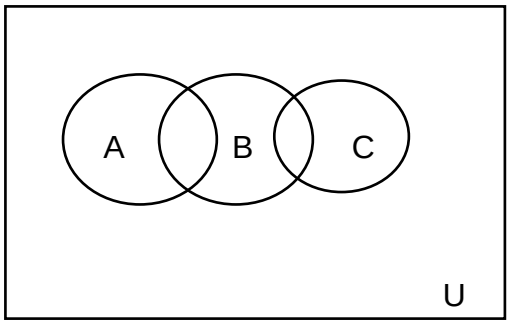
1.13 Exercícios de Conjuntos

1) Assinale no Diagrama o conjunto indicado:

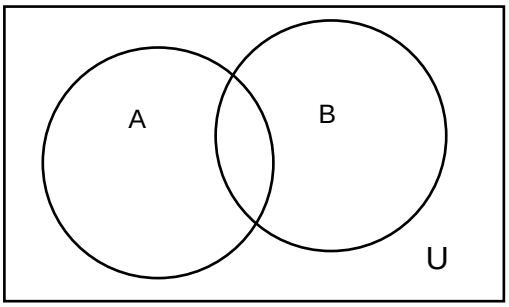
a) $(A \cup B)'$



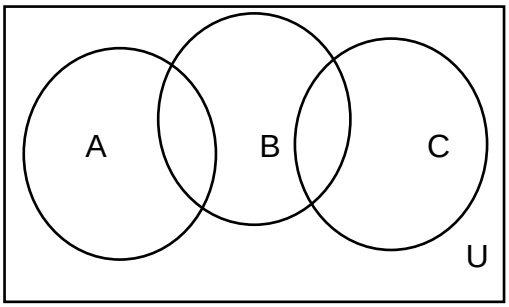
c) $(A \cup B) \Delta C$



b) $(A - B)$



d) $(A \cap B) \cup C$



2) Dados os Conjuntos A = { 0, 1, 2 } e B = { 1, 2, 3, 4 } , determine o Conjunto X tal que $A \cap X=\{0, 1\}$, $B \cap X=\{1, 4\}$ e $A \cup B \cup X =\{0, 1, 2, 3, 4, 5, 6\}$.

3) Dados os conjuntos A = {1,2,3,4,5} ; B = {2,4,5,6,7} e C = {0,2,3,4,5}
Determinar:

a) $(A \Delta B)$

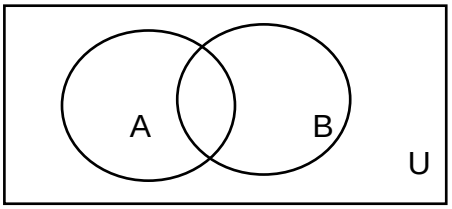
b) $(A \cap B) \cup (A \cap C)$

c) (C - B)

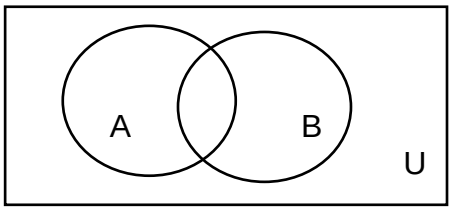
d) $(A \Delta C) \Delta (B \Delta C)$

4) Assinalar no diagrama o Conjunto indicado:

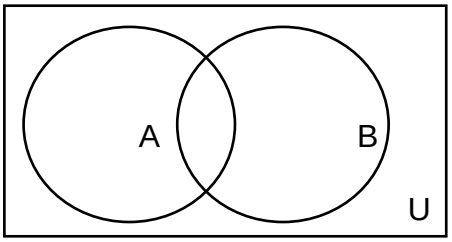
a) **$B \cup A'$**



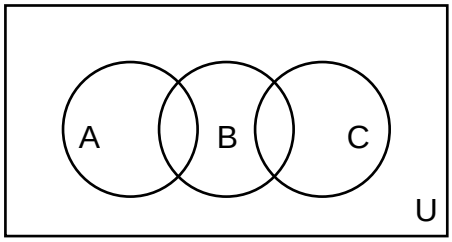
b) **$(B \cup A)'$**



c) **$(A - B) \cap B'$**



d) **$(A - B)' \Delta (B \cup C)'$**



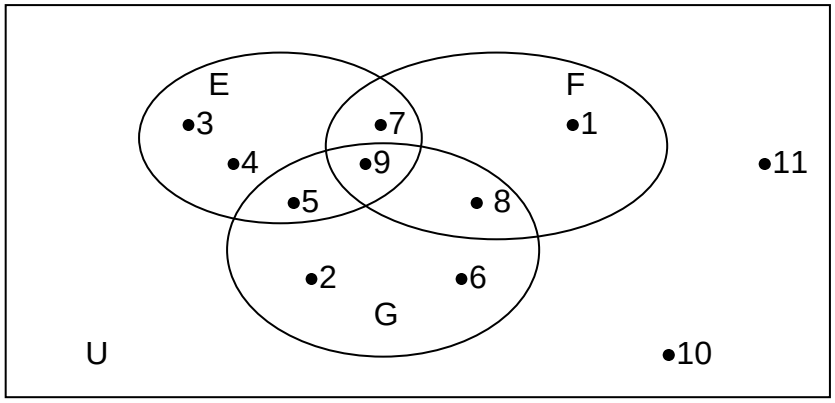
5) Uma pesquisa foi realizada em uma empresa com 500 funcionários, todos ouvidos. Os resultados obtidos foram os seguintes: a) 120 funcionários lêem o jornal A; b) 98 funcionários lêem o jornal B; e c) 15 funcionários lêem os dois jornais. Nessas condições, pergunta-se:

a) Quantas pessoas lêem apenas o jornal A?
b) Quantas pessoas lêem apenas o jornal B?
c) Quantas pessoas lêem apenas um dos jornais?
d) Quantas pessoas não lêem nenhum dos jornais?

6) Numa sala de 630 alunos, 350 deles estudam Português, 210 estudam espanhol e 90 estudam as duas matérias (português e espanhol). Pergunta-se:

a) Quantos alunos estudam apenas português?
b) Quantos alunos estudam apenas espanhol?
c) Quantos alunos estudam português ou espanhol?
d) Quantos alunos não estudam nenhuma das duas matérias?

7) Considere o diagrama e escreva os seguintes conjuntos:



a) E, F e G
b) E ∪ F
c) F – G
d) (E ∩ G) – F
e) (E ∩ G)'
f) (E ∪ G)' – (F ∪ E)
g) (F – G)' Δ (E ∪ G)'

8) Dado o conjunto $A = \{1, 2, \{3\}\}$, diga se as proposições a seguir são verdadeiras ou falsa:

( ) 0 ∈ A
( ) 1 ∈ A
( ) 1 ⊂ A
( ) ∅ ∈ A
( ) {0, 1} ∈ A
( ) {0, 1} ⊂ A
( ) ∅ ⊂ A
( ) 3 ∈ A
( ) {{3}}⊂ A
( ) {3}⊂ A
( ) {3} ∈ A
( ) {{3}}∈ A

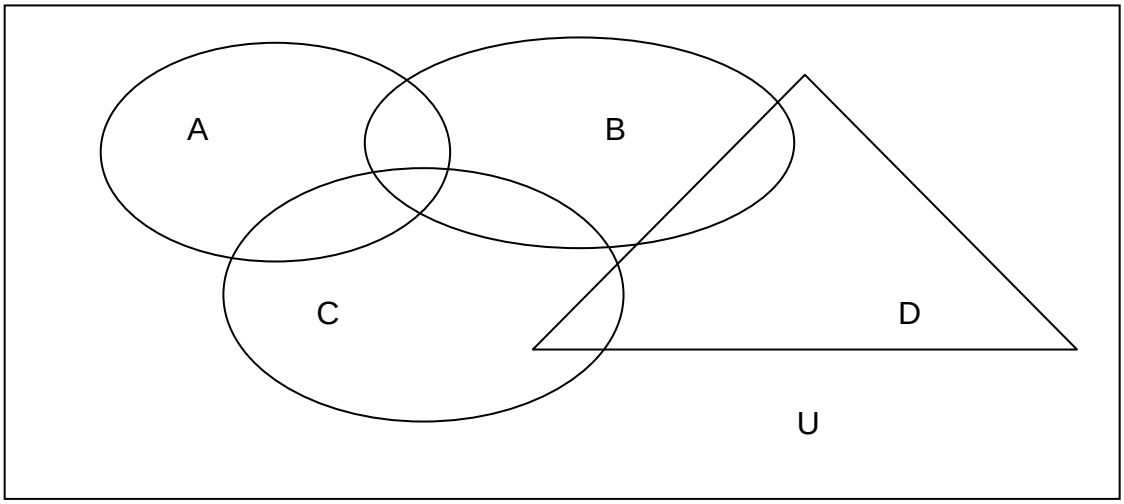
9) Numa prova constituída de dois problemas, 300 alunos acertaram somente um dos problemas, 260 acertaram o segundo, 100 alunos acertaram os dois e 210 erraram o primeiro. Quantos alunos fizeram a prova?

10) Marque no diagrama abaixo os seguintes conjuntos:

$a)\quad ((A \cup B) - C)^{|}$ $\qquad b)\quad (A - B)\Delta(C - D)^{|}$

$c)\quad (A \cap B) - (C \cup D)^{|}$ $\qquad d)\quad (C \cup B) \cap (B - D^{|})$

$e)\quad (A \cap B) \cup (C\ \Delta\ D)$ $\qquad f)\quad ((A \cap C) - D)^{|}$



11) Numa pesquisa com jovens, foram feitas as seguintes perguntas para que respondessem sim ou não: Gosta de música? Gosta de esportes? Responderam sim à primeira pergunta 90 jovens; 70 responderam sim à segunda; 25 responderam sim a ambas; e 40 não a ambas. Quantos jovens foram entrevistados?

2. Lógica

2.1 O que é Lógica

É essencialmente o estudo da natureza do raciocínio e as formas de incrementar sua utilização. Poderíamos dizer que é a arte ou a técnica que nos ensina a usar corretamente o pensamento, isto é, ordenar nossa razão; é o conhecimento das formas gerais e regras do pensamento correto e verdadeiro, independentemente dos conteúdos pensados; regras para a demonstração científica verdadeira; regras para os pensamentos não-científicos; regras sobre o modo de expor o conhecimento; regras para verificação da verdade ou falsidade de um pensamento.
Exemplo

1. Todo Homem é mamífero.Todo mamífero é animal. Portanto, todo homem é animal.

2. Todo homem é mortal. Sócrates é um homem. Portanto Sócrates é mortal

2.2 Estudo da Lógica Matemática

2.2.1 Noções de Lógica Matemática

A Lógica Matemática é uma ciência exata que só admite margem de erro zero.
Exemplos

Sequência de Fibonacci: 0, 1, 1, 2, 3, 5, ...
Série das potências de 2: 1, 2, 4, 16, 256, 512, 1024, ..

2.2.2 Proposições

É todo conjunto de palavras ou símbolos que exprimem de forma completa um pensamento de sentido complexo.

Exemplos

- A Lua é satélite da Terra.
- $\pi > \sqrt{5}$
- 1 = 2

A Lógica Matemática é baseada em dois princípios básicos ou axiomas:

i) **Princípio da Não-Contradição:** Uma proposição não pode ser verdadeira ou falsa ao mesmo tempo.

ii) **Princípio do Terceiro Excluído:** Uma proposição só pode ser verdadeira ou falsa. Por este motivo a Lógica Matemática também é denominada lógica bivalente.
Exemplos:

- Vasco da Gama descobriu o Brasil.
- π é número racional.

2.2.2.1 Valores Lógicos das Proposições

O valor lógico de uma proposição é verdade se a proposição é verdadeira e falsidade se a proposição é falsa.

2.2.2.2 Proposições Simples x Proposições Compostas

Uma proposição é dita **simples** quando não têm nenhuma outra proposição como parte integrante de si mesma. Por esse motivo as proposições simples são também denominadas proposições atômicas. As proposições simples são em geral representadas por letras minúsculas.

Exemplos

- p: Carlos é careca.
- q: Pedro é estudante.

Uma proposição é dita **composta** quando é formada por duas ou mais proposições simples. Por esse motivo as proposições compostas são também denominadas proposições moleculares. As proposições compostas são em geral representadas por letras maiúsculas.

Exemplos:

- P: Carlos é careca **e** Pedro é estudante.
- Q: Carlos é careca **ou** Pedro é estudante.

Note nos exemplos acima que as proposições compostas são formadas por duas ou mais proposições simples cada uma, que por sua vez são relacionadas através de conectivos. No caso da proposição composta P o conectivo utilizado é da conjunção (equivale ao operador lógico e), já na proposição composta Q o conectivo utilizado é o conectivo disjunção (equivalente ao operador lógico ou).

2.2.2.3 Conectivos

São palavras que se usam para formar novas proposições a partir de outras. São exemplos de conectivos: negação, conjunção, disjunção, disjunção exclusiva, condicional e bi-condicional.

A seguir serão estudadas as tabelas-verdade dos conectivos básicos da lógica matemática:

1 - Negação ($\sim$)

Dada a proposição simples p, a tabela-verdade da negação é dada por:

p	~p
V	**F**
F	**V**

$V(\sim p) = \sim V(p)$

O conectivo negação é equivalente a palavra **não** em língua portuguesa.

2 - Conjunção ($\wedge$)

Dada as proposições simples p e q, a tabela-verdade da conjunção é dada por:

p	q	$p \wedge q$
V	V	**V**
V	F	**F**
F	V	**F**
F	F	**F**

$V(p \wedge q) = V(p) \wedge V(q)$

O conectivo conjunção é equivalente a letra **"e"** em língua portuguesa.

3 - Disjunção ($\vee$)

Dada as proposições simples p e q, a tabela-verdade da disjunção é dada por:

p	q	$p \vee q$
V	V	**V**
V	F	**V**
F	V	**V**
F	F	**F**

$V(p \vee q) = V(p) \vee V(q)$

O conectivo disjunção é equivalente a palavra "**ou"** em língua portuguesa. Note que o sentido da palavra ou nesse tipo de disjunção é inclusivo, pois não exclui a possibilidade de uma ou outra proposição simples ocorrer. Por esse motivo essa disjunção pode também ser denominada disjunção inclusiva.

4 - Disjunção Exclusiva ($\underline{\vee}$)

Dada as proposições simples p e q, a tabela-verdade da disjunção exclusiva é dada por:

p	q	$p \underline{\vee} q$
V	V	**F**
V	F	**V**
F	V	**V**
F	F	**F**

$V(p \underline{\vee} q) = V(p) \underline{\vee} V(q)$

O conectivo disjunção é equivalente a palavra **ou** em língua portuguesa. Note que o sentido da palavra ou nesse tipo de disjunção é exclusivo.

5 - Condicional ($\rightarrow$)

Dada as proposições simples p e q, a tabela-verdade do condicional é dada por:

p	q	$p \rightarrow q$
V	V	**V**
V	F	**F**
F	V	**V**
F	F	**V**

$V(p \rightarrow q) = V(p) \rightarrow V(q)$

Note que a o fato da primeira proposição p ser verdadeira **não implica** que a segunda proposição q tenha que ser também verdadeira. Isso equivale a dizer que o fato da primeira proposição p ser verdadeira **é condição necessária mas não suficiente** para que a segunda proposição q seja verdadeira.

6 - Bi-condicional ($\leftrightarrow$)

Dada as proposições simples p e q, a tabela-verdade do bi-condicional é dada por:

p	q	$p \leftrightarrow q$
V	V	**V**
V	F	**F**
F	V	**F**
F	F	**V**

$V(p \leftrightarrow q) = V(p) \leftrightarrow V(q)$

Note que a o fato da primeira proposição p ser verdadeira **implica** que a segunda proposição q tenha que ser também verdadeira. Isso equivale a dizer que o fato da primeira

proposição p ser verdadeira **é condição necessária e suficiente** para que a segunda proposição q seja verdadeira.

2.2.3 Operações Lógicas Sobre Proposições

Denomina-se cálculo proposicional os resultados e conclusões que podem ser obtidos através da resolução da tabela-verdade de uma proposição composta. A construção da tabela verdade será assim feita:

São montadas *n* colunas, onde *n* é o número de proposições simples que compõem a proposição composta que se quer resolver, explorando todas as possibilidades possíveis. Em uma proposição composta de *n* proposições simples, existem 2^n combinações possíveis de resultados de proposições simples. A coluna *i* (onde $i \leq n$) é formada por $2^n / 2^i$ valores verdadeiros e falsos alternados.

Atenção:

- A prioridade entre os operadores lógicos é : 1º Não, 2º E , 3º OU
- Quando houver parênteses resolve-o primeiro
- Da esquerda para direita.

Exemplo

Construir a tabela-verdade da proposição composta P (p,q) = ~ (p $\wedge$ ~q)

p	q	~q	p $\wedge$ ~q	~ (p $\wedge$ ~q)
V	V	F	F	**V**
V	F	V	V	**F**
F	V	F	F	**V**
F	F	V	F	**V**

2.3 Exercícios e Teste de Raciocínio Lógico

I - Nos exercícios de 1 até 10 marque qual a sequência que vem logo após.

1- 1322543210, 1344543210, 1366543210, ...

A) 1388543211 B) 1366543211 C)1388543210 D) 1300543210

2- 1234567890, 6789012345, 1234567890, ...

A) 6789012345 B) 1234567890 C) 2345678901 D) 7890123456

3- 1234678125, 1236784125, 1237846125, ...

A) 1234678126 B) 1234678125 C) 1238467125 D) 1234678251

4- 2938204185, 3938204184, 4938204183, ...

A) 1938204182 B) 5938204182 C) 5938204183 D) 5938204186

5- 5494859485, 5496959406, 5498059427, ...

A) 5494859448 B) 5490159427 C) 5490159448 D) 5494859485

6- 4356787874, 4361787879, 4366787884, ...

A) 4361787889 B) 4351787889 C) 4377787889 D) 4371787889

7- 4135646789, 2115444769, 6155848709, ...

A) 4135646789 B) 8175040729 C) 0195242749 D) 2115444769

8- 4564310041, 4575533341, 4586756641, ...

A) 4597867741 B) 4597979941 C) 4597978841 D) 4597869941

9- 4594594854, 9594594859, 9894594889, ...

A) 4894594854 B) 4894594884 C) 9874594789 D) 9594594859

10- 3429857414, 3021827314, 3122837415, ...

A) 3220817315 B) 3323847315 C) 3428807315 D) 3223847516

Respostas:
1=C 2=A 3=C 4=B 5=C 6=D 7=A 8=B 9=B 10=D

II – Resolva as seguintes proposições lógicas

A	B	C	$A \vee \bar{B}$	$(A \vee B) \wedge C$	$(A \vee \bar{B}) \wedge \bar{C}$	$(A \vee \bar{B}) \wedge (\bar{B} \vee \bar{C})$	$A \wedge \bar{B}$
F	F	F					
F	F	V					
F	V	F					
F	V	V					
V	F	F					
V	F	V					
V	V	F					
V	V	V					

3. Algoritmos

3.1 Definição

É uma sequência finita de ações que descrevem como um problema deve ser resolvido; É uma sequência ordenada, sem ambiguidade de passos que levam à solução de um dado problema; É a descrição de um padrão de comportamento, expressado em termos de um repertório bem definido e finito de ações "primitivas", das quais damos por certo que elas podem ser executadas.

Exemplos de Algoritmos do nosso dia a dia:

Receitas culinárias

Instruções de Montagem de um brinquedo.

3.2 Algoritmos Estruturados

Quando queremos resolver um problema utilizando um computador, a sequência de passos que conduzem à solução é detalhada até que se chegue a um conjunto de ações primitivas que chamamos de comandos ou instruções, que podem ser entendidas e executadas pela máquina. Este algoritmo no maior nível de detalhamento possível, quando "traduzido" para uma linguagem de programação, constituirá um programa. Uma das vantagens de se programar utilizando um algoritmo é que a partir dele o programador pode implementá-lo em qualquer linguagem de programação que conheça.

A programação estruturada ou o desenvolvimento de algoritmos estruturados consiste na utilização de técnicas que permitem a sistematização do desenvolvimento, que simplificam bastante a solução de problemas grandes e complexos, mas que podem também ser utilizados para algoritmos mais simples. Objetivos dessas técnicas são:

- Facilitar o desenvolvimento e representação de algoritmos;
- Facilitar a leitura e o entendimento dos algoritmos pelas pessoas;
- Antecipar a comprovação de sua correção;
- Facilitar a manutenção (correções e modificações) dos programas e algoritmos;
- Permitir que o desenvolvimento de algoritmos pudesse ser empreendido simultaneamente por uma equipe de pessoas.

Para isso os algoritmos estruturados são desenvolvidos levando-se em conta três premissas básicas.

1) Desenvolver algoritmos do geral para o particular, por refinamentos sucessivos (desenvolvimento "top-down").
2) Decompor os algoritmos em módulos funcionais, organizados de preferência em um sistema hierárquico. Esses módulos trazem vantagens adicionais: para o desenvolvimento, pois cada módulo poderá ser desenvolvido por um programador/analista diferente; para teste, pois testa-se um módulo de cada vez independentemente.
3) Dentro de cada módulo utilizar alguns poucos e bem definidos comandos e estruturas de controle.

Exemplo: Uma consulta num caixa Eletrônico.

```
Inicio
        Execute Apresentação
        Execute Pesquisa
        Execute Finalização
Fim
```

Apresentação
Inicio
    Execute Tela Inicial
    Execute Senha
    Execute Menu
Fim

Pesquisa
Inicio
    Execute Operação desejada
    Execute Outro Serviço
Fim

Finalização
Inicio
    Execute Impressão/Dinheiro
Fim

3.3 Programas

É um conjunto de instruções ordenadas logicamente escritos em uma linguagem de programação a serem interpretadas e executadas pelo computador. Essas instruções são representadas por um conjunto de símbolos armazenados na memória do computador.

3.4 Técnicas Usadas para Desenvolver os Algoritmos:

Os algoritmos podem ser representados de diversas formas gráficas, a saber:

- Fluxogramas (Hoje em dia em desuso)
- Diagramas de Nassi-Shneiderman
- Pseudo-Código: Aqui não tem os inconvenientes da ambiguidade de uma língua e nem os rigores da sintaxe de uma linguagem de programação, e, por isso, é também bastante recomendada.

3.5 Tipos de Dados

Tudo em informática é baseado na manipulação de Informações, que podem ser classificadas em vários tipos de dados:

3.5.1 Dados Numéricos

Os dados numéricos são representados pelos números Inteiros e Reais:
Inteiro: qualquer número inteiro positivo ou negativo: (...,-1, 0, -1....).
Real: qualquer número real: (-5, 30.9, 0, 23, -345.62).

3.5.2 Dados Literais

É constituído por uma sequência de caracteres contendo letras, dígitos e/ou símbolos especiais. Esses dados são conhecidos como String, alfanuméricos ou cadeia de caracteres. São representados entre aspas (“”).
Exemplos

“Engenharia de Produção”	Literal de comprimento 22
“ Página”	Literal de comprimento 8
“ ”	Literal de comprimento 1
“123”	Literal de comprimento 3

3.5.3 Dados Lógicos

Esses dados são muitas vezes chamados de Booleanos, Só poderá assumir dois valores: V (Verdadeiro) e F (Falso).

Exercícios

Marque nos parênteses com as respectivas letras: Se for **I** tipo de dado Inteiro, **R** tipo de dado Real, **L** tipos de dados literais e **B** tipos de dados Booleanos e **X** com os tipos de dados que não podemos classificar.

( )	123	( )	V	( )	reti”	( )	Casa
( )	“.F.”	( )	“ m12”	( )	.V.	( )	.F
( )	“Mar”	( )	-0.0	( )	“.....”	( )	“.F>>>”
( )	“-0.0”	( )	-1.000	( )	-12	( )	23.98

3.6 Constantes

É um identificador que armazena um valor fixo e imutável, durante toda a execução do algoritmo. As constantes dos tipos caracteres serão colocadas entre aspas (“”).

Ex: PI = 3,14
NOME = “Maria Catarina”

3.7 Variáveis

Como o próprio nome diz essa informação poderá ser alterada em um dado instante. Exemplo: Índice da Bolsa de Valores, Cotação do Ouro etc.

3.8 Identificadores

É um conjunto de caracteres (letras, números ou símbolos). Devem-se observar alguns pontos na construção desses identificadores:
1º Devem começar por caracteres alfabéticos;
2º Podem ser seguidos por mais caracteres alfabéticos e/ou numéricos;
3º Não é permitido o uso de caracteres especiais;
4º Os caracteres alfabéticos deverão ser escritos em maiúsculos ou minúsculos, mas deve ser mantida uma padronização.

Ex: SOMA - identifica a soma de determinados números
MEDIA - identifica a média de um conjunto de números (idades, pesos, etc.).

Exercícios
Marque com V os identificadores válidos e com F os identificadores inválidos.

( )	OMEGA	( )	3D/P
( )	VB67	( )	3P
( )	R]F	( )	MEMO
( )	R/F	( )	DIOGO@
( )	E908R	( )	FAVO

3.9 Declaração de Variáveis

Sempre no início do algoritmo deverão ser declaradas todas as variáveis que forem ser utilizadas no algoritmo para que quando codificarmos em uma linguagem, o compilador possa reservar um espaço na memória para as mesmas. Será feita da seguinte forma:

VAR

<nome da variável> : <tipo da variável>

Onde o nome da variável poderá ser uma lista de variáveis caso elas sejam do mesmo tipo.

Atenção

1- A palavra VAR deverá ser utilizada sempre e uma única vez.
2- Quando for definida uma ou mais variáveis do mesmo tipo serão separadas por vírgulas.
3- Tipos diferentes de variáveis serão declaradas em linhas diferentes.

```
Exemplo   VAR
              NOME          : Literal[30]
              IDADE         : Inteiro
              PROFISSAO     : Literal[15]
              SALARIO       : Real
```

3.10 Operadores

São elementos funcionais que atuam sobre determinados operandos produzindo determinados resultados. Eles classificam-se em:

3.10.1 Operadores Aritméticos

É um conjunto de símbolos que representam as operações matemáticas. São eles:

+ soma
- subtração
* multiplicação
/ divisão
** exponenciação (Em pascal : $2^3 \Rightarrow$ Exp(3*Ln(2)))

OBS: Também usaremos os operadores: MOD e DIV
m mod i (resto): Fornece o resto da divisão inteira de m por i.
m div n (quociente): Fornece o quociente inteiro da divisão de m por n.
Ex: 2 + 2 = 4
3 ** 2 = 9
15 mod 6 = 3
15 div 6 = 2
Exemplo Mod/Div

```
Program Exemplo_Mod_Div;
Var
        num:integer;
        Resto,Divi:Integer;
 Begin
        Write('Digite um número : ');
        Readln(Num);
        Resto:=Num mod 3;
        Divi:=Num div 3;
        Writeln(Resto,'    ', Divi);
        Readkey;
 End.
```

Saída:

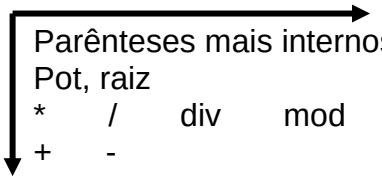
```
Digite um número : 7          Digite um número : 2          Digite um número : -7
1       2                     2       0                     -1      -2

Digite um número : -2         Digite um número : -5
-2      0                     -2      -1
```

Obs: Imagine que a divisão fosse por -3 isto é: resto:=Num mod -3 e Divi:=Num div -3
Saída:

```
Digite um número : 7          Digite um número : 2          Digite um número : -2
1       -2                    2       0                     -2      0
Digite um número : -14        Digite um número : -7         Digite um número : -3
-2      4                     -1      2                     0       1
```

Prioridades



3.10.2 Operadores Relacionais

São operadores usados para fazer a comparação entre variáveis de mesma natureza, ou seja, do mesmo tipo de dado. O resultado sempre será um valor lógico Booleano (V ou F). São eles:

Operador	Função
=	Igualdade
< >	Diferente
>	Maior
>=	Maior igual
<	Menor
<=	Menor igual

Exemplo

Seja A=45 e B= 35

A>=B ⟶ Verdadeiro
A> B ⟶ Verdadeiro
B > A ⟶ Falso
B < > A ⟶ Verdadeiro

3.10.3 Operadores Lógicos

São operadores usados em expressões lógicas cujo resultado da avaliação será V (verdadeiro) ou F (falso). São eles: AND, OR XOR e NOT .

3.11 Expressões Matemáticas

3.11.1 Expressões Aritméticas

São expressões cujo resultado é um valor inteiro ou real.

Exemplo

Sejam A, B valores Inteiros e C, D valores Reais, então:
A + B = terá como resultado um valor Inteiro
A + B * D = terá como resultado um valor Real
A / C = terá como resultado um valor Real.

Atenção: A prioridade na execução será:
1º Parênteses mais internos
2º Funções matemáticas
3º Prioridade da esquerda para a direita
4º Inversão do sinal (- para +)
5º Exponenciação (**)
6º Multiplicação e Divisão (* e /)
7º Adição e Subtração (+ e -)

3.11.2 Expressões Lógicas e Relacionais

São expressões cujos operadores são relacionais e/ou lógicos e cujos operandos são relações e/ou variáveis e/ou constantes do tipo lógico.

Atenção: A prioridade na execução será:

1º Parênteses mais internos
2º Funções matemáticas
3º Operadores aritméticos
4º Operadores relacionais
5º Operadores lógicos

3.11.3 Expressões Literais

São aquelas cujo resultado da avaliação é um valor literal.

Exemplo

A concatenação de Dados Literais (Strings).

Se, A = “Casa” B = “ ” C = “Velha” então:

A + B + C = “Casa Velha”

3.12 Funções Matemáticas

As funções matemáticas quando usadas nos algoritmos podem ser representadas como:

Log(x) : logaritmo de um numero na base 10.
Raiz(x) : Calcula a raiz quadrada de um número.
Int (x) : a parte inteira de um número fracionário.
Frac(x) : a parte fracionária de x.
Abs (x) : Valor absoluto de x.

Exercícios:

1 - Marque V ou F se os identificadores são ou não válidos:

( )	$ret	( )	UH!	( )	JOSÉ
( )	B567	( )	RET1	( )	TEMA&1
( )	P[oi]	( )	12ROP	( )	TEMA2
( )	O2O	( )	XPTO	( )	AL PO

2 - Seja A = 6 , B = 2 C = 1,6 , D = “Meta” , E = “Info” coloque V ou F nas expressões abaixo.

( )	A >= C+B	( )	A**B >= B**A	( )	(B *2 <> 0) E (A<>B)
( )	D+E = “MetaInfo”	( )	C+ B <= B**5	( )	Não(C+5 > B**4)
( )	C <> B*4,5	( )	(B div 3 <> 0) OU (A<>C)	( )	Raiz(B**4) <> 4
( )	Raiz(A**2) = B	( )	Não(A<B*4)	( )	Não(A>B**4)
( )	A mod B = 3	( )	E+D = “Infometa”	( )	C+ B = B**2
( )	A div B = 3	( )	Raiz(B**4) = 4	( )	abs (A/B) = 3
( )	(A*B) div 3 = 4	( )	A mod 8 = 0	( )	4 + B <> 6
( )	frac (C) = 0,6	( )	(B div 2 <> 0) OU (A<>B)	( )	E+D = “InfoMeta ”
( )	Int (C) = 1	( )	abs (A/B) = -3	( )	C*6 >= C + 67
( )	E+D = “ InfoMeta”	( )	(abs (A/B) = -3) E (A+2= 8)	( )	A/2 <= B/2

3.13 Comandos de atribuição

Serve para atribuir um valor a uma variável, usaremos o símbolo ← ou **=**

3.14 Comandos de Entrada e Saída de Dados

Entrada: Ler (variável) ; Ler(arq, registroalu) ⇒ quando for lido de um arquivo.

Saída: Escreva (variável), Escreva (Registroalu, Nome, Idade) ⇒ quando for impresso o conteúdo de um registro.

Exemplo:

```
Inicio
        Ler(NOME)
        Ler(IDADE)
        Escreva(“ O Nome é = ”, NOME)
        Escreva(“A Idade é = ”, IDADE)
Fim
```

3.15 Documentação

A documentação é uma etapa muito importante e deve ser feita desde o inicio até o final do desenvolvimento do projeto. Devemos ter em mente alguns princípios básicos:

1º Ao criar constantes e variáveis devemos utilizar nomes significativos e comentá-las

2º Usar a indentação, para mostrar a estrutura e sequência dos comandos.

3º Documentar as rotinas e fazer um dicionário de dados que especifique o conteúdo de cada variável ou constante.

Exemplo

Será colocado entre chaves({ }) ou da seguinte forma :(*.....*)

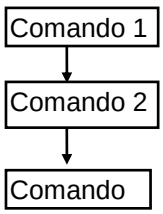
```
(* somatot  é a soma das idades dos alunos *}
{ num - representa o código de cada produto }
```

3.16 Estruturas Básicas de Controles

3.16.1 Sequência

É um grupo de comandos executados um após o outro.

Fluxograma

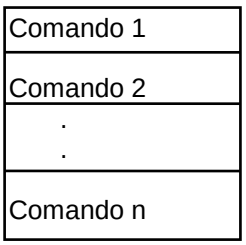


Pseudo-Código

```
Inicio
        comando 1
        comando 2

        comando n
Fim
```

NS



Exemplo

```
Inicio
        x = 1
        y = 127
        d = x *  y
        Escreva ( x, y, d )
Fim
```

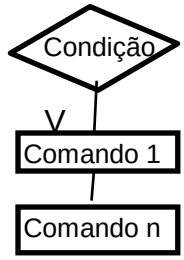
3.16.2 Seleção

Também chamada de estrutura de decisão ou de processamento condicional. A estrutura de seleção é utilizada quando a execução de um comando ou uma sequência de comandos depende de um teste anterior (uma ou mais comparações). A seleção pode ser simples ou composta.

3.16.2.1 Seleção Simples

Quando a execução de um ou vários comandos depender de uma condição verdadeira, e não há comandos a executar se a condução for falsa.

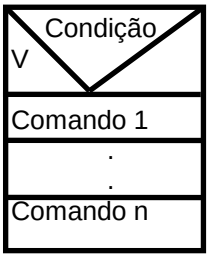
Fluxograma



Pseudo Código

```
Inicio
    SE  Condição      Então
        Comando 1
        Comando 2
        :
    Fim-SE
Fim
```

NS



Exemplo

Faça um algoritmo que leia dois números inteiros e escreva ambos os números, se o 1º for divisível pelo 2º:

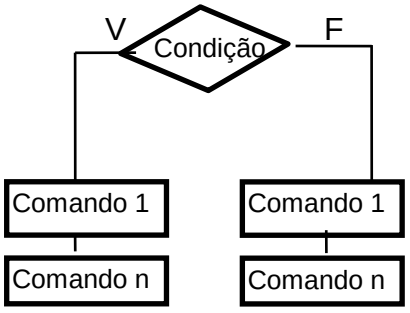
Programa Exemplo

```
var
    Num1, Num2: Inteiro
Inicio
    Leia (Num1)
    Leia (Num2)
    SE  Num1 mod Num2 = 0  Então
        Escreva (Num1)
        Escreva (Num2)
    Fim-SE
Fim
```

3.16.2.2 Seleção Composta

Se a condição for Verdadeira executa um ou mais comandos e se for falso também executará um ou mais comandos.

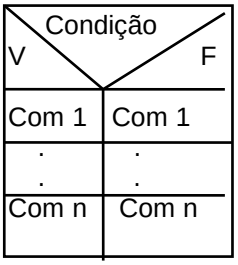
Fluxograma



Pseudo-Código

```
Inicio
    SE  Condição  Então
        Comando 1
        Comando 2
    Senão
        Comando 1
        Comando 2
    Fim-SE
Fim
```

NS



Exemplo

Faça um algoritmo que leia dois números inteiros e escreva o maior deles. Assumir que não podemos entrar com números iguais.

```
Programa exemplo
    Var
        Num1, Num2: Inteiro.
    Inicio
    |   Leia (Num1)
    |   Leia (Num2)
    |   SE  Num1  >  Num2  Então
    |   |       Escreva (“O maior nº é:”, Num1)
    |   Senão
    |   |       Escreva (“O maior nº é: ”, Num2)
    |   |       { Escreverá o Num2 se forem iguais}
    |   Fim-SE
    Fim
```

Exercícios

1 - Fazer um algoritmo que leia um nº inteiro e escreva-o se for diferente de zero.

2 - Fazer um algoritmo que leia dois números inteiros. Escreva a soma dos números se o 1º for maior que o 2º, escreva o produto se o 2º for maior que o 1º, e escreva um deles se os números forem iguais.

3 - Fazer um algoritmo que leia 3 números inteiros e escreva a soma deles.

4 - Fazer um algoritmo para calcular e escrever a área (S) de um triângulo de base b (real) e altura h(real). (S = b * h / 2).

5 - Fazer um algoritmo para calcular e escrever a área (S) de um quadrado de lado l(real), onde a área é dada por $S = l^2$.

6 - Fazer um algoritmo para calcular e escrever a raiz da equação do 1º grau, do tipo: A * X + B = 0.

7 - Fazer um algoritmo que leia dois valores inteiros e escreva qual dos dois é o maior deles. Se os números forem iguais deve ser dada uma mensagem: ”Iguais”

8 - Fazer um algoritmo que leia três valores inteiros e escreva o maior deles. Considere que não poderá ter números iguais. Deverá ser usado o encadeamento de “SE”

9 - Fazer um algoritmo que determine se o número inteiro lido é divisível por 23.

10 - Fazer um algoritmo que leia um número inteiro e escreva-o se ele for par ou se ele for ímpar.

11 - Fazer um algoritmo que leia um nº inteiro e escreva se ele é par e maior do que 50 ou se ele é ímpar menor que 39.

12 - Fazer um algoritmo que leia A, B, e C os três coeficientes de uma equação do 2º grau. Pede-se: Calcule o determinante e escreva se a equação possui 2 raízes reais e iguais, ou 2 raízes reais e diferentes ou não existe raízes. Logo após deverá ser calculada e exibida as raízes da equação quando existir.

13 - Fazer um algoritmo que leia 3 números quaisquer. Pede-se Calcular o produto e a Soma destes números e imprimir qual dos dois resultados é maior. Se a Soma for igual ao produto deverá ser impressa a mensagem: “Soma e Produto são iguais”.

14 – Fazer um algoritmo que leia o nome e três notas de um determinado aluno e ao final imprimir se ele está aprovado ou reprovado. O critério para aprovação é média maior ou igual a 7,0.

15 – Fazer um algoritmo que leia o nome, o Salário bruto, e a quantidade de filhos de um determinado funcionário. Pede-se: Escrever o Nome, o Desconto do INSS, o Salário Família e o Salário Liquido. O desconto do INSS será de 8% se o funcionário ganhar um salário bruto de até R$ 800,00(inclusive), de 12% se ganhar entre R$ 800,00 e R$ 2000,00(inclusive) e de 20% se for acima de R$ 2000,00. O Salário família será acrescido pelo número de dependente (R$ 12,00 por dependente).

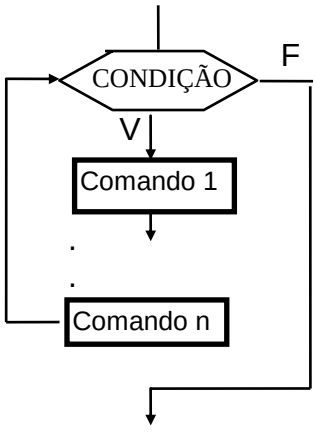
16 – Fazer um algoritmo que leia as três medidas de um triângulo, escreva se o mesmo é escaleno (todos os lados diferentes), isósceles (dois lados iguais) ou equilátero (todos os lados iguais). Considere que as medidas são válidas para formar um triangulo.

17 – Fazer um algoritmo para calcular o preço pago por uma conta telefônica, para isso você terá que ler o número de minutos gastos com ligações comuns e também com ligações para celulares. O cálculo será feito da seguinte forma: até 200 minutos o usuário pagará R$ 45,00 de tarifa básica; até 240 minutos o usuário pagará a tarifa básica mais R$ 0,10 por minuto excedente; até 540 minutos o usuário pagará a tarifa básica, mais R$ 0,15 por minuto excedente e mais uma alíquota de 10% somente sobre os minutos excedentes. Acima de 540 minutos, o usuário pagará a tarifa básica, mais R$0,20 por minuto excedente e mais 15% de alíquota sobre os minutos excedentes. Para ligações de celulares o preço será de R$ 0,95 por minuto mais uma alíquota de 22% sobre o total de minutos gastos. Você deverá imprimir discriminadamente o total gasto com ligações simples, o total com ligações de celulares e um total da conta.

3.16.3 Repetição

Quando um conjunto de ações é executado repetidamente enquanto uma determinada condição (expressão lógica) permanece verdadeira. Dependendo do resultado do teste da condição, o conjunto de comandos poderá não ser executado nem uma vez (se for falsa no 1º teste), ou várias vezes (Enquanto for Verdadeira). Chama-se isso de laço (**LOOP**).

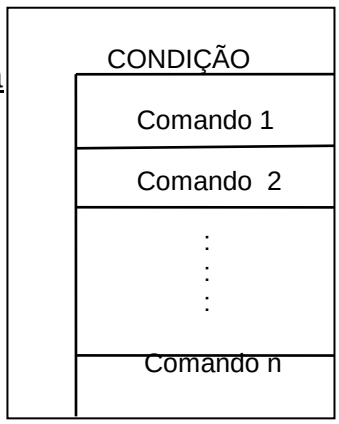
Fluxograma



Pseudo-Código

```
Inicio
Enquanto Condição Faça
        Comando 1
        Comando 2
        :
        Comando n
Fim-Enquanto
Fim
```

NS



Exemplo

Faça um algoritmo que leia n números e ao final escreva a soma deles. Para finalizar basta que digite o número zero.

```
Programa Exemplo
    Var
            Soma:Inteiro
            Num: Inteiro
    Inicio
    |       Leia (Num)
    |       Soma = 0   (* Variável que soma de todos os número lidos *)
    |       Enquanto Num <> 0 Faça
    |       |       Soma = Soma + Num
    |       |       Leia (Num)
    |       Fim-Enquanto
    |       Escreva (“A soma dos valores é :”, Soma)
    Fim
```

Exercícios

1 - Fazer um algoritmo que leia 10 números inteiros e ao final escreva a soma, a média aritmética e a média geométrica dos números.

2 - Fazer um algoritmo para escrever na tela os números de 1 até 20.

3 - Fazer um algoritmo para imprimir na tela o quadrado dos números de 1 até 20.

4 - Fazer um algoritmo para imprimir na tela o quadrado dos números ímpares compreendidos entre(compreende os limites) 1 e 100.

5 - Fazer um algoritmo que leia 30 registros contendo cada um: Nome, Idade, Sexo e duas notas(P1 e P2). Deverá ser escrito uma lista contendo o Nome, sexo e a média do aluno. Ao final escreva: A média das idades, o total de pessoas do sexo masculino e o total de pessoas do sexo feminino.

6 - Fazer um algoritmo para imprimir os 50 primeiros números pares.

7 - Fazer um algoritmo que leia 20 números inteiros e escreva o menor deles.

8 - Fazer um programa para gerar e imprimir a série 1, 5, 9, 13, 17,.....n. Imprimir a soma deles até o 50º termo(inclusive).

9 - Fazer um algoritmo que leia 30 temperaturas de uma determinada cidade, isto é, cada dia é fornecida a temperatura média daquele dia. Pede-se imprimir a menor e a maior temperatura ocorrida naquele mês.

10 - Faça o exercício 14 da lista anterior, supondo que existe uma turma de 45 alunos. Deverá ser calculada a média da turma, além do que agora deverá ser lido também o sexo do aluno (M – Masculino e F – Feminino) deverá ser impresso, ao final, o nome e o sexo do aluno de maior média e também quantos alunos foram reprovados e quantas alunas foram aprovadas.

11 - Fazer um algoritmo que leia 20 números inteiros e escreva o menor e o maior deles.

12 - Fazer um algoritmo que imprima a série $\left(\frac{1}{2},\frac{1}{4},\frac{1}{16},\frac{1}{256},.....\right)$ até o 56º elemento.

13 - Fazer um algoritmo que escreva a soma da série $\left(\frac{1}{2},\frac{3}{4},\frac{5}{6},.....\right)$ até o 45º elemento.

14 - Fazer um programa que calcule e imprima a soma da série até o 20º elemento.
$\left(1,\frac{1}{2},\frac{3}{16},\frac{1}{16},\frac{5}{256},....\right)$. Dica Denominador 4^0, 4^1, 4^2, 4^3, 4^4,....E numerador números sequenciais.

15 - Calcule o fatorial de um nº lido até 12. Deverá ser verificado se o número lido não está acima de 12. Ex.: 5! = 5.4.3.2.1= 120.

16 - Uma coleção de 30 bolas numeradas de 1 a 30 e coloridas (azul, verde, vermelha e preta) estão dentro de uma urna. Vamos retirando uma a uma e contabilizando. Pede-se:

- Total de bolas vermelhas
- Total de bolas azuis e Pares
- Total de bolas verdes ou ímpares.
- Total de bolas retiradas.

O algoritmo termina quando for retirada uma bola preta. As bolas pretas não têm numeração e não são contabilizadas.

17 - Fazer um programa que calcule e imprima a soma da série até o 10º elemento.
$\left(0,1,\frac{10}{25},\frac{3}{25},\frac{4}{125},\frac{1}{125},....\right)$.

18 - O setor imobiliário é um segmento do mercado responsável por um grande volume de transações e por um importante significado social. A imobiliária XYZ contratou você para desenvolver um algoritmo que controla os imóveis de sua carteira. Cada imóvel quando registrado na imobiliária consta dos seguintes itens: Código do imóvel (um número inteiro sequencial de 1 a 50), Tipo do imóvel (C – Casa, A – Apartamento), metragem, número de quartos, número de banheiros, existência de garagem(S – sim ou N – não), a condição do imóvel (1 – efetivamente alugado, 2 – para ser alugado, 3 – efetivamente vendido 4 – a ser vendido) e o preço. O programa termina quando for digitado zero no código do imóvel.No final do mês o dono da imobiliária precisa saber:

a) Total de imóveis para serem alugados
b) Total de casas que estão efetivamente alugadas
c) Total de apartamentos que estão efetivamente alugados
d) Total de imóveis que estão à venda;
e) Quantos apartamentos tem garagem?
f) Quantidade de faturamento bruto de aluguel do mês
g) Quanto que a imobiliária retirou com a comissão de vendas(5% sobre o preço do imóvel).
h) Total de casas com 2 ou mais banheiros

19 - O padrão de comportamento de uma sequência de números é: A partir do terceiro termo (inclusive) cada termo é obtido somando-se os dois termos antecedentes. A sequência será: 1, 1, 2, 3, 5, 8, 13, 21, ... Lembrando que o total de termo é lido através da variável N . Pede-se: Imprimir a série e a soma dos elementos.

OBS: Num algoritmo muitas vezes precisamos usar um artifício para sabermos quando termina um arquivo, nesse caso iremos usar um sinalizador chamado FLAG, ele geralmente sinalizará que é fim de arquivo e sairá do LOOP.

3.16.4 Repetição com uma variável de Controle

É uma estrutura semelhante a repetição, a diferença é que aqui é necessário saber a priori o número de vezes que será executado o LOOP. A vantagem desse sobre o anterior é que dispensa a inicialização e modificações de variáveis de controle do LOOP, isso será feito automaticamente, além do valor do incremento do índice ser definido pelo programador.
Formato:

```
Variando CONT de LI a LF [Passo P], Faça
|       comando1
|       comando2
|       :
Fim Variando
```

Obs: As representações do Fluxograma e NS são o inverso da representação de Repetição
CONT : é a variável de controle do laço, isto é, o número de vezes que o laço está sendo executado;
LI : é o limite inferior da variável de controle; o inicio da contagem;
LF: é o limite superior da variável de controle; o fim da contagem;
P: é o passo entre dois valores de CONT; podendo ser positivo(1, 2,...) ou negativo (-1, -2,..) e também é opcional.

Exemplo:

Faça um algoritmo que escreva todos os números ímpares entre 1 e 60.
Programa exemplo

```
Var
        I : Inteiro
Inicio
|       Variando  I  de 1 a 60 passo 2  Faça
|       |
|       |       Escreva ( I )
|       Fim Variando
Fim
```

Exercícios

1 - Faça um algoritmo que escreva 30 vezes o nome e a sua idade.

2 - Faça um algoritmo que escreva os números de uma P.A., sabendo que o 1º termo é 1 e o último termo é 31 e sua razão é 3. Ao final escreva a quantidade de números da P.A .

3 - Faça um algoritmo para imprimir os 50 primeiros números naturais inteiros em ordem decrescente.

4 - Faça um algoritmo que escreva os 30 primeiros números impares.

5 - Faça um algoritmo que escreva todos os múltiplos de 3 ou de 7 entre 20 e 100.

6 - Faça um algoritmo que escreva todos os anos em ordem decrescente até o ano de seu nascimento.

7 - Faça um algoritmo que escreva todas as potências (de 2) dos números pares de 1 até 100 (inclusive).

8 - Faça um algoritmo que escreva a raiz quadrada de todos os números ímpares de 1 até 167.

9 - Faça um algoritmo para escrever os 30 primeiros números primos

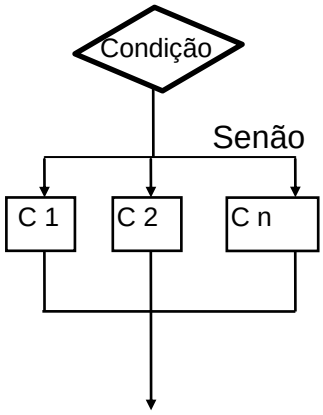
10 - Faça um algoritmo que escreva os números pares maiores que 50 e menores que 349.

11 – Faça um algoritmo que escreva as letras do alfabeto de forma decrescente

3.16.5 Seleção dentre Múltiplas Alternativas

Quando uma variável puder assumir vários valores pré-definidos.

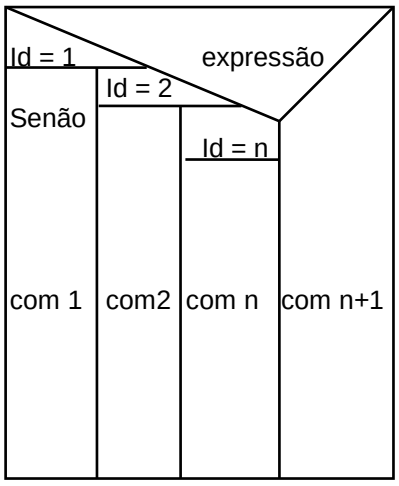
Fluxograma



Pseudo-Código

```
Inicio
    Conforme
        Caso ident = 1
                comando 1
        Caso ident = 2
                comando 2
        :
        Senão
                comando n
    Fim Conforme
Fim
```

NS



OBS.: É muito útil esse comando quando estamos trabalhando com subrotinas dentro de um algoritmo

Exemplo:

Imagine um arquivo contendo Nome, Salário, Classe. Caso o funcionário seja da classe A seu desconto será de 10% para o INSS, se for classe B seu desconto será de 8%, se for classe C será 6% de outra classe o desconto será Nulo.

```
Inicio
        Leia ( arqfun, Nome, Salario, Classe)
        Conforme
                Caso Classe =”A”   desconto = Salario * 10%
                Caso Classe = “B”  desconto = Salario * 8%
                Caso Classe = “C”  desconto = Salario * 6%
                senão   desconto = 0
        Fim Conforme
        Escreva ( “O desconto é : ”, desconto )
Fim
```

Exercícios

1 - Faça um algoritmo que leia de 100 números. Ao final escreva a soma dos números pares, a soma dos números ímpares e maior soma dentre elas. Deverá ser impresso cada número par lido. Usar o comando CASE.

2 - Escreva um algoritmo que leia um arquivo de uma turma contendo 5 registros, onde cada um contém:

- Nome
- Altura
- Sexo (M, F)

Pede-se:

- Escreva a menor altura das meninas e seu nome;
- Escreva a maior altura entre os meninos e seu nome;
- Escreva a média de altura da turma.

3 - O movimento do mês de uma empresa foi registrado em 30 operações onde cada uma contém:

- Valor
- Data da transação
- Tipo (D - débito, C - Crédito).

Pede-se:

Escreva o Valor e o tipo de transação
Escreva o Total de débitos da empresa no mês
Escreva o total de Créditos da empresa no mês
Escreva o Resultado final: Se a empresa teve ou não lucro e de quanto foi

4. Estrutura de Dados Agregados Homogêneos: Vetores e Matrizes

4.1 Vetores

Um vetor ou agregado homogêneo, ou ainda uma variável composta homogênea, é uma estrutura de dados que contém elementos do mesmo tipo, que podem ser referenciados como um todo. Da mesma forma que os registros, ao declararmos um valor, estamos reservando na memória principal do computador uma série de células para uso da variável desse tipo. O nome do vetor aponta para a base das células e o seu índice dá a posição relativa do elemento referenciado ao primeiro.
Exemplo
NUM(I)

32	87	5	34	4	5	123	23	58	94	13	0

onde,

NUM = é o nome do vetor
I = é o índice do vetor que varia de 1 até 12

NUM(4) = 34, NUM (7) = 123 e assim por diante.

Obs.: O índice do vetor poderá começar com o valor zero(0)

Declaração de Vetores:

```
Tipo
      Num = vetor[1..12] de Inteiros
Var
```

Vetnum : Num

Exercícios
1 - Fazer um algoritmo que leia um vetor com 50 posições inteiras e escreva a soma de seu conteúdo.

2 - Fazer um algoritmo que a partir de um vetor de 100 posições inteiras positivas, escreva o conteúdo das posições pares e ao final escreva a soma das posições ímpares.

3 – Faça um algoritmo que leia um vetor de 30 posições de caracteres e ao final imprima:
- Quantas vezes a palavra “LUA” aparece nesse vetor.
- O Total de vogais de vogais desse vetor.

4 - Faça um algoritmo que leia um vetor de 50 posições de números inteiros positivos. O algoritmo consiste em ordenar o vetor de forma crescente. Deverá ser usada somente uma estrutura de vetor.

4.2 Matrizes

São estruturas de dados similares ao vetor (agregados homogêneos), com a diferença que podem ter várias dimensões. Cada célula da matriz será indicada pelo nome e tantos índices quantas forem as dimensões da matriz. No nosso curso trabalharemos só com duas dimensões, mas nada impede de ter mais.

4.2.1 Definição

Exemplo

$$A_{i\ x\ j} = \begin{bmatrix} 2 & 5 & 7 & 4 & 8 \\ 9 & 3 & 7 & -8 & 2 \\ 3 & 5 & -3 & -7 & -2 \\ 6 & 6 & 4 & 1 & 0 \end{bmatrix}_{4x5}$$

A representação anteriormente é uma **matriz** e cada número dentro da matriz é chamado de **elemento** da matriz. A matriz pode ser representada entre parênteses ou entre colchetes. No exemplo a matriz A é do tipo 4 x 5 (quatro por cinco), isto é, 4 linhas por 5 colunas

Onde, i = Linha da matriz
j = Coluna da Matriz

Para indicarmos a ordem da matriz, dizemos primeiro o número de linhas e, em seguida, o número de colunas.

Exemplo

$\begin{bmatrix} 2 & 4 & -1 \\ 4 & 7 & 3 \end{bmatrix}$ matriz de ordem 2 x 3 (2 Linhas e 3 colunas)

$\begin{bmatrix} 9 & -4 \end{bmatrix}$ matriz de ordem 1 x 3 (1 Linha e 3 colunas)

4.2.2 Representação Algébrica

Utilizamos letras minúsculas para indicar matrizes genéricas e letras maiúsculas correspondentes aos elementos. Algebricamente a matriz pode ser representada por:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}_{mxn}$$

com m e n $\in \mathbb{N}^*$. Como o A é bastante extenso podemos representar por $A = (a_{ij})_{m\ x\ n}$

a_{11} (Lê-se a um um) elemento localizado na 1ª Linha e 1ª Coluna
a_{12} (Lê-se a um dois) elemento localizado na 1ª Linha e 2ª Coluna

4.2.3 Matriz Quadrada

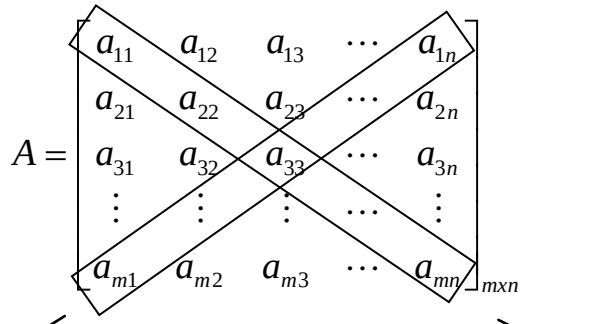
Se o número de linhas de uma matriz for igual ao número de colunas, a matriz é dita **Quadrada.**

Ex.: $A = \begin{bmatrix} 1 & 4 \\ 3 & -4 \end{bmatrix}$ é uma matriz de ordem 2

$B = \begin{bmatrix} 1 & 3 & 7 \\ 3 & 6 & 7 \\ 4 & 5 & -2 \end{bmatrix}$ é uma matriz de ordem 3

Observações:

- Quando uma matriz tem todos os seus elementos iguais a zero, dizemos que é uma matriz nula.
- Os elementos a_{ij} onde i = j, formam a **diagonal principal.**



Diagonal Secundária **Diagonal Principal**

4.2.4 Matriz Unidade ou Identidade

A matriz quadrada de ordem n, em que todos as elementos da diagonal principal são iguais a 1 (um) e os demais elementos iguais a 0 (zero) é denominada matriz identidade.
Representação: I_n

$$I_n = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.2.5 Matriz Transposta

Se A é uma matriz de ordem m x n , denominamos transposta de A a matriz de ordem n x m obtida pela troca ordenada das linhas pelas colunas.

Representação : A^t
Exemplo

$$A=\begin{bmatrix}1 & 3 & -9\\ 3 & 5 & 4\\ 6 & 5 & 0\end{bmatrix} \quad \text{então a transposta será} \quad A^t=\begin{bmatrix}1 & 3 & 6\\ 3 & 5 & 5\\ -9 & 4 & 0\end{bmatrix}$$

4.2.6 Operação com Matrizes

4.2.6.1 Adição e Subtração

A adição ou a subtração de duas matrizes A e B, efetuada somando-se ou subtraindo-se os seus elementos correspondentes e deverá ter a mesma dimensão.
Exemplo

Seja $A=\begin{bmatrix}1 & -8\\ 4 & 5\end{bmatrix}$ e $B=\begin{bmatrix}2 & -2\\ -3 & 4\end{bmatrix}$ então

$$A+B=\begin{bmatrix}1 & -8\\ 4 & 5\end{bmatrix}+\begin{bmatrix}2 & -2\\ -3 & 4\end{bmatrix}=\begin{bmatrix}3 & -10\\ 1 & 9\end{bmatrix}$$

$$A-B=\begin{bmatrix}1 & -8\\ 4 & 5\end{bmatrix}-\begin{bmatrix}2 & -2\\ -3 & 4\end{bmatrix}=\begin{bmatrix}-1 & -6\\ 7 & 1\end{bmatrix}$$

4.2.6.2 Multiplicação

Na multiplicação de duas matrizes A e B, o número de colunas de A tem que ser igual ao número de linhas de B ; O produto AB terá o mesmo número de linhas de A e o mesmo número de colunas de B.

$$A_{mxn}\times B_{nxp}=(A\times B)_{mxp}$$

Sejam as matrizes A e B, então a multiplicação das matrizes é:

$$A=\begin{bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}_{3x3} \quad e \quad B=\begin{bmatrix}b_{11} & b_{12}\\ b_{21} & b_{22}\\ b_{31} & b_{32}\end{bmatrix}_{3x2}$$

$$A\times B=\begin{bmatrix}a_{11}b_{11}+a_{12}b_{21}+a_{13}b_{31} & a_{11}b_{12}+a_{12}b_{22}+a_{13}b_{32}\\ a_{21}b_{11}+a_{22}b_{21}+a_{23}b_{31} & a_{21}b_{12}+a_{22}b_{22}+a_{23}b_{32}\\ a_{31}b_{11}+a_{32}b_{21}+a_{33}b_{31} & a_{31}b_{12}+a_{32}b_{22}+a_{33}b_{32}\end{bmatrix}$$

Obs.: A multiplicação de um número real K qualquer por uma matriz é feita multiplicando cada um dos elementos da matriz pelo número real K.
Exemplo:

Seja K = 3

Se a matriz $A = \begin{bmatrix} 2 & 0 \\ 4 & -5 \end{bmatrix} \Rightarrow k \times A = \begin{bmatrix} 6 & 0 \\ 12 & -15 \end{bmatrix}$

4.2.7 Matriz Oposta

Denominamos de matriz oposta a matriz cujos elementos são simétricos dos elementos correspondentes.

Exemplo

$$A = \begin{bmatrix} -2 & 7 \\ 1 & 9 \end{bmatrix} \quad \Rightarrow \quad -A = \begin{bmatrix} 2 & -7 \\ -1 & -9 \end{bmatrix}$$

4.2.8 Matriz Simétrica

Denominamos de matriz simétrica a matriz cujos elementos fora da diagonal principal são iguais($A_{ij} = A_{ji}$).

Exemplo

$$A = \begin{bmatrix} -1 & 2 & -7 \\ 2 & 4 & 5 \\ -7 & 5 & 6 \end{bmatrix}$$

4.2.9 Matriz Anti-Simétrica

Denominamos de matriz anti-simétrica a matriz cujos elementos fora da diagonal principal são iguais($A_{ij} = -A_{ji}$).

Exemplo

$$A = \begin{bmatrix} -1 & 2 & 17 \\ -2 & 6 & -3 \\ -17 & 3 & -2 \end{bmatrix}$$

4.2.9 Propriedades

Adição

1. A + B = B + A (Comutativa)
2. (A + B) + C = A + (B + C) (Associativa)
3. A + 0 = A (Elemento Neutro)
4. A + (-A) = 0 (Elemento Oposto)

Multiplicação

1. A . (BC) = (AB) . C (Associativa)
2. A . (B + C) = AB + AC (Distributiva à Direita)
3. (B + C) . A = BA + CA (Distributiva à Esquerda)

Obs.: A multiplicação de matrizes não é comutativa, mas se ocorrer AB = BA, dizemos que as matrizes A e B comutam.

4.2.10 Matriz Inversa

Seja A uma matriz quadrada, se existir uma matriz B tal que A.B = B.A = I, dizemos que a matriz B é uma matriz inversa de A e a indicaremos por A^{-1}. Vale lembrar que I é a matriz identidade. Caso exista a inversa dizemos que a matriz A é *inversível* e, em caso contrário, *não inversível* ou *singular*. Se a matriz quadrada A é inversível, a sua inversa é única.

$$A \cdot A^{-1} = A^{-1} \cdot A = I_n$$

Exemplo:

Determinar a inversa da matriz $A=\begin{bmatrix}1 & 2\\ 3 & 2\end{bmatrix}$.

Fazendo $A^{-1}=\begin{bmatrix}a & b\\ c & d\end{bmatrix}$ e sabendo que $A\cdot A^{-1}=I_n$ então teremos :

$$\begin{bmatrix}1 & 2\\ 3 & 2\end{bmatrix}.\begin{bmatrix}a & b\\ c & d\end{bmatrix}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\Rightarrow\begin{bmatrix}a+2c & b+2d\\ 3a+2c & 3b+2d\end{bmatrix}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

$$\begin{cases}a+2c=1\\ 3a+2c=0\end{cases}\Rightarrow a=\frac{-1}{2}\quad e\quad c=\frac{3}{4}$$

$$\begin{cases}b+2d=0\\ 3b+2d=1\end{cases}\Rightarrow\quad b=\frac{1}{2}\quad e\quad d=\frac{-1}{4}$$

Logo a matriz inversa é: $A^{-1}=\begin{bmatrix}\frac{-1}{2} & \frac{1}{2}\\ \frac{3}{4} & \frac{-1}{4}\end{bmatrix}$

4.2.11 Determinantes

É um número real associado a matriz quadrada.

4.2.11.1 Determinante de uma matriz quadrada de 2ª Ordem

O determinante de uma matriz quadrada de 2ª ordem é obtido pela diferença entre o produto dos elementos da diagonal principal e o produto dos elementos da diagonal secundária.

$$\det A=|A|=\begin{vmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{vmatrix}=a_{11}\cdot a_{22}-a_{12}\cdot a_{21}$$

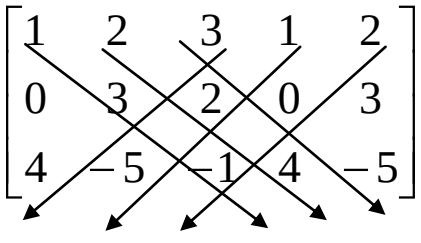
Exemplo

Achar o valor do determinante $\begin{vmatrix}4 & 5\\ -1 & 7\end{vmatrix}=4\times 7-5\times(-1)=28-(-5)=33$

4.2.11.2 Determinante de uma matriz quadrada de 3ª Ordem

Para calcular o determinante de uma matriz quadrada de 3ª ordem utilizaremos uma regra muita prática denominada regra de Sarrus.

Seja a matriz $A=\begin{bmatrix}1 & 2 & 3\\ 0 & 3 & 2\\ 4 & -5 & -1\end{bmatrix}$. Para Calcularmos o determinante vamos repetir a 1ª e a 2ª colunas à direita da matriz, conforme o esquema abaixo:



\- - - + + +

det A = | A | = (1)(3)(-1) + (2)(2)(4) + (3)(0)(-5) – (2)(0)(-1) – (1)(2)(-5) – (3)(3)(4)

det A = | A | = -3 + 16 – 0 + 0 + 10 – 36 = -13

4.2.12 Declaração de uma Matriz no Algoritmo

```
Tipo
        TPMATRIZ = Vetor[1..4,1..5] de Inteiros

Var
        MAT = TPMATRIZ
```

Obs.: A melhor estrutura de Controle para se trabalhar com Vetores e/ou Matrizes é a REPETIÇÃO com uma variável de controle

Exercícios

1 - Fazer um algoritmo que escreva a soma da diagonal principal de uma matriz 5x5.

2 - Fazer um algoritmo que calcule a soma dos valores marcados na matriz abaixo, imprimir também a multiplicação dos valores restantes:

Mat =

12	4	36	7	9
12	83	2	-4	-3
34	-67	-5	1	35
56	4	17	21	-24
2	3	-8	-12	77

3 - Faça um algoritmo que leia um arquivo de notas dos 30 alunos de uma classe. Cada aluno possui 4 notas. Deverá ser armazenado em uma matriz de 4 X 30. Calcule a média de cada aluno e também a média geral da turma.Use um vetor para armazenar os nomes.

4 – Faça um algoritmo que leia uma matriz 5 x 5 de valores inteiros e gere e imprima um vetor com todos os valores das colunas ímpares dessa matriz e também um vetor com todos os números pares dessa matriz.

5. Modularização de Algoritmos/Programas

A modularização traz algumas vantagens para o entendimento, construção, codificação, teste e reutilização dos mesmos. A grande maioria das linguagens de programação tem essa facilidade. Seja como subrotinas, procedimentos, funções, subprogramas etc. Sempre é possível subdividir um programa de modo a facilitar o entendimento, permitir a reutilização, evitando-se a repetição de blocos. No pseudo-código definiremos dois tipos de módulos: os procedimentos (“procedure”) e as funções (“functions”).

Os procedimentos e funções são normalmente definidos antes de serem utilizados pelo programa principal. Nos nossos algoritmos poderemos defini-los em qualquer parte do algoritmo, seguindo os formatos e as normas de documentação.

5.1 Variável Global e Local

Uma variável Global será declarada no inicio do programa, e poderá ser utilizada por qualquer sub-rotina ou procedimento dentro do programa. Dizemos que é visível em todo o programa.

A variável Local é declarada dentro de uma sub-rotina ou procedimento e somente será válida dentro da sub-rotina que foi declarada.

Exemplos:

```
Program Um;
Var
        A:integer;
        B:Integer;
        X:integer;
Begin(*Inicio do prog. Principal*)
        Writeln('Digite um número ');
        Readln(A);
        Writeln('Digite número dois');
        Readln(B);
        X:=A;
        A:=B;
        B:=X;
        Writeln(A);
        Writeln(B);
End.
```

```
Program Dois;
Var
        A:integer;
        B:Integer;
Procedure Troca;
Var
        X:integer;
Begin
        X:=A;
        A:=B;
        B:=X;
End;
Begin(*Inicio do prog. Principal*)
        Writeln('Digite um número ');
        Readln(A);
        Writeln('Digite número dois');
        Readln(B);
        Troca;
        Writeln(A);
        Writeln(B);
End.
```

Apesar de o programa dois ser um pouco maior que o primeiro em número de linhas, no segundo há uma economia de memória, uma vez que quando não está sendo utilizada a rotina TROCA, a variável X será liberada, ficando somente os espaços reservados para variáveis Globais. Caso déssemos um comando de impressão da variável X dentro do programa dois(na parte principal) teríamos como resposta uma mensagem de erro: “Variável fora do Escopo”, pois se trata de uma variável local.

Uma observação importante quanto a declaração de variáveis: Estas só serão globais referentes as sub-rotinas e procedimentos que forem colocadas após a declaração. Por isso se quiser que as variáveis sejam globais a todas as rotinas, deverão ser declaradas em primeiro lugar no programa. Se declarada antes, esta será global a todas as sub-rotinas existentes após a sua declaração, porém, se forem declaradas após uma sequência de sub-rotinas, esta será global do ponto que está para baixo e local com relação às sub-rotinas definidas acima dela(e que devem estar declaradas dentro das rotinas anteriores, pois senão teremos um erro de escopo!!!!). Veja o programa a seguir:

```
Program Tres;
Procedure Troca;
        Var
                X:integer;
                A:integer;
                B:integer;
        Begin
                A:=9;
                X:=A;
                A:=B;
                B:=X;
                Writeln(A);
                Writeln(B);
                Writeln(X);
        End;
        Var
                A:integer;
```

```
                B:Integer;
    Begin(*Inicio do prog. Principal*)
        Writeln('Digite um número: ');Readln(A);
        Writeln('Digite um número: ');Readln(B);
        Troca;
        Writeln(A);
        Writeln(B);
    End.
```

Sua saída seria:
Digite um número: 12
Digite um número: 8
0
9
9
12
8

Exercícios
a) Faça um programa que crie uma calculadora com as quatro operações básicas. Deverá ser usado um sub-rotina para cada operação com suas variáveis locais e um menu(Utilizando o comando “Case”) no programa principal que chamará essas subrotinas quando solicitadas. Deverá também ter pelo menos uma variável Global. Deverá ser usado o comando gotoxy que define a posição do cursor do teclado na tela.

Sintaxe

```
gotoxy ( Coluna: integer ; Linha: integer ) ;
```

```
Program Exemplo;
{ Programa que move o cursor do teclado para a Coluna 10, Linha 20, e imprime "Programação" na tela. As linhas são de 1 á 25 e as colunas de 1 à 80}
    Begin
        gotoxy(10,20);
        write('Programação');
    End.
```

5.2 Sub-Rotinas

A modularização dos programas pode ser feitos por meio de procedures (procedimentos) com ou sem passagem de parâmetros, funções(functions) e Units(unidades).

5.2.1 Procedimento sem Passagem de Parâmetros.

Exemplo

```
Program Primo_Proced;
 Var
    N:integer;{Número a ser gerado para verificar se é primo}
    Primo:integer;{Conta os números primos}
    Divi:Integer;{assume os divisores 2,3,4,5,6,7....}
    Achou:Boolean;{ Quando encontra um primo ele avisa}
  Procedure Procura_Primo;
    Begin
        Achou:=False;
        IF N >2 Then
        For Divi:=2 to N-1 Do
            If N mod Divi =0 Then
```

```
                    Achou:=True;
        End;
 Procedure Imprima_Primo;
   Begin
        If Not achou Then
                Begin
                    Writeln(N, '  É Primo');
                    primo:=primo+1;
                End;
        End;
Begin (*Inicio do pgm principal*)
        Primo:=0;
        N:=1;
        While primo< 30 Do
            Begin
                Procura_Primo;
                Imprima_Primo;
                N:=N+1;
            End;
End.(*Fim do programa principal*)
```

5.2.2 Procedimento com Passagem de Parâmetros

Esses parâmetros podem ser **Formais ou Reais**. Serão formais quando forem declarados através de variáveis juntamente com a identificação do nome da sub-rotina, os quais serão tratados exatamente da mesma forma que são tratadas as variáveis globais ou locais. Os parâmetros Reais são aqueles que estão presentes na chamada da rotina.

Exemplo:

```
Program Soma_Raiz;
Var
    {Programa que lê dois valores e chama uma rotina que
    calcula as suas raízes e soma os valores}
    X:integer;{Número a ser Lido- Parâmetro Real}
    Y:Integer;{Número a ser Lido- Parâmetro Real}
Procedure Calc_Soma_Raiz(A,B:Integer);
    (* A e B são os parâmetros Formais*)
    Var
        Z: Real;
    Begin
        Z:=SQRT(A)+SQRT(B);
        Writeln(Z:8:3, ' É a soma das raízes quadradas de : ', A, ' e ',B);
    End;
Begin (*Inicio do pgm principal*)
    Write('Digite o valor de um número positivo inteiro: ');Readln(X);
    Write('Digite o valor de um número positivo inteiro: ');Readln(Y);
    Calc_Soma_Raiz(X,Y);(*X e Y parâmetros reais*)
    Calc_Soma_Raiz(49,81); (*49 e 81 parâmetros reais*)

End.(*Fim do programa principal*)
```

Essa passagem de parâmetros poderá ser feita de duas formas: Por Valor e Por Referência.

5.2.2.1 Por Valor

Esta modalidade caracteriza-se pela não alteração do valor do parâmetro real quando o parâmetro formal é manipulado dentro da sub-rotina. Somente uma cópia do parâmetro real é passada para o parâmetro formal e qualquer modificação feita, dentro da rotina, só afetará a variável local da sub-rotina e não o parâmetro real.

Exemplo

```
Program Fatorial_Valor;
    Uses
        Crt;
    Procedure Fatorial(N:Integer);
        Var
            I, Fat:Integer;
        Begin
            Fat:=1;
            For I:=1 to N do
                Fat:=Fat*I;
            Writeln('O fatorial de ', N, ' equivale a : ', Fat);
            N:=N-1;(*Para mostrar que não vai alterar o parâmetro fora do
procedimento*)
            Writeln(N);
        End;
    Var
        Lim:Integer;
        Resp:Char;
    Begin
        Clrscr;
        Writeln('Deseja calcular um fatorial ? S/N');Readln(Resp);
        While Resp='s' do
            Begin
                Write('Informe um valor inteiro : ');
                Readln(Lim);
                Fatorial(Lim);
                Writeln;
                Writeln('Deseja Continuar ? S/N');Readln(Resp);
            End;
    End.
```

5.2.2.2 Por Referência

Esta modalidade caracteriza-se pela ocorrência da alteração do valor do parâmetro real quando o parâmetro formal é manipulado dentro da sub-rotina. A alteração efetuada é devolvida para a rotina chamadora

Exemplo:

```
Program Fatorial_Referencia;
    Uses
        Crt;
    Procedure Fatorial(N:Integer; var Fat:integer);
        Var
            I:Integer;
        Begin
            Fat:=1;
            For I:=1 to N do
                Fat:=Fat*I;
        End;
```

```
Var
        Limite:Integer;
        Retorno:Integer;
        Resp:Char;
Begin
        Clrscr;
        Writeln('Programa Fatorial ');
        Writeln('Deseja calcular um fatorial ? s/n');Readln(Resp);
        While Resp='s' do
                Begin
                        Write('Informe um valor inteiro : ');
                        Readln(Limite);
                        Fatorial(Limite, Retorno);
                        Writeln('O Fatorial de ',Limite,'  equivale a : ',Retorno);
                        Writeln;
                        Writeln('Deseja Continuar ? s/n');Readln(Resp);
                End;
End.
```

Obs.: Neste exemplo a variável “N” continua sendo do tipo: “Passagem de parâmetro por valor”, pois ela apenas receberá o valor da variável “Limite” através da chamada da sub-rotina “Fatorial”; A variável “Fat” é do tipo “passagem de parâmetro por referência” (pois aparece através da instrução “var” na declaração do nome da sub-rotina), a “Fat” será alterada dentro da sub-rotina e passará o cálculo do fatorial para a variável “Retorno” dentro do programa principal.

5.2.3 Funções – Function

Uma function tem o mesmo objetivo de uma procedure, isto é, desviar a execução do programa principal para a realização de uma tarefa específica com uma única diferença: O Valor de uma função é retornado no próprio nome da função e quando mencionamos “Valor” devem ser levados em conta os valores numéricos, lógicos ou literais(String ou Char). Somente não poderão ser retornados tipos ARRAY e RECORD, justamente por serem tipos que definem variáveis que armazenam mais de um valor. Ela poderá usar variáveis Locais e/ou Globais. A Function pode ser com ou sem passagem de parâmetros.
Sintaxe :

Function <nome da Função>[parâmetros:Tipo dos dados]: <Tipo do dado do valor retornado>;

```
        Var
                (* declaração das variáveis Locais*)
        Begin
                Comando 1
                Comando 2
                .....
        End;
```

Exemplo:

5.2.3.1 Function sem Passagem de Parâmetros.

```
Program Funcao_sem_passagem;
        Uses
                Crt;
        Var
                Elevado:Real;(* Recebe o valor da função no programa principal*)
                X:Real;(*base a ser elevado a um expoente 2*)
```

```
Function POT:Real;
(*Função que calcula o valor de um número intero elevado a potencia 2*)
        Begin
                POT:=EXP(2*Ln(abs(X)));
                {POT:=SQR(X);}
        End;
Begin
        Clrscr;
        WriteLn('Programa Potência de 2 ');
        Writeln('Digite um valor ');Readln(X);
        Elevado:=POT;
        Writeln('O Valor de ', X:5:2, '  Elevado a 2 é :',Elevado:5:2);
End.
```

5.2.3.2 Function com Passagem de Parâmetros.

```
Program Funcao_com_passagem;
        Uses
                Crt;
        Var
                Elevado:Integer;(* Recebe o valor da função no programa principal*)
                I:Integer;(*Valor da base Limite a ser elevado a um expoente 2*)
                J:Integer;(*Controle do For para gerar a base *)
        Function POT(Num:Integer):Integer;
        (*Função que calcula o valor de um número real elevado a potencia 2*)
                Begin
                        POT:=SQR(Num);
                End;
        Begin
                Clrscr;
                WriteLn('Programa Potencia de 2 ');
                Writeln('Digite um valor inteiro maior que zero e menor que 1000');Readln(I);
                If I<= 0 then
                        Writeln('Valor Inválido ')
                else
                        For J:=1 to I do
                                Begin
                                        Elevado:=POT(J);
                                        Writeln('O Valor de ', J, '  Elevado a 2 é :',Elevado);
                                End;
        End.
```

Obs: O valor J(que irá variar de 1 até o número que foi digitado – o valor máximo para ser calculada a potência de 2) será passado para dentro da Função pela variável “Num” onde será calculada a potência de Num e então retorna o valor pelo nome da Função(POT).

6. Registro e Arquivos

6.1 Registros ou Agregados Heterogêneos

São conjuntos de informações relacionadas entre si, que podem ser referenciadas como uma unidade e que, normalmente são compostas de informações(campos) de tipos diferentes. Eles são conhecidos como variáveis compostas heterogêneas.

6.1.1 Declaração de Registros

Declaramos os registros em nossos algoritmos como TIPO definido pelo programador. Assim cada registro poderá ter tantos campos quanto o programador deseje. O ideal do tamanho de um registro é de 10 campos. Quando forem necessário estruturas de dados maiores poderemos utilizar registros previamente definidos dentro de outros registros

Exemplo(Em algoritmo):

```
Tipo
        Regaluno = Record
                        Nome : Texto
                        Sexo : Caracter
                        Media : Real
                Fim -Record
Var
        Reg1,Reg2 : Regaluno
```

Exemplo(Em Pascal):

```
Type
        Regaluno = Record
                     Nome : String[30];
                     Sexo : Char;
                     Media : Real;
                   End;
Var
        Registro : Regaluno;
        Vet: array [1..30] of Regaluno;(* Aqui está declarado um vetor com 30 posições sendo que cada uma conterá o nome, o sexo e a média de cada aluno*)
```

6.1.2 Para Acessar um Campo do Registro:

Alguns exemplos:

```
Vet[25].Nome:=’Maria’
Read(Vet[I].Nome)
Writeln(Vet[I].Sexo)
Regaluno.Media=7.4
```

6.1.3 Exemplo Prático da Utilização de Registros

Fazer um programa que leia 4 notas bimestrais de 5 alunos e apresente no final os dados dos alunos classificados de forma crescente por nome com as suas respectivas médias.

```
Program Exemplo_7;
Type
        Bim=array[1..4] of real;
        Cadalu=record
                        Nome:string;
                        Nota:bim;
                end;
Var
        Aluno:array[1..5] of Cadalu;
        I,J,Fim:integer;
        Aux:Cadalu;
        Soma,Media:Real;
Begin
        (*Rotina de Entrada*)
```

```
        Writeln('Cadastro de Aluno');
        Writeln;
        For J:=1 to 5 do
                begin
                        writeln('Informe o nome do ',J:2,' aluno....: ');
                        Readln(Aluno[j].nome);
                        For I:=1 to 4 do
                                begin
                                        Write('Informe a ', I:2, ' Nota....: ');
                                        Readln(Aluno[J].Nota[I]);
                                end;
                end;
        (*Rotina de Ordenação*)
        For Fim:=5 downto 2 do
                For I:=1 to Fim -1 do
                        If  Aluno[I].Nome > Aluno[I+1].Nome then
                                Begin
                                        Aux:=Aluno[I+1];
                                        Aluno[I+1]:=Aluno[I];
                                        Aluno[I]:=Aux;
                                end;
        (*Rotina de Saída*)
        Writeln;
        For J:=1 to 5 do
                begin
                        Soma:=0;
                        Writeln(Aluno[J].Nome);
                        For I:= 1 To 4 do
                                Soma:=Soma+Aluno[J].Nota[I];
                        Media:=Soma/4;
                        Writeln('A Média é = ',Media:5:2)
                End;
End.
```

6.1.4 Exercícios

1) Criar um registro de Funcionários onde cada registro contém: Nome, Endereço (Rua, Número, Complemento, Bairro, Cidade, Estado e Cep) Tel(Resid,Comercial e Celular) e Salário.
2) Fazer um programa que leia um vetor contendo 15 registros de funcionários contendo cada um (Nome, sexo, idade, salário). Você deverá fazer um programa que escreva o nome do funcionário e o seu salário acrescido de uma bonificação de 10% sobre o salário, calcule e escreva a média das idades desses funcionários, bem como escrever o nome daquele que possui o maior salário. Para isso deverá usar um vetor e uma estrutura de registro.

6.2 Arquivos

É um conjunto de dados ou informações de tamanho limitado que é identificado por um nome (identificador do arquivo), e que está localizado na memória secundária.

Um arquivo é uma coleção de registros, em que cada registro é formado por campos, um desses campos é chamado de chave(Identificador do registro) e que irá ser único dentro do arquivo para distinguir dos demais.

Um Banco de Dados é um conjunto de vários arquivos e nos quais poderemos fazer algumas operações como: Inclusão, alteração, exclusão(físicas ou Lógicas), consulta etc.

O PASCAL possui dois tipos de arquivos: FILE e TEXT

6.2.1 Arquivos do Tipo File

Um arquivo do tipo FILE tem acesso aleatório. É o mais importante e também o mais utilizado. Um arquivo randômico é caracterizado pelo fato de ser possível buscar uma determinada informação em qualquer posição que a mesma se encontre, sem haver a necessidade de se percorrer todo o arquivo. O acesso a informação é direto.

6.2.2 Arquivos do Tipo Text

É um conhecido como sequencial, é um tipo especial de arquivo que, ao contrário do arquivo FILE, pode ser editado normalmente através de um editor de textos qualquer. Ele é sequencial porque a leitura tem que ser feita sequencialmente do início ao fim do arquivo, não podendo desta forma, como é feito no arquivo FILE através do comando SEEK, posicionar de forma direta em um registro em particular. Todas as informações armazenadas são texto, mesmo assim, é possível armazenar no arquivo informações de qualquer tipo de dado simples (integer, real, byte etc) as quais, ao serem fisicamente armazenadas no arquivo, serão automaticamente convertidas do seu tipo original para o tipo STRING. A leitura se processa de forma inversa, ou seja, quando é lida uma informação em um arquivo TEXT, a mesma será automaticamente convertida para o tipo da variável que irá armazenar a informação, isto é, do tipo STRING para o tipo da variável receptora da informação lida.

6.2.3 Declaração de Arquivos

```
Tipo(* Tipo File*)
        <Nome do arquivo> = File of < nome do registro>;
Var
        Arqfun : <Nome do arquivo>;
        ArqTexto : TEXT;
```

Obs: No nosso curso vamos trabalhar só com arquivo do tipo File que são os mais utilizados e os mais rápidos.

Exemplo: Imagine armazenar os nomes alunos de uma determinada turma. Onde cada registro contém: Matricula, Nome, Idade, Sexo.

```
Type   registro=Record
                        Matricula:Integer;
                        Nome:String;
                        Idade:Integer;
                        Sexo:Char;
                    End;
       Arquivo = File of registro;
Var
       Alunos:Arquivo;
       Regalu:registro;
```

6.2.4 Comandos de Arquivos em Pascal

1) Assign : Associa nomes dos arquivos físicos a variáveis locais, isto é, a variáveis do tipo arquivo com o arquivo do dispositivo da memória secundária(disco)

Sintaxe:

Assign(<nome do arquivo no programa>, ‘ caminho do arquivo no disco:\nome do arquivo no disco’)

Exemplo:

Assing(Alunos,’Alunos.Dat’)

2) Rewrite: É utilizado para abrir novos arquivos apagando todo seu conteúdo.
Exemplo:

```
Rewrite(Alunos)
```

3) Reset : É utilizado para abrir arquivos e posicionar o ponteiro no registro de nº zero sem destruir os dados existentes.
Exemplo:

```
Reset(Alunos)
```

4) Close : Serve para fechar arquivos que foram abertos pelo Rewrite ou Reset.
Exemplo

```
Close(Alunos)
```

5) Read: Ler dados que estão nos registros de um arquivo.
Exemplo

```
Read(Alunos, Regalu)
```

6) Write: Grava os dados que estão nos registros de um arquivo.
Exemplo

```
Write(Alunos, Regalu)
```

7) Seek: Serve para posicionar o ponteiro no registro desejado. O primeiro registro é o de número zero.
Exemplo:

```
Seek(Alunos,3)
```

8) Filespos: Retorna o numero do registro onde o ponteiro está localizado.
Exemplo:

```
Filepos(Aluno)
```

9) Filesize: Informa o número do registros presentes em um arquivo
Exemplo:

```
Filesize(Aluno)
```

10) NOT EOF: Verificar se é o final do arquivo
Exemplo:

```
While Not EOF Do
  Begin
      Comando 1
      Comando 2
      ……..
  End;
```

Exemplo: Fazer um programa que simule um controle dos alunos matriculados em uma determina escola. Este programa deverá ser capaz de cadastrar, alterar e excluir os nomes dos alunos. O Arquivo será composto de vários registros, onde cada registro contém: Matricula(Inteiro),Nome, Idade e Sexo. Vamos usar o arquivo do tipo TEXT.

```
Program Arquivo_Texto;(*programa arquivotexto salvo na pasta Pascal*)
Uses
    Crt;
var
```

```
   opcao:char;
   Arqalunos:text;
   Matricula:integer;
   Nome:String[30];
   Idade:Integer;
   Sexo:Char;
   Tecla:char;
(*    Rotinas de Visualização    *)

Procedure Linha;
        Var
                I:integer;
        Begin
                For I:= 1 to 80 do
                        Write(#205);
        End;
Procedure Center(Msg:String);
        Var
                tam:integer;
        Begin
                Tam:=40+length(Msg) div 2;
                Writeln(Msg:tam);
        End;
Procedure Writexy(X, Y:integer; Msg:String);
        Begin
                Gotoxy(X,Y);
                Write(Msg);
        End;

(*              Rotinas de Manipulação                *)
Procedure Arquivo;
        Begin
                Clrscr;
                Linha;
                center('Criação de Arquivo');
                Linha;
                Rewrite(Arqalunos);
                gotoxy(1,12); Center('Arquivo foi Criado');
                writexy(25,24,' Tecle algo para voltar ao menu');
                Tecla:=Readkey;
                Close(Arqalunos);
        end;
Procedure Cadastra;
        Begin
                Clrscr;
                Linha;
                Center(' Cadastramento de Registro');
                Linha;
                Append(Arqalunos);
                writexy(10,5,' Entre com a Matricula...');Readln(Matricula);
                writexy(10,6,' Entre com o Nome.....');Readln(Nome);
                writexy(10,7,' Entre com a Idade.....');Readln(Idade);
                writexy(10,8,' Entre com o Sexo.....');Readln(Sexo);
```

```
                Writeln(Arqalunos,Matricula);
                Writeln(Arqalunos,Nome);
                Writeln(Arqalunos,Idade);
                Writeln(Arqalunos,Sexo);
                Writexy(25,24,' Tecle algo para voltar ao menu');
                Close(Arqalunos);
        End;
Procedure Exibir;
        Var
                Lin:integer;
        Begin
                Clrscr;
                Linha;
                Center(' Apresentação de Registros');
                Linha;
                Lin:=5;
                Reset(Arqalunos);
                While not EOF(Arqalunos) do
                        begin
                                Readln(Arqalunos,Matricula);
                                Readln(Arqalunos,Nome);
                                Readln(Arqalunos,Idade);
                                Readln(Arqalunos,Sexo);
                                gotoxy(5,Lin);Write(Matricula);
                                gotoxy(20,Lin);Write(Nome);
                                gotoxy(35,Lin);Write(Idade);
                                gotoxy(45,Lin);write(Sexo);
                                Lin:=Lin+1;
                        end;
                Writexy(25,24,' Tecle algo para voltar ao menu');
                Tecla:=Readkey;
        Close(Arqalunos);
    End;

 (********************Programa Principal******************)

 Begin
        Opcao:='0';
        assign(Arqalunos,'C:\Pascalzim\alunos.txt');
        While(opcao<>'4') do
                Begin
                        Clrscr;
                        Linha;
                        Center('Menu principal');
                        Linha;
                        Gotoxy(28,6);Write('1.......Criar um arquivo');
                        Gotoxy(28,8);Write('2.............Cadastrar');
                        Gotoxy(28,10);Write('3.......Exibir Registros');
                        Gotoxy(28,12);write('4.......Fim do Programa');
                        Gotoxy(28,16);Write('Escolha uma opção....:  ');
                        Readln(opcao);
                        If opcao<> '4' then
                                case opcao  of
```

```
                    '1':Arquivo;
                    '2':Cadastra;
                    '3':Exibir;
                else
                    Begin
                        gotoxy(27,24);Writeln(' opção Inválida - Tecle algo ');
                        opcao:=Readkey;
                    end;
                End;
        End;
 End.
```

OBS: A tabela ASCII muito usada para fazermos formatação de tela pode ser vista através desse programa.

```
// ------------------------------------------------------------
// Este programa imprime o conteúdo da tabela ASCII
// ------------------------------------------------------------

Program PASCII ;
Var
    I: integer;
Begin
    for I:= 1 to 255 do
        write( I, chr(I), ' ' );
End.
```

No programa do arquivo foi usado o caractere de nº 205(=)

7. Linguagem Pascal

O Programa em Pascal ficaria da seguinte forma:

```
Program <Nome do programa>;
Const
Type
Var
Procedures
Functions
Begin
        <Comandos>;
End.
```

7.1 Tipos de Dados

Pascal	Descrição
a) integer	Números entre -32768 até +32767. ocupa 2 bytes na memória.
b) real	Representa os números entre 2.9 x 10^{-39} até 1.7 x 10^{38} . Ocupa 6 bytes na memória.
c) char	Um caracter da tabela ASCII. Ocupa 1 byte na memória.
d) string	Conjunto de caracteres. Ocupa de 1 a 255 bytes na memória.
e) boolean	Valor lógico. Assuma somente dois valores: TRUE(Verdade) ou FALSE(Falso). ocupa 1 byte na memória.
f) word	Números inteiros de 0 até 65535. Ocupa 2 bytes na memória.
g) byte	Números inteiros de 0 até 255. Ocupa 1 byte na memória.
h) shorint	Números entre -128 até 127 . Ocupa 1 bytes na memória.
i) longint	Números entre - 2.147.483.648 até 2.147.483.647 . Ocupa 4 bytes na memória.
j) single	Números reais entre 1.5 x 10^{-45} até 3.4 x 10^{38} . Ocupa 4 bytes na memória.
k) double	Números reais entre 5 x 10^{-324} até 1.7 x 10^{308} . Ocupa 8 bytes na memória.

7.2 Palavras Reservadas

As palavras reservadas fazem parte da estrutura da linguagem e têm significados pré-determinados. Elas não podem ser usadas como identificadores de variáveis, procedures, functions etc. Algumas das palavras reservadas são:

absolute	and	array	begin
case	const	div	do
destructor	downto	else	end
file	for	function	goto
if	in	inline	interface
label	mod	nil	not
of	or	packed	procedure
program	record	repeat	set
shl	shr	string(*)	then
to	type	until	unit
uses	var	while	xor

7.3. Constantes

Podemos definir tantas constantes quantas quisermos.

```
Const
        Nome = 'Maria';
        Idade= 25;
```

Toda vez que nos referirmos às constantes acima, o pascal substituí-las-á pelos seus respectivos valores.

7.4 Declaração de Variáveis

```
Program Declara;
Var
        Idade:shortint;
        Num_Filhos:Integer;
        Nome:String[30]      (* A variável Nome só poderá ter até 30 caracteres*);
        Altura, Peso: Real;
        Achou:Boolean;
        Sexo:Char
Begin
        <comandos>;
End.
```

7.5 Operadores Aritméticos

Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Quociente Inteiro	DIV
Resto Inteiro	Mod

7.6 Operadores Relacionais

Igual	=
Diferente	<>
Menor igual	<=
Maior igual	>=
Menor	<
Maior	>

7.7 Operadores Lógicos

E	AND
OU	OR
NÃO	NOT
OU Exclusivo	XOR

7.8 Funções Matemáticas

|X|	Abs(x)
e^x	Exp(x)
Ln x	Ln(x)
Parte inteira de um número	Trunc(x)
Parte Fracionária de um número	Frac(x)
Arredondar um número	Round(x)
Raiz Quadrada	Sqrt(x)
x^2	Sqr(x)
Sen x	Sin(x)
Cos x	Cos(x)
Arctg x	Arctan(x)

Obs.:

Se quiséssemos resolver a operação $4^5 = \exp(5 * \ln(4))$

Se quiséssemos resolver a operação $\sqrt[3]{6} = 6^{\left(\frac{1}{3}\right)} = \exp\left(\frac{1}{3} * \ln(6)\right)$

7.9 Comando de Atribuição(:=)

Exemplo

```
Program Teste;
Var
        Num: Integer;
Begin
        Num: =48;
        Num:= Num + 5;
End.
```

7.10 Comando de Entrada e Saída

```
Read( < variável>)
Write (<variável>)
```

Exemplo

```
Program Ler;
Var
        Nome: string;
        Idade: Integer;
Begin
        Write(´ Digite seu nome : ´); Readln(Nome);
        Write(´ Digite sua Idade : ´); Readln(Idade);
        Writeln;
        Writeln(´ Seu Nome é = ´, Nome)
        Writeln(´ Sua idade é = ´, Idade);
End.
```

Em Pascal, quando usamos os comandos READ e WRITE, o cursor continuará posicionado na mesma linha. Para passar para nova linha devemos acrescentar as letras LN (de LINE NEW) no final dos comandos READ e WRITE.

7.11 Comentários

Poderá ser feito em qualquer parte do programa, basta usar (*.....*) ou {....comentar...}

7.12 Formatação de Números Reais

Para formatar números reais usaremos a seguinte sintaxe:

X:7:2 significa que das 7 casas, 2 será para a parte decimal, 1 para o ponto e 4 para parte inteira.

7.13 Funções de Tratamento de Caracteres

Copy(cadeia, posição, número) → Copia da cadeia, a partir da posição dada, o número de caracteres estipulados

Length(cadeia)→Informa o tamanho da cadeia de caracteres

Pos(cadeia1, cadeia2) →Mostra, a partir de que posição, a cadeia1 aparece dentro da cadeia2

Delete(Cadeia, posição, número) →Apaga da cadeia, a partir da posição dada, o número de caracteres estipulados

Insert(cadeia1, cadeia2, posição) →Insere a cadeia1 na cadeia2 a partir da posição dada

Concat(cadeia1, cadeia2) ou Cadeia1 + Cadeia2→ Soma duas cadeias de caracteres

Exemplos:

1) Faça um programa que receba uma frase e imprima a quantidade de palavras na frase

```
Program exemplo;
Uses crt;(* Para usar algumas funções definidas como reakey, clrscr e gotoxy*)
Var
        frase,letra: string;
        qtde, tam, i : integer;
begin
        clrscr;
        qtde:=0;
        writeln('Digite uma frase');
        readln(frase);
        tam := length(frase);          {tamanho da frase}
        For i := 1 to tam do           {vai percorrer cada caractere da frase}
                Begin
                        letra := copy(frase,i,1);       {coloca cada caract. da frase na variável letra}
                        If letra = ' ' Then             {compara o caractere com espaço em branco}
                                qtde := qtde + 1;
                end;
        qtde := qtde + 1;                       {depois da última palavra não tem espaço}
        writeln('Quantidade de palavras da frase = ',qtde);
        readln;
end.
```

2) Faça um programa que leia uma frase e uma palavra. Caso a palavra contenha “Produção”, substitua pela palavra digitada.

```
Program exemplo;
Uses
        crt;
Var
        frase, palavra_digitada, palavra_frase: string;
        tam, i: integer;
Begin
        clrscr;
        writeln('Digite uma frase :');
        readln(frase);
        writeln('Digite a palavra para substituição :');
        readln(palavra_digitada);
        tam := length(frase);
        i:=1;
        while i<= tam do
                begin
                        palavra_frase := COPY(frase,i,8);
                        if palavra_frase = 'Produção' then
                                begin
                                        delete(frase,i,8);
                                        insert(palavra_digitada, frase,i);
```

```
                    end;
                tam := length(frase);
                i:= i + 1;
            end;
        writeln(frase);
        Readkey;
End.
```

7.14 Declaração de Registros

Formato:

```
        TYPE  Reg=Record
                Matricula:integer;
                Nome:string[30]
                Idade: byte;
        Var
            Reg_Aluno:Reg
```

Ou

```
        Var
            Reg_Aluno:array[1..30] of reg(* Aqui as informações de cada um dos 30 alunos serão colocadas em cada posição do vetor*)
```

Para Ler ou Imprimir os campos dos registros faremos:

```
        Readln(Reg_Aluno.Matricula)
        Readln(Reg_Aluno.Nome)
        Readln(Reg_Aluno.Idade)
```

ou

```
        For i=1 to 30 do
            Begin
                Readln(Reg_Aluno[i].Matricula);
                Readln(Reg_Aluno[i].Nome);
                Readln(Reg_Aluno[i].Idade);
            End;
```