



SENAI - SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL
C.F.P. DE PONTA GROSSA

CURSO TÉCNICO EM INFORMÁTICA
DISCIPLINA: TÉCNICAS DE PROGRAMAÇÃO I

Prof. Ademir Mazer Jr

SUMÁRIO

CONTEÚDO PROGRAMÁTICO.....	8
HISTÓRIA DA LÓGICA	9
INTRODUÇÃO A LÓGICA PROPOSICIONAL	10
Preliminares	10
Proposição.....	10
Conectivos Lógicos	11
Valor Lógico	11
Princípios Fundamentais da Lógica	12
Tabela - Verdade	12
Operações Lógicas sobre Proposições	13
Operação Negação (\sim).....	13
Operação Conjunção (\wedge).....	14
Operação Disjunção (\vee)	15
Operação Condicional (\rightarrow)	17
Operação Bicondicional (\leftrightarrow)	18
Exercícios	19
Tautologia, Contradição e Contingência.....	19
Tautologia	19
Contradição	20
Contingência	20
Exercícios	20
Implicação Lógica ou Consequência Lógica (\Rightarrow)	21
Equivalência Lógica (\equiv)	21
Exercícios	22
Introdução a Algoritmos	23
Noções de Lógica.....	23
Algoritmizando a Lógica.....	24
Fases de um programa.....	25
Planejamento.....	25
Projeto.....	25
Escrita	25
Depuração	26
Manutenção.....	26
Programação Estruturada	26
Pseudo-linguagem	26
Algoritmo X Qualidade	26
Método para construção de algoritmos	27
Método Cartesiano	27
Planejamento Reverso	27
Tabelas de decisão.....	27
Exercícios	28

Conceitos Básicos	29
Tipos Primitivos	29
Exercícios:	29
Arquitetura de Memória	29
Memória	30
Armazenamento de dados na memória	30
Formação de Identificadores	30
Constantes	30
Variáveis	31
Exercício	31
Expressões	31
Expressões Aritméticas	31
Operadores Aritméticos	31
Funções Matemáticas	32
Prioridades	32
Expressões Lógicas	32
Operadores Relacionais	33
Operadores Lógicos	33
Comandos de Atribuição	33
Comandos de Entrada e Saída de Dados	34
Entrada de Dados	34
Saída de Dados	34
Blocos	34
Estruturas de Controle	36
Estrutura Sequencial	36
Exercícios	37
Estruturas de Seleção ou Decisão	37
Decisão Simples	37
Exercícios	38
Decisão Composta	38
Exercícios	38
Seleção Múltipla	38
Exercício	40
Estruturas de Repetição	40
Repita ... Até - Estrutura com teste no final	41
Exercício	41
Enquanto .. Faça - Estrutura com teste no Início	41
Exercício	42
Para .. Passo - Estrutura com variável de Controle	42
Exercício	42
Estrutura de Dados	45
Agregados Homogêneos	45
Variáveis Compostas Homogêneas	45
Declaração	45
Exercício	46
Agregados Multidimensionais	46
Declaração	46
Exercícios	48

	SUMÁRIO
Agregados Heterogêneos	48
Registros	48
Declaração	48
Conjunto de Registros	49
Exercício	50
Procedimentos e Funções	53
Procedimentos	53
Escopo de Variáveis	54
Parâmetros	54
Funções	55
Passagem de Parâmetros por Referência e Valor.....	55
Exercícios	56
Recursividade.....	57
Recursividade Direta	57
Recursividade Indireta.....	57
Arquivos.....	59
Definição.....	59
Organização dos Arquivos	59
Seqüencial.....	59
Direta ou Aleatória	59
Declaração	60
Manipulação de Arquivos.....	60
Abertura	60
Fechamento.....	60
Leitura.....	60
Gravação	61
Posicionamento e verificação de fim e início de arquivo	61
Avance (nome-arquivo);.....	61
FDA (nome-arquivo)	61
Posicione (nome-arquivo, chave);.....	61
Estudo de Concepções	62
Arquivo Direto Acessado Seqüencialmente	62
Arquivo Seqüencial Acessado Randomicamente: Arquivo Indexado	62
Exercícios:.....	62

CONTEÚDO PROGRAMÁTICO

- 1.Noções de Lógica
- 2.Conceito de Algoritmos
- 3.Componentes de Algoritmos
 - 3.1. Tipos de dados primitivos
 - 3.2. Arquitetura da memória
 - 3.3. Armazenamento de dados em memória
 - 3.4. Variáveis e constantes
 - 3.5. Expressões
 - 3.5.1. Expressões aritméticas
 - 3.5.2. Expressões lógicas
 - 3.5.3. Avaliação de expressões
 - 3.6. Instruções primitivas
 - 3.6.1. Atribuição
 - 3.6.2. Entrada de dados
 - 3.6.3. Saída de dados
 - 3.7. Estruturas de controle
 - 3.7.1. Estrutura Seqüencial
 - 3.7.2. Estrutura de Decisão
 - 3.7.3. Estrutura de Repetição
 - 3.7.4. Alinhamentos
 - 3.8. Estruturas de dados
 - 3.8.1. Variáveis Compostas Homogêneas
 - 3.8.2. Variáveis Compostas Heterogêneas
 - 3.9. Arquivos
- 4. Subalgoritmos
 - 4.1. Funções
 - 4.2. Procedimentos
 - 4.3. Passagem de Parâmetros
 - 4.4. Escopo de variáveis

□ Turbo Pascal

HISTÓRIA DA LÓGICA

Aristóteles e lógica silogística (base da argumentação legal). Sofre deficiências da linguagem natural

Os estóicos e a introdução dos conectivos.

1700 - Leibniz e lógica simbólica (proposicional e predicados)

1800 - Álgebra de Boole : leitura da lógica simbólica para valores de um domínio e operadores

Foco atual : lógica de primeira ordem, lógica de segunda ordem, resolução por padrões.

Dijkstra: “O objetivo da lógica é o raciocínio automático”

McCarty: é a principal ferramenta para a computação

INTRODUÇÃO A LÓGICA PROPOSICIONAL

Preliminares

Proposição

Frase é o elemento de comunicação que relaciona palavras entre si de modo a estabelecer uma mensagem com sentido completo.

As frases podem ser de vários tipos:

- **Declarativa:** “O Sol é uma estrela.”
- **Imperativa:** “Não faça isto!”
- **Interrogativa:** “Onde você mora?”
- **Exclamativa:** “Parabéns!”
-

A linguagem natural nem sempre é clara e precisa, sendo muito comum a ocorrência de ambigüidades que geram dúvidas sobre o significado do que se está falando.

Por isso, um dos objetivos da **lógica** é estabelecer uma **linguagem formal**, onde se pode expressar com **clareza**, **precisão** e emitir juízo de **verdadeiro** ou **falso** para determinadas frases.

Proposição é um conceito primitivo (aceito sem definição). Mas nada impede que se estabeleçam as suas características para melhor entendimento.

Proposição é uma frase **declarativa** (com sujeito e predicado), à qual pode ser atribuído, sem ambigüidade, um dos valores lógicos: **verdadeiro (V)** ou **falso (F)**.

Exemplos:

1) São proposições:

- a) O Japão fica na África.
- b) O Brasil é banhado pelo Oceano Atlântico.
- c) $3 + 4 = 7$
- d) $5 > 8$

2) Não São proposições:

- a) $3 + 4$ → Não tem predicado
- b) Onde você vai? → Sentença interrogativa.
- c) Os estudantes jogam vôlei. → O sujeito não está claramente especificado e a sentença não pode ser classificada em V ou F.

As proposições podem ser **simples** ou **compostas**.

Proposição simples é única, ou seja, não contem nenhuma outra proposição como parte integrante. Indicaremos tais proposições por **letras minúsculas** de nosso alfabeto.

Exemplos:

p: O México fica na América do Norte.

Q: O número 16 é quadrado perfeito.

Proposição composta ou fórmula é formada por duas ou mais proposições relacionadas pelo conectivos lógicos. Serão indicadas por **letras maiúsculas** de nosso alfabeto.

Notação:

P (p,q,r,...) indica que a proposição composta **P** é formada pelas proposições simples **p, q, r, ...**

As proposições que fazem parte de uma proposição composta podem ser, elas mesmas, proposições compostas.

Exemplos:

P: $1 + 2 = 3$ e $2 \neq 1$

Q: $1 + 2 = 3$ ou $2 \neq 1$

R: Se $1 + 2 = 3$, então $2 \neq 1$

Conectivos Lógicos

Conectivos lógicos (ou operadores lógicos) são palavras ou expressões usadas para formar novas proposições a partir de proposições.

Os conectivos lógicos são:

- não
- e
- ou
- se ..., então ...
- ... se, e somente se ...
-

Valor Lógico

O valor de uma proposição é chamado **valor lógico**. Os valores lógicos possíveis são: **verdade (V)** e **falsidade (F)**.

Notação:

V (p) indica o valor lógico da proposição **p**. Assim, se a proposição **p** for **verdadeira**, $V(p) = V$; se a proposição **p** for **falsa**, $V(p) = F$.

O valor lógico de uma proposição composta depende exclusivamente dos valores lógicos das suas proposições componentes e dos conectivos lógicos que as ligam.

Exemplos:

a) **p:** O Sol é verde. $\rightarrow V(p) = F$

b) **q:** Um hexágono tem seis lados. $\rightarrow V(q) = V$

c) r : 2 é raiz da equação $x^2 + 3x - 4 = 0 \rightarrow V(r) = F$

Princípios Fundamentais da Lógica

Princípio da Não-Contradição

Uma proposição não pode ser simultaneamente verdadeira e falsa.

Princípio do Terceiro Excluído

Toda proposição ou é só verdadeira ou é só falsa, nunca ocorrendo um terceiro caso.

Logo, toda proposição admite um e um só dos valores V ou F.

Tabela - Verdade

Tabela-verdade é uma maneira prática de dispor organizadamente os valores lógicos envolvidos em uma proposição composta.

Para a proposição simples p , temos:

Tabela-Verdade de p

p
V
F

Para proposições compostas, veremos que o número dos componentes simples determina o número de linhas das tabelas-verdade. A princípio, vamos construir as tabelas-verdade dispondo, apenas, **todas as possibilidades** de valores lógicos das proposições componentes; os possíveis valores lógicos das proposições compostas estudados mais adiante.

Proposição Composta por 2 Proposições Simples - $P(p,q)$

Tabela-Verdade

p	q
V	F

V	V
F	F
F	V

Proposição Composta por 3 Proposições Simples - $P(p,q,r)$

Tabela-Verdade

p	q	r
V	V	V
V	V	F
V	F	V
V	F	F
F	V	V
F	V	F
F	F	V
F	F	F

Teorema:

O **número de linhas** distintas de uma tabela-verdade é dado por 2^n , onde n é número de proposições simples componentes e 2 representa o número de valores lógicos possíveis (V ou F).

Operações Lógicas sobre Proposições

Operação Negação (\sim)

Se p é uma proposição, a **negação** da proposição p é denotada por $\sim p$ (lê-se **não p**).

- Se $V(p) = V$, então $V(\sim p) = F$
- Se $V(p) = F$, então $V(\sim p) = V$
-

Logo, a **negação** de uma proposição apresenta valor lógico **oposto** ao da proposição dada.

A **tabela-verdade** da operação **negação** é:

p	$\sim p$
V	F
F	V

Exemplos:

1) Dada a proposição:

p: O Sol é um planeta.

A sua negação é:

$\sim p$: O Sol não é um planeta.

2) Dada a proposição:

q: $2 + 3 = 5$

a sua negação é:

$\sim q$: $2 + 3 \neq 5$

3) Dada a proposição:

r: Rio de Janeiro é um país.

A sua negação é:

$\sim r$: Rio de Janeiro não é um país.

A negação pode, ainda, ser expressa de outras maneiras, como: $\sim r$: Não é verdade que Rio de Janeiro é um país.

$\sim r$: É falso que Rio de Janeiro é um país.

*Negar uma proposição **p** não é apenas afirmar algo diferente do que **p** afirma, ou algo com valor lógico diferente.*

Exemplo:

*A proposição “O Sol é uma estrela”, que é verdadeira, **não é negação** da proposição “O Sol é um planeta”, que é falsa.*

Operação Conjunção (\wedge)

Duas proposições **p** e **q** podem ser combinadas pelo conectivo **e** para formar uma proposição composta denominada **conjunção** das proposições originais.

Notação: $p \wedge q$ (lê-se: **p e q**)

Exemplos:

1) Dada as proposições:

p: Carlos estuda Matemática.

Q: Carlos joga xadrez.

A conjunção é:

$p \wedge q$: Carlos estuda Matemática e joga xadrez.

2) Dadas as proposições:

p : $2 > 0$

q : $2 \neq 1$

a conjunção é:

$p \wedge q$: $2 > 0$ e $2 \neq 1$

O símbolo \wedge pode ser usado, também, para definir a intersecção de dois conjuntos.

$$A \cap B = \{x / x \in A \wedge x \in B\}$$

A **conjunção** de duas proposições ($p \wedge q$) é **verdadeira** se, e somente se, **cada** componente for **verdadeiro**.

A **tabela-verdade** da operação **conjunção** é:

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

OU

p	\wedge	q
V	V	V
V	F	F
F	F	F
F	F	F

Operação Disjunção (V)

Duas proposições **p** e **q** podem ser combinadas pelo conectivo **ou** (com sentido de e/ou) para formar uma proposição composta denominada **disjunção** das proposições originais.

Notação: $p \vee q$ (lê-se: **p ou q**)

*Na linguagem natural, o conectivo **ou** pode traduzir tanto a idéia de hipóteses mutuamente exclusivas (ou ocorre isto ou ocorre aquilo) quanto a de que pelo menos uma das hipóteses ocorre.*

Exemplos:

1) A sentença “chove ou faz frio” é verdadeira nos seguintes casos: só chove;

• só faz frio;

•chove e faz frio.

2) O mesmo não acontece com a sentença “Pedro passará nos exames ou repetirá de ano”, que só é verdadeira nos seguintes casos:

•Pedro passará nos exames;

•Pedro repetirá de ano.

Mas é falsa a hipótese:

•Pedro passará nos exames e repetirá de ano.

No primeiro exemplo, a disjunção é **inclusiva** e, no segundo, a disjunção é **exclusiva**.

Em nosso estudo, vamos nos preocupar apenas com a **disjunção inclusiva**.

Exemplos:

1) Dadas as proposições:

p: João é estudante.

q: João é mecânico.

A disjunção é:

$p \vee q$: João é estudante **ou** mecânico.

2) Dadas as proposições:

p: 10 é número primo

q: 10 é número composto

a disjunção é:

$p \vee q$: 10 é número primo **ou** número composto.

O símbolo \vee pode ser usado, também, para definir a união de dois conjuntos: $A \cup B = \{x / x \in A \vee x \in B\}$

A **disjunção inclusiva** de duas proposições ($p \vee q$) é **falsa** se, e somente se, **todas** as componentes forem **falsas**.

A **tabela-verdade** da operação **disjunção** é:

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

Exemplos:

1) Determinar o valor lógico da proposição composta P dada a seguir: $P: 3 < \pi$ ou 2 não é número primo.

Resolução:

“ $3 < \pi$ ” é uma proposição verdadeira.

“2 não é número primo” é uma proposição falsa.

Com as proposições simples dadas estão ligadas pelo conectivo **ou**, então: $V(P) = V$

2) Sejam as proposições:

p: Maurício é jogador de vôlei.

q: Maurício é bonito.

Escrever em linguagem natural as seguintes proposições:

a) $p \wedge q$

b) $p \vee \sim q$

Resolução:

a) Maurício é jogador de vôlei e Maurício é bonito.

b) Maurício é jogador de vôlei ou Maurício não é bonito.

3) Construir a tabela-verdade para a proposição $p \vee \sim q$.

Resolução:

p	q	$\sim q$	$p \vee \sim q$
V	V	F	V
V	F	V	V
F	V	F	F
F	F	V	V

Operação Condicional (\rightarrow)

Duas proposições **p** e **q** podem ser combinadas pelo conectivo lógico “se ..., então ...” para formar uma nova proposição denominada **condicional**.

Notação: $p \rightarrow q$ (lê-se: se p, então q).

Outras maneiras de se ler o condicional $p \rightarrow q$:

- q, se p.
- p é condição suficiente para q.
- q é condição necessária para p.

Exemplo:

A proposição condicional “Se chove, então a rua fica molhada”, também pode ser lida das seguintes formas:

Chover é uma condição suficiente para a rua ficar molhada.

A rua ficar molhada é uma condição necessária quando chove.

No condicional $p \rightarrow q$, a proposição **p** é chamada **antecedente** e a proposição **q** é **consequente**.

Exemplos:

1) Dadas as proposições:

p: 1 litro = 1 dm³q: 1 ml = 1 cm³

a condicional é:

p → q: Se 1 l = 1 dm³, então 1 ml = 1 cm³.

2) Dadas as proposições:

p: Chove

q: Faz frio

a condicional é:

p → q: Se chove, então faz frio.

3) Dadas as proposições:

p: 5 < 2

q: 2 ∈ Z

a condicional é:

p → q: Se 5 < 2, então 2 ∈ Z.

Obs.: O símbolo Z representa o conjunto dos número inteiros.

A proposição condicional $p \rightarrow q$ só é **falsa** quando **p é verdadeira** e **q é falsa**. Caso isto não ocorra, a proposição condicional será verdadeira.

A **tabela-verdade** da operação **condicional** é:

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

Operação Bicondicional (↔)

Duas proposições **p** e **q** podem ser combinadas pelo conectivo lógico “... se, e somente se ...” para formar uma nova proposição denominada **bicondicional**.

Notação: $p \leftrightarrow q$ (lê-se: **p se, e somente se q**)Outras maneiras de se ler o bicondicional $p \leftrightarrow q$:

- p é condição necessária e suficiente para q.
- q é condição necessária e suficiente para p.

Exemplos:

1) Dadas as proposições:

p: Perereca se transforma em sapo.

Q: Sapo se transforma em príncipe.

A bicondicional é:

$p \leftrightarrow q$: Perereca se transforma em sapo se, e somente se, sapo se transforma em príncipe.

2) Dadas as proposições:

p: João é homem.

Q: João tem a voz grave.

A bicondicional é:

$p \leftrightarrow q$: João é homem se, e somente se, João tem a voz grave.

A proposição bicondicional $p \leftrightarrow q$ só é **verdadeira** quando $V(p) = V(q)$, caso contrário é falsa.

A **tabela-verdade** da operação **bicondicional** é:

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

Tabela verdade resumo dos conectivos

p	q	$p \wedge q$	$p \vee q$	$\sim p$	$p \rightarrow q$	$p \leftrightarrow q$
V	V	V	V	F	V	V
V	F	F	V	F	F	F
F	V	F	V	V	V	F
F	F	F	F	F	V	V

Prioridade dos conectivos

a) Parênteses

Usar parênteses em todas as fórmulas indicando assim a prioridade das subfórmulas.

$$(((\sim p) \vee q) \rightarrow ((r \wedge \sim q) \leftrightarrow p))$$

Quando o significado for claro podemos simplificar o uso de parênteses.

b) Tabela abaixo

$(\sim) > (\wedge) > (\vee) > (\rightarrow) > (\leftrightarrow)$

Exercícios

Dadas as proposições a seguir, monte a expressão e a tabela verdade correspondente:

1.a) Está chovendo

2.b) Está frio

2.a) João vai viajar

3.b) Maria vai viajar

4.c) Se Maria viajar ao lado de João, João ficará feliz

3.a) João vai viajar se Maria for

4.b) Maria vai viajar se Mario for

5.c) Mario vai viajar se Carlos ou Maria for

4. Dadas as expressões abaixo, monte as tabelas verdade

5.a) $p \wedge (q \rightarrow r) \vee (p \vee q)$

6.b) $(p \rightarrow q) \wedge (r \leftrightarrow q) \vee (p \wedge r)$

1.c) $((p \leftrightarrow \sim p) \wedge \sim(q \vee r)) \rightarrow (\sim q \wedge r) \vee s$

Tautologia, Contradição e Contingência

Tautologia

Uma proposição composta é uma **tautologia** quando o seu valor lógico é **sempre verdade (V)**, quaisquer que sejam os valores lógicos das proposições componentes.

Exemplos:

p: Chove

$\sim p$: Não chove

$(p \vee \sim p)$

A tabela-verdade é:

p	$\sim p$	$p \vee \sim p$
V	F	V
F	V	V

Logo, $(p \vee \sim p)$ é uma tautologia.

Contradição

Uma proposição composta é uma **contradição** quando o seu valor lógico é **sempre a falsidade (F)**, quaisquer que sejam os valores lógicos das proposições componentes.

Exemplo:

p: Chove

$\sim p$: Não chove

$(p \wedge \sim p)$

A tabela-verdade é:

p	~ p	p ∧ ~p
V	F	F
F	V	F

Logo, $(p \vee \sim p)$ é uma tautologia.**Contingência**

Uma proposição é contingente (ou uma **indeterminação**) quando não é uma tautologia e não é uma contradição.

Exercícios

Dadas as proposições abaixo, diga se ela é uma Tautologia, Contradição ou Contingência

1. $(p \wedge \sim p) \wedge p$
2. $(p \wedge \sim q) \wedge p$
3. $(p \wedge \sim p) \vee q$
4. $(q \wedge \sim p) \wedge r$
5. $p \wedge (q \rightarrow r) \vee (p \vee q)$

Implicação Lógica ou Consequência Lógica (\Rightarrow)

Dadas as proposições compostas **P** e **Q**, diz-se que ocorre uma **implicação lógica** (ou relação de implicação) entre **P** e **Q** quando a proposição condicional $P \rightarrow Q$ é uma **tautologia**.

OU

Uma fórmula **A** é consequência lógica de uma fórmula **B**, se e somente se para qualquer interpretação em que **B** é verdadeira **A** também é verdadeira.

Assim **B** implica logicamente em **A** se $B \rightarrow A$ é uma tautologia, e $B \wedge \sim A$ é uma contradição.

Notação: $P \Rightarrow Q$ (lê-se: “**P** implica **Q**”).

- Os símbolos \rightarrow e \Rightarrow têm significados diferentes:
- O símbolo \rightarrow realiza uma **operação** entre proposições, dando origem a uma nova proposição $p \rightarrow q$ cuja tabela-verdade pode conter tanto **V** quanto **F**.
- O símbolo \Rightarrow entre duas proposições dadas indica uma **relação**, isto é, que a proposição condicional associada é uma **tautologia**.

Exemplo:

Mostrar que $(p \wedge q) \Rightarrow p$.

p	q	$p \wedge q$	$(p \wedge q) \rightarrow p$
V	V	V	V
V	F	F	V
F	V	F	V
F	F	F	V

Como $(p \wedge q) \rightarrow p$ é uma tautologia, então $(p \wedge q) \Rightarrow p$, isto é, ocorre a implicação lógica.

Equivalência Lógica (\equiv)

Dadas as proposições compostas **P** e **Q**, diz-se que ocorre uma **equivalência lógica** entre **P** e **Q** quando suas tabelas-verdade forem **idênticas**.

OU

Uma fórmula **A** é equivalência lógica de uma fórmula **B** (**A** é logicamente equivalente a **B**) se **A** é implicação lógica de **B** e **B** é implicação lógica de **A**.

Assim **A** é equivalência lógica de **B** se $A \leftrightarrow B$ é uma tautologia.

Notação: $P \equiv Q$ ou $P \leftrightarrow Q$ (lê-se: “**P é equivalente a Q**”) Intuitivamente, proposições logicamente equivalentes transmitem a mesma informação, a mesma idéia, a partir das mesmas proposições componentes.

Exemplo:

Mostrar que $(p \rightarrow q) \wedge (q \rightarrow p)$ e $p \leftrightarrow q$ são equivalentes.

p	q	$p \rightarrow q$	$q \rightarrow p$	$(p \rightarrow q) \wedge (q \rightarrow p)$	$p \leftrightarrow q$
V	V	V	V	V	V
V	F	F	V	F	F
F	V	V	F	F	F
F	F	V	V	V	V

Logo, $(p \rightarrow q) \wedge (q \rightarrow p) \equiv p \leftrightarrow q$.

•O bicondicional não é uma operação lógica básica, mas a conjunção de proposições condicionais.

Exemplo:

Mostrar que $(p \wedge q) \equiv \sim(\sim p \vee \sim q)$.

Analisemos a tabela-verdade envolvendo as seguintes proposições:

A					B		~B
p	q	$p \wedge q$	$\sim p$	$\sim q$	$\sim p \vee \sim q$	$\sim(\sim p \vee \sim q)$	$A \leftrightarrow \sim B$

V	V	V	F	F	F	V	V
V	F	F	F	V	V	F	V
F	V	F	V	F	V	F	V
F	F	F	V	V	V	F	V

Como $(p \wedge q) \leftrightarrow \sim(\sim p \vee \sim q)$ é uma tautologia, então $(p \wedge q) \equiv \sim(\sim p \vee \sim q)$, isto é, ocorre a equivalência lógica.

Exercícios

Verifique se os pares de fórmulas abaixo são equivalências lógicas ou implicações lógicas.

- a) $(p \wedge q)$ e q
- b) $\sim(p \wedge q)$ e $(\sim p \vee \sim q)$
- c) $(\sim p \vee q)$ e $(p \rightarrow q)$

Introdução a Algoritmos

Noções de Lógica

O que é Lógica?

Lógica trata da correção do pensamento. Como filosofia, ela procura saber por que pensamos assim e não de outro jeito. Com arte ou técnica, ela nos ensina a usar corretamente as leis do pensamento.

Poderíamos dizer também que a Lógica é a arte de pensar corretamente e, visto que a forma mais complexa do pensamento é o raciocínio, a Lógica estuda ou tem em vista a **correção do raciocínio**. Podemos ainda dizer que a lógica tem em vista a **ordem da razão**. Isto dá a entender que a nossa razão pode funcionar desordenadamente, pode pôr as coisas de pernas para o ar. Por isso a Lógica ensina a colocar **Ordem no Pensamento**.

Exemplos:

a) Todo o mamífero é animal.

Todo cavalo é mamífero.

Portanto, todo cavalo é animal.

b) Todo mamífero bebe leite.

O homem bebe leite.

Portanto, todo homem é mamífero e animal.

Existe Lógica no dia-a-dia?

Sempre que pensamos, o raciocínio e a lógica nos acompanham necessariamente. Também quando falamos, pois a palavra falada é a representação do pensamento; e, visto que a palavra escrita é a representação da palavra falada, também pensamos quando escrevemos, utilizando a Lógica ou a Ilógica. Daí percebemos a importância da Lógica na nossa vida não só na teoria, como também na prática, já que quando queremos pensar, falar ou escrever corretamente precisamos colocar em **Ordem o Pensamento**, isto é, utilizar a Lógica.

Exemplos:

a) A gaveta está fechada.

A bala está na gaveta.

Preciso primeiro abrir a gaveta, para depois pegar a bala.

b) João é mais velho que José.

Marcelo é mais novo que José.

Portanto, João é mais velho que Marcelo.

Algoritmizando a Lógica

Construir algoritmos é o objetivo fundamental de toda a programação, mas, afinal,

O que é Algoritmo?

“Algoritmo é uma seqüência de passos que visam atingir um objetivo bem definido.”

(Ordem do Pensamento e, portanto, Lógica)

Apesar de achar este nome estranho, algoritmos são comuns em nosso cotidiano, como, por exemplo, uma receita de bolo. Nela está descrita uma série de ingredientes necessários, uma seqüência de diversos passos - ações - a serem cumpridos para que se consiga fazer determinado tipo de bolo - objetivo bem definido aguardado ansiosamente por todos. Para aprimorar nosso conceito de algoritmo, vamos tornar mais evidente alguns outros conceitos, como, por exemplo, o de ação:

“Ação é um acontecimento que a partir de um estado inicial, após um período de tempo finito, produz um estado final previsível e bem definido”, em que:

“Estado é a situação atual de dado objeto.”

Portanto, podemos redefinir Algoritmo como:

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

Em geral um algoritmo destina-se a resolver um problema: fixa um padrão de comportamento a ser seguido, uma norma de execução a ser trilhada, com o objetivo de alcançar a solução de um problema.

O que é padrão de comportamento?

Imagine a seguinte seqüência de números: 1, 6, 11, 16, 21, 26....

Para determinar o sétimo elemento da série, precisamos descobrir qual a sua regra de formatação, isto é, seu padrão de comportamento.

Para tal, observamos que a série obedece uma constância; visto que existe uma diferença constante entre cada elemento, a qual pode ser facilmente determinada, somos capazes de determinar o sétimo e qualquer outro termo.

Podemos, então, descrever uma atividade bem cotidiana, como, por exemplo, trocar uma lâmpada. Apesar de aparentemente óbvias demais, muitas vezes fazemos esse tipo de atividade inconscientemente. Sem percebermos seus pequenos detalhes. Vejamos se ela fosse descrita passo a passo:

- ⇒pegue a escada;
- ⇒posicione-a embaixo da lâmpada;
- ⇒busque uma lâmpada nova;
- ⇒suba na escada;
- ⇒retire a lâmpada velha;
- ⇒coloque a lâmpada nova.

Involuntariamente, já seguimos uma determinada seqüência de ações que, representadas neste algoritmo, fazem com que ele seja seguido naturalmente por pessoas, estabelecendo um padrão de comportamento.

É assim também com os algoritmos escritos para computador, você deve especificar todos os passos, para que o computador possa chegar ao objetivo.

Por exemplo:

Dados os números naturais(N)

0, 1, 2, 3, 4, 5, 6, ...

passo1 faça N igual a zero
passo2 some 1 a N
passo3 volte ao passo 2

Soma dos primeiros 100 números naturais:

passo1 faça N igual a zero
passo2 some 1 a N
passo3 se N for menor ou igual a 100
 então volte ao passo 2
 senão pare

Nos dois exemplos acima, o primeiro possui repertório bem definido mas não finito, enquanto que o segundo tem um critério de parada, ou seja, é finito e descreve um padrão de comportamento, ou seja, temos um algoritmo.

Fases de um programa

Como tudo na terra, o programa tem um tempo de vida, chamado de ciclo de vida de um programa.

Planejamento

É a fase onde definimos o problema a ser resolvido utilizando um computador. Nesta fase relacionamos a entrada e a saída do futuro programa, assim como a definição dos arquivos auxiliares que ele venha a utilizar.

Projeto

É a fase onde a resolução do problema é concebida. Neste ponto são definidos detalhes do algoritmo, estrutura de dados empregados pelo programa.

Escrita

Consiste em codificar o programa em uma linguagem de programação apropriada.

Depuração

Ao final da escrita estaremos com o programa quase pronto; mas será que ele funciona? Esta é a fase onde depuramos o programa, ou seja, corrigimos os erros.

Manutenção

Passada a fase de depuração, o programa será então liberado para utilização. Mas durante sua vida útil, um programa pode ser alterado; neste caso teremos que fazer novas mudanças, ou seja, manutenção.

Programação Estruturada

É um método de projeto que tem por objetivo gerar um produto (programa), que tem certas características desejáveis, tais como:

1. **correção** - os programas devem antes de mais nada, dar respostas certas para entradas certas.
- 2.
3. **complitude** - os programas devem dar respostas inteligíveis para entradas erradas.
- 4.
5. **flexibilidade** - os possíveis erros de programação devem ser fáceis de serem removidos, e as alterações devido a mudanças devem ser implementadas facilmente.
- 6.
7. **eficiência** - programas devem ser eficientes quanto aos recursos computacionais como economia de memória, tempo de processamento.
- 8.
9. **transparente** - programas devem ser fáceis de serem compreendidos.

Pseudo-linguagem

Escrever um algoritmo em português (portugol) visa principalmente facilitar o projetista, pensar no problema e não na máquina.

Algoritmo X Qualidade

Todo algoritmo deve ser feito de maneira lógica e racional, visando principalmente a sua eficiência e clareza.

Ao construir algoritmos devemos:

1. Saber que estes serão lidos por outras pessoas, além de nós mesmos, permitindo sua fácil correção.
2. Escrever comentários na sua elaboração. Algoritmos sem comentários é sinal de amadorismo, e um dos grandes erros que programadores cometem. Sua elaboração deve ser clara e resumida, limitando-se às ocasiões de maior detalhamento. Devem acrescentar alguma coisa, não apenas frasar.
3. Os comandos nos dizem o que está sendo feito, os comentários dizem o **porquê**.
4. Todo algoritmo deve possuir comentários no prólogo, explicando o que ele faz e dar instruções para seu uso.
5. Utilizar espaços e/ou linhas em branco para melhorar a legibilidade.
10. Nomes representativos para variáveis. **“Uma seleção adequada de nomes de variáveis é o**

princípio mais importante da legibilidade de algoritmos”.

11. Um comando por linha é suficiente.
12. Uso de parênteses aumenta a legibilidade e previne erros.
13. Utilize indentação, pois mostra a estrutura lógica do algoritmo. Deve ser feita segundo certos padrões estabelecidos.

Método para construção de algoritmos

A. Ler atentamente o enunciado.

É justamente o enunciado do exercício que fornece o encaminhamento necessário à resolução do problema, que se torna, portanto, dependente de sua completa compreensão.

B. Retirar do enunciado a relação das entradas de dados.

C. Retirar do enunciado a relação das saídas de dados.

D. Determinar o que deve ser feito para transformar as entradas determinadas nas saídas específicas.

Nesta fase é que determinamos a construção de algoritmos propriamente dito, pois, a partir de alguns requisitos especificados, devemos determinar qual a seqüência de ações é capaz de transformar um conjunto definido de dados nas informações de resultado. Para isso, podemos:

D.1. Utilizar o Método Cartesiano quando a complexidade (variedade) não estiver totalmente absorvida, conhecida.

Método Cartesiano

Nosso principal objetivo enquanto programadores é vencer a complexidade, o que mantém célebre a frase de Descartes “**Dividir para Conquistar**”. Este método consiste justamente em atacar o problema abrangente dividindo-o em partes menores, a fim de torná-lo mais simples ou específico e, se necessário, dividir novamente as partes não compreendidas.

Podemos esquematizar o seguinte procedimento (algoritmo) para o método:

- i. Dividir o problema em suas partes principais.
- ii. Analisar a divisão obtida para garantir coerência.
- iii. Se alguma parte não for bem compreendida, aplicar a ela o método.
- iv. Analisar o objeto para garantir entendimento e coerência.

D.2. Aplicar o Planejamento Reverso.

Planejamento Reverso

Processo utilizado que, a partir das saídas (informações de resultado), procura desagregar, desmontando a informação, a fim de atingir os dados de entrada, quando então teríamos (do fim para o início) todas as ações.

D.3. Montar uma tabela de decisão quando uma ou mais ações dependentes de um conjunto de condições assumirem determinadas combinações de valores.

Tabelas de decisão

Objetiva basicamente relacionar as ações que dependem de alguma condição com as próprias condições, a fim de esclarecer e visualizar facilmente quais valores o conjunto de condições deve

assumir para que se efetue sua respectiva ação.

E. Construir o algoritmo.

F. Executar o algoritmo.

Implica executar todas as ações descritas seguindo o fluxo de execução estabelecido, verificando se os resultados obtidos correspondem ao esperado quando da montagem do algoritmo, detectando então algum possível erro no desenvolvimento deste. Essa atividade é conhecida por “**teste de mesa**”.

Exercícios

1. Um homem precisa atravessar um rio com um barco que possui capacidade de carregar apenas ele mesmo e mais uma de suas três cargas, que são: um lobo, um bode e um maço de alfafa. O que o homem deve fazer para conseguir atravessar o rio sem perder suas cargas?
2. Suponha que você possua um robô e queira fazê-lo trocar uma lâmpada, sendo que o mesmo foi programado para obedecer aos seguintes comandos:
 - pegue <objeto>
 - pressione <objeto>
 - gire garras 180 no sentido horário
 - gire garras 180 no sentido anti-horário
 - mova <objeto> para <lugar>
 - desloque-se para <lugar>
e ainda é capaz de:
 - perceber quando algum comando não pode mais ser executado
 - sentir alguma fonte de calorQue ordens você daria para que seu robô trocasse a lâmpada?
3. Construa um algoritmo que mostre todos os passos que você segue para escolher o tipo de roupa com que vai sair, após levantar, levando em consideração apenas o tempo (bom, nublado, chuvoso) e a temperatura (quente, moderado, frio).
4. Elabore um algoritmo que mova três discos de uma Torre de Hanói, que consiste em três hastes (a - b - c), uma das quais serve de suporte para três discos diferentes (1 - 2 - 3), os menores sobre os maiores. Pode-se mover um disco de cada vez para qualquer haste, contanto que nunca seja colocado um disco maior sobre um menor. O objetivo é transferir os três discos para outra haste.
5. Três jesuítas e três canibais precisam atravessar um rio; para tal, dispõem de um barco com capacidade para duas pessoas. Por medidas de segurança não se permite que em alguma margem a quantidade de jesuítas seja inferior à de canibais. Qual a sequência de passos que permitiria a travessia com segurança?
6. Numa determinada noite, acontece uma queda de energia. Você sabia que poderia encontrar uma vela na gaveta da cozinha, um lampião embaixo da cama, fusíveis de reserva no armário da sala e fósforos na estante da cozinha. Descreva a sequência de passos que poderia ser utilizada para diagnosticar e resolver o problema, que pode ser previsto em duas possibilidades:
 - a) o fusível queimou;
 - b) a queda é na estação da companhia elétrica.

Conceitos Básicos

Em cada linguagem a frase de construção envolve dois aspectos:

- **a sintaxe** - forma como pode ser escrita;
- **a semântica** - conteúdo - a lógica de cada comando.

Tipos Primitivos

As informações manipuladas pelo computador se apresentam através dos dados (informações) e das instruções (comandos).

Os dados possuem diferentes tipos, que podem ser classificados em quatro tipos primitivos:

1. **Inteiro:** valores de -32,768 até +32,768
2. **Real:** valores com virgulas negativos ou positivos com grande abrangência.
3. **Caracter:** toda e qualquer informação composta de caracteres alfanuméricos (0..9, a..z, A..Z) e/ou caracteres especiais (!, @, #, \$, %, ^, &, *, etc). Estes caracteres (podendo ser um só ou uma cadeia) aparecerão sempre entre apóstrofes, ou aspas. Exemplo: "ASD", '2e3', "V".
4. **Lógico:** também conhecido como tipo Booleano, toda e qualquer informação que pode assumir somente dois valores (aberto/fechado, acesso/apagado), no nosso caso apenas pode-se usar **V** (verdadeiro) ou **F** (falso).

Exercícios:

1) Classifique os dados de acordo com o seu tipo, sendo (I = inteiro, R = real, C = caracter, L = lógico).

- | | | | |
|---------|-------------|--------------|-----------|
| () 0 | () + 36 | () "+3257 | () F |
| () 1 | () + 32 | () "+3257" | () 'F' |
| () 0,0 | () - 0,001 | () "-0,0" | () ".V." |
| () 0 | () + 0,05 | () ".V." | () F |
| () -1 | () + 3257 | () V | () -32 |
| () "a" | () "abc" | () -1,9E123 | () '0' |

Arquitetura de Memória

De um modo geral, a memória do computador pode ser vista como um conjunto de células, cada uma identificada unicamente por um número inteiro distinto, conhecido como **endereço**.

Memória

endereço	informações
1	"A"
2	10
:	:
N	.V.

O acesso às informações armazenadas na memória (leitura, escrita) é feita Byte a Byte. Cada Byte é formado por uma combinação de 8 dígitos binários (0 e 1), sendo assim, um Byte possui 256 (2^8) estados possíveis ou representa 256 símbolos de informação.

Armazenamento de dados na memória

Cada informação de determinado tipo é representada de diferentes formas, e necessita de uma certa quantidade de memória para armazená-las. O número de bytes necessários para armazenar uma informação dependerá do seu tipo. Por exemplo, na maioria das linguagens de programação o tipo inteiro é armazenado ocupando 2 bytes, ou seja 65536 ($2^8 \times 2^8 = 2^{16}$) possibilidades (-32767, -32766,, -1,0, 1..... 32767, 32768). Um dado do tipo caractere é formado por vários símbolos, sendo que cada um ocupa 1 byte. Desta forma o número de bytes utilizados no armazenamento do tipo caractere dependerá do seu tamanho. 1 endereço - 1 byte

Formação de Identificadores

Podemos imaginar a memória como sendo um armário repleto de gavetas, no qual as gavetas seriam os locais físicos responsáveis por armazenar objetos; os objetos (que podem ser substituídos ou não) seriam as informações (dados), e as gavetas as variáveis ou constantes.

Visto que na memória (armário) existem inúmeras variáveis (gavetas), precisamos diferenciá-las, o que é feito por meio de identificadores (etiquetas). Cada variável (gaveta), no entanto, pode guardar apenas uma informação (objeto) de cada vez, sendo sempre do mesmo tipo (material).

Os identificadores possuem certas regras para serem formados: Devem começar por um caracter alfabético;

5.Podem ser seguidos por mais caracteres alfabéticos e/ou alfanuméricos;

6.Não é permitido o uso de caracteres especiais ou espaços, com exceção do underscore(_).

Variáveis e Constantes

São entidades que armazenam valores. A diferença entre variáveis e constantes está na possibilidade de alteração do valor armazenado, durante todo o tempo de duração do programa.

Constantes

Entendemos que uma informação é constante quando não sofre nenhuma variação no decorrer do tempo.

A declaração de constantes deve ser feita da seguinte maneira:

Const

um_cm = 1

nome = 'João'

$\pi = 0,1415$

Variáveis

Entendemos que uma informação é variável quando sofre variações durante o decorrer do tempo.

A declaração de variáveis é feita da seguinte maneira:

Variáveis

dólar : real

endereço : caracter

existe : lógico

Exercício

Assinale os identificadores válidos:

- | | | | |
|----------------------|-----------|-----------------|------------|
| 1 - abc | 2 - AB/C | 3 - “João” | 4 - [x] |
| 5 - 123 ^a | 6 - 080 | 7 - 1 a 3 | 8 - (x) |
| 9 - #55 | 10 - AH! | 11 - Etc... | 12 - ...a |
| 13 - ΔBAC | 14 - xyz | 15 - Porta_mala | 16 - A_B-C |
| 17 - U2 | 18 - p{0} | 19 - A123 | 20 - A. |

Expressões

É um conjunto de constantes e/ou variáveis e/ou funções ligadas por operadores aritméticos ou lógicos.

Expressões Aritméticas

Denominamos expressão aritmética aquela cujos operadores são aritméticos e cujos operandos são constantes e/ou variáveis do tipo numérico (inteiro e/ou real).

Operadores Aritméticos

Chamamos de operadores aritméticos o conjunto de símbolos que representa as operações básicas da matemática:

Operadores Binários:

- ☞ + adição
- ☞ * multiplicação
- ☞ ** potenciação
- ☞ - subtração
- ☞ / divisão
- ☞ // radicação
- ☞ mod resto da divisão
- ☞ div quociente da divisão inteira

Operadores Unários:

✎+ manut. sinal
✎- manut. sinal

Exemplo:

$$9/4 = 2,25$$

$$9 \text{ div } 4 = 2$$

$$9 \text{ mod } 4 = 25$$

$$15 \text{ div } 7 = 2$$

$$15 \text{ mod } 7 = 1$$

Funções Matemáticas

Além das operações aritméticas básicas anteriormente citadas, podemos usar nas expressões aritméticas algumas funções da matemática:

✎ sen(x)	- seno de x;
✎ cos(x)	- coseno de x;
✎ tg(x)	- tangente de x;
✎ arctg(x)	- arco cuja tangente é x;
✎ arccos(x)	- arco cujo coseno é x;
✎ arcsen(x)	- arco cujo seno é x;
✎ abs(x)	- valor absoluto (módulo) de x;
✎ int(x)	- a parte inteira de um número fracionário;
✎ frac(x)	- a parte fracionária de x;
✎ ard(x)	- transforma por arredondamento, um número fracionário em inteiro;
✎ sinal(x)	- fornece o valor -1, +1 ou 0 conforme o valor de x seja negativo, positivo ou nulo;
✎ rnd(x)	- valor randômico de x;

Onde x pode ser um número, variável, expressão aritmética ou também outra função matemática.

Prioridades







A hierarquia das expressões e funções aritméticas é a seguinte:

✎**()** parênteses mais internos
✎**funções matemáticas**
✎******
✎**//**
✎*****
✎**/**
✎**div**
✎**mod**
✎**+**
✎**-**

Expressões Lógicas

Denominamos expressão lógica aquela cujos operadores são lógicos e/ou relacionais e cujos

Utilizamos os operadores relacionais para realizar comparações entre dois valores de mesmo tipo primitivo. Tais valores são representados por constantes, variáveis ou expressões aritméticas.

	=	igual a
	>	maior que
	<	menor que
	< >	diferente de
	> =	maior ou igual a
	< =	menor ou igual a

Exemplos:

V F

F F

Utilizaremos três conectivos básicos para a formação de novas proposições a partir de outras já conhecidas. Os operadores lógicos são:

Exemplo:

V

Comandos de Atribuição

Um comando de atribuição permite-nos fornecer um valor a uma certa variável, onde o tipo dessa informação deve ser compatível com o tipo da variável, isto é, somente podemos atribuir um valor lógico a uma variável capaz de comportá-lo, ou seja, uma variável declarada do tipo lógico.

O comando de atribuição é uma seta apontando para a variável ou dois pontos e o sinal de igual (:=):

Exemplo:

variáveis

A, B : lógico;

X : inteiro;

A := V;

X := 8 + 13 div 5;

B := 5 = 3;

Comandos de Entrada e Saída de Dados

Entrada de Dados

Para que nossos algoritmos funcionem, em quase todos os casos precisaremos de informações que serão fornecidas somente após o algoritmo pronto, e que sempre estarão mudando de valores, para que nossos algoritmos recebem estas informações, devemos então construir entradas de dados, pelas quais o usuário (pessoa que utilizar o programa) poderá fornecer todos os dados necessários.

A sintaxe do comando de entrada de dados é a seguinte:

leia (variável);

leia ("Entre com o valor de var1 e var 2: ", var1, var2);

Saída de Dados

Da mesma forma que nosso algoritmo precisa de informações, o usuário precisa de respostas as suas perguntas, para darmos estas respostas usamos um comando de saída de dados para informar a resposta.

A sintaxe do comando de saída de dados é a seguinte:

escreva (variável);

escreva ('cadeia de caracteres');

escreva ('cadeia', variável);

escreva (número, 'cadeia', variável);

Blocos

Delimitam um conjunto de ações com uma função definida; neste caso, um algoritmo pode ser definido como um bloco.

Exemplo:

início início do algoritmo

seqüência de ações

fim. Fim do bloco (algoritmo)

Estruturas de Controle

Estrutura Sequencial

É o conjunto de ações primitivas que serão executadas numa sequência linear de cima para baixo e da esquerda para direita, isto é, na mesma ordem em que foram escritas. Como podemos perceber, todas as ações devem ser seguidas por um ponto-e-vírgula (;), que objetiva separar uma ação de outra.

Variáveis;

(declaração de variáveis);

início

comando 1;

comando 2;

comando 3;

fim.

Exemplo:

Construa um algoritmo que calcule a média aritmética entre quatro notas quaisquer fornecidas pelo usuário.

Resolvendo através do Método de Construção de Algoritmos, temos:

- 1.Dados de entrada: quatro notas bimestrais (N1, N2, N3, N4);
- 2.Dados de saída: média aritmética anual;
- 3.O que devemos fazer para transformar quatro notas bimestrais em uma média anual?
- 4.Resposta: utilizar média aritmética.
- 5.O que é média aritmética?
- 6.Resposta: a soma dos elementos divididos pela quantidade deles. Em nosso caso particular: $(N1 + N2 + N3 + N4)/4$;
- 7.Construindo o algoritmo:

variáveis (declaração das variáveis)

N1, N2, N3, N4, MA : **real**;

início (começo do algoritmo)

conheça (N1, N2, N3, N4); {entrada de dados}

Ma := $(N1 + N2 + N3 + N4) / 4$; {processamento}

escreva ('A média anual e: ', MA); {saída de dados}

fim.

Exercícios

- 1.Dados dois números inteiros, achar a média aritmética entre eles.
- 2.Dados dois números inteiros, trocar o conteúdo desses números.
- 3.Dados três notas inteiras e seus pesos, encontrar a média ponderada entre elas.
- 4.Calcular a área de um triângulo reto.
- 5.Escriva um algoritmo que tenha como entrada nome, endereço, sexo, salário. Informe-os.
- 6.Escriva um algoritmo que calcule: $C = (A + B) * B$.
- 7.Faça um algoritmo que calcule o valor a ser pago em uma residência, que consumiu uma quantidade X de energia elétrica.
- 8.Faça um algoritmo para calcular e imprimir a tabuada.
- 9.Fazer a transformação de um valor em dólar, para a moeda corrente do Brasil.

Estruturas de Seleção ou Decisão

Uma estrutura de decisão permite a escolha de um grupo de ações e estruturas a ser executado quando determinadas condições, representadas por expressões lógicas, são ou não satisfeitas.

Decisão Simples

Se <condição> **então**

início {bloco verdade}

 C; {comando único (ação primitiva)}

fim {bloco}

<condição> é uma expressão lógica, que, quando inspecionada, pode gerar um resultado falso ou verdadeiro.

Se V, a ação primitiva sob a cláusula será executada; caso contrário, encerra o comando, neste caso, sem executar nenhum comando.

Exemplo

variáveis

altura1, altura2 : **real**

início

leia (altura1, altura2);

se altura1 > altura2 **então**

início

escreva(‘Altura 1 maior’);

fim {se}

fim. {algoritmo}

Exercícios

1. Faça um algoritmo que conheça as quatro notas bimestrais e, informe a média do aluno e se ele passou; média para aprovação = 6.
2. Conheça três números inteiros, e informe qual é o maior.
3. Encontrar o dobro de um número se este for par, se ímpar encontrar o triplo.

Decisão Composta

Se <condição> então

início

C;

B;

fim;

senão

início

A;

fim;

Notamos agora que se a condição for satisfeita (Verdadeira), os comandos C e B serão executados, mas se a condição for falsa, também podem ser executados comandos, neste caso o comando A entrando no senão.

Exercícios

1. Construa um algoritmo que verifique a validade de uma senha fornecida pelo usuário. A senha válida deve ser igual a 'LÓGICA'.
2. Conhecendo-se três números, encontrar o maior.
3. Dados três números inteiros, colocá-los em ordem crescente.
- 4.

Seleção Múltipla

Escolha variável

caso valor1 : comando1;

caso valor2 : comando2;

caso valor3 : comando3;

caso valor4 : comando4;

caso contrário comando_F

fimescolha;

Esta estrutura evita que façamos muitos blocos **se**, quando o teste será sempre em cima da mesma variável.

Exemplo:

```
se X = V1 então início
    C1;
fim
senão início
    se X = V2 então início
        C2;
    fim;
    senão início
        se X = V3 então início
            C2;
        fim
        senão início
            se X = V4 então início
                C3;
            fim
            senão início
                C4;
            fim;
        fim;
    fim;
fim;
```

Com a estrutura de escolha múltipla, o algoritmo ficaria da seguinte maneira:

```
escolha X
    caso V1 : C1;
    caso V2 : C2;
    caso V3 : C2;
    caso V4 : C3;
    senão C4;
fimescolha;
```


Exercício

1. Numa festinha de fim de curso, foi feito um sorteio para distribuir o dinheiro restante em caixa. Dez pessoas foram sorteadas com direito a tentar a sorte mais uma vez, da seguinte forma: Deveriam apanhar uma bola numerada de 0 a 9 e conforme o algarismo sorteado o prêmio seria:

Número da Bola	Prêmio (% do valor do caixa)
0	05
1	25
2	10
3	07
4	08
5	05
6	15
7	12
8	03
9	10

Faça um algoritmo que calcule o prêmio recebido individualmente por pessoa.

2. Sendo dados 3 números positivos, verificar a natureza do triângulo formado, quanto aos seus ângulos, com estes números como medida dos lados.
3. Considerando três notas inteiras, encontrar a média aritmética simples entre as que correspondem a números pares.
4. Dados 4 números, colocá-los em ordem crescente.
5. Conhecer o nome e a idade de três pessoas, informar quem é o mais velho e quem é o mais novo.
6. Dadas duas medidas, informar a figura geométrica que se forma, sendo que cada medida corresponde ao tamanho de dois lados paralelos.
7. Dada a hora (apenas a hora, sem minutos ou segundos), informar qual a direção do sol.
8. Sendo conhecidos os valores de Z e W encontrar:

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

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

se $w > 0$ ou $z < 7$

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

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

caso contrário

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

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

Estruturas de Repetição

Estas estruturas possibilitam que nosso algoritmo seja muito mais enxuto e fácil de se programar.

Imagine um algoritmo de fatorial de 8:

variáveis

fat : real;

início

fat := 8 * 7;

fat := fat * 6 ;

fat := fat * 5;

fat := fat * 4;

fat := fat * 3;

fat := fat * 2;

escreva(fat);

fim.

O resultado será o fatorial com certeza mas, imagine se fosse o fatorial de 250. Ou ainda, o usuário deseja fornecer o número e o algoritmo deve retornar o fatorial, qual número será digitado? Quantas linhas deverão ser escritas?

Para isso servem as estruturas de repetição, elas permitem que um determinado bloco de comandos seja repetido várias vezes, até que uma condição determinada seja satisfeita.

Repita ... Até - Estrutura com teste no final

Esta estrutura faz seu teste de parada após o bloco de comandos, isto é, o bloco de comandos será repetido, até que a condição seja V. Os comandos de uma estrutura repita .. até sempre serão executados pelo menos uma vez.

Repita

comando1;

comando2;

comando3;

até <condição>;

Exercício

1.Construa o algoritmo de cálculo do fatorial com a estrutura Repita .. Até.

Enquanto .. Faça - Estrutura com teste no Início

Esta estrutura faz seu teste de parada antes do bloco de comandos, isto é, o bloco de comandos será repetido, até que a condição seja F. Os comandos de uma estrutura **enquanto .. faça** poderão ser executados uma vez, várias vezes ou nenhuma vez.

Enquanto < condição > **Faça**

início

< bloco de comandos >;

fim;

Exercício

1. Construa o algoritmo de cálculo do fatorial com a estrutura Enquanto .. Faça.

Para .. Passo - Estrutura com variável de Controle

Nas estruturas de repetição vistas até agora, ocorrem casos em que se torna difícil determinar quantas vezes o bloco será executado. Sabemos que ele será executado enquanto uma condição for satisfeita - **enquanto..faça**, ou até que uma condição seja satisfeita - **repita...até**. A estrutura **para .. passo** repete a execução do bloco um número definido de vezes, pois ela possui limites fixos:

Para <variável> := <valor> **até** <valor> **passo N faça**

início

< bloco de comandos >;

fim;

Exercício

1. Construa o algoritmo de cálculo do fatorial com a estrutura Para .. Passo.
2. Analise o desempenho dos três algoritmos de fatorial feitos e responda qual a melhor estrutura a se usar.
3. Dado um conjunto de números inteiros, obter a soma e a quantidade de elementos.
4. Encontrar os N primeiros termos de uma progressão geométrica, onde o primeiro termo e a razão são conhecidos.
5. Idem ao exercício 1, calculando também a soma dos números negativos, positivos, pares e ímpares.
6. Determinar o menor entre um conjunto de números inteiros fornecidos um de cada vez.
7. A conversão de graus Fahrenheit para centígrados é obtida pela fórmula $C = 5/9 (F-32)$. Escreva um algoritmo que calcule e escreva uma tabela de graus centígrados em função de graus Fahrenheit que variem de 50 a 150 de 1 em 1.
8. Uma rainha requisitou os serviços de um monge e disse-lhe que pagaria qualquer preço. O monge, necessitando de alimentos, indagou à rainha sobre o pagamento, se poderia ser feito com grãos de trigo dispostos em um tabuleiro de xadrez, de tal forma que o primeiro quadro deveria conter apenas um grão e os quadros subsequentes, o dobro do quadro anterior. A rainha achou o trabalho barato e pediu que o serviço fosse executado, sem se dar conta de que seria impossível efetuar o pagamento. Faça um algoritmo para calcular o número de grão que o monge esperava receber.
9. Prepare um algoritmo que calcule o valor de H, sendo que ele é determinado pela série
10. $H = 1/1 + 3/2 + 5/3 + 7/4 + \dots + 99/50$.
11. Elabore um algoritmo que determine o valor de S, onde:
12. $S = 1/1 - 2/4 + 3/9 - 4/16 + 5/25 - 6/36 \dots - 10/100$
13. Escreva um algoritmo que calcule e escreva a soma dos dez primeiros termos da seguinte série:

$$2/500 - 5/450 + 2/400 - 5/350 + \dots$$

14. Construa um algoritmo que calcule o valor aproximado de PI utilizando a fórmula
15. $PI = 3/(H*32)$, onde: $H = 1/1**3 - 1/3**3 + 1/5**3 - 1/7**3 + 1/9**3 - \dots$
16. Fulano tem 1,50 metro e cresce 2 cm pôr ano, enquanto Ciclano tem 1,10 metro e cresce 3 cm por ano. Construa um algoritmo que calcule e imprima quantos anos serão para que Ciclano seja maior que Fulano.
17. Em uma eleição presidencial, existem quatro candidatos. Os votos são informados através de código. Os dados utilizados para a escrutinagem obedecem à seguinte codificação:
- 1, 2, 3, 4 = voto para os respectivos candidatos;
 - 5 = voto nulo;
 - 6 = voto em branco;
- Elabore um algoritmo que calcule e escreva: total de votos para cada candidato;
- total de votos nulos;
 - total de votos em branco;
 - percentual dos votos em branco e nulos sobre o total;
 - situação do candidato vencedor sobre os outros três, no caso, se ele obteve ou não mais votos que os outros somados;

Como finalizador do conjunto de votos, tem-se o valor 0.

13. Calcule o imposto de renda de um grupo de contribuintes considerando que os dados de cada contribuinte, número do CPF, número de dependentes e renda mensal são valores fornecidos pelo usuário. Para cada contribuinte será feito um desconto de 5% de salário mínimo por dependente.

Os valores da alíquota para cálculo do imposto são:

Renda Líquida	Alíquota
até 2 salários mínimos	isento
2..3 salários mínimos	5%
3..5 salários mínimos	10%
5..7 salários mínimos	15%
acima de 7 salários mínimos	20%

O último valor, que não será considerado, terá o CPF igual a zero. Deve ser fornecido o valor atual do salário mínimo.

14. Realizou-se uma pesquisa para determinar o índice de mortalidade infantil em um certo período. Construa um algoritmo que leia o número de crianças nascidas no período e, depois, num número indeterminado de vezes, o sexo de uma criança morta (masculino, feminino) e o número de meses da vida da criança.

Como finalizador, teremos a palavra “fim” no lugar do sexo da criança.

Determine e imprima:

- a porcentagem de crianças mortas no período;
- a porcentagem de crianças do sexo masculino mortas no período;
- a porcentagem de crianças que viveram dois anos ou menos no período.

15. Suponha que exista um prédio sem limites de andares, ou seja, um prédio infinito, onde existam três elevadores, denominados A, B e C. Para otimizar o sistema de controle dos elevadores, foi realizado um levantamento no qual cada usuário respondia:

- o elevador que utilizava com maior frequência;
- o andar ao qual se dirigia;
- o período que utilizava o elevador, entre:
- M = matutino;
- V = vespertino;
- N = noturno.

Construa um algoritmo que calcule e imprima:

- qual é o andar mais alto a ser utilizado;
- qual é o elevador mais freqüentado e em que horário se encontra seu maior fluxo;
- qual o horário mais usado de todos e a que elevador pertence;
- qual a diferença percentual entre o mais usado dos horários e o menos usado (especificando qual o menos usado);
- qual a percentagem sobre o total de serviços prestados do elevador de média utilização.

Estrutura de Dados

Mesmo possuindo tipos básicos para as declarações de variáveis, você deve ter notado que eles muitas vezes não são suficientes para representar toda e qualquer informação que possa surgir. Portanto, em muitas situações esses recursos são escassos, o que poderia ser suprido se existissem mais tipos de informação ou, ainda melhor, se esses tipos pudessem ser “**construídos**”, criados, à medida que se fizessem necessários.

Agregados Homogêneos

Da mesma forma que na Teoria dos Conjuntos, uma variável pode ser interpretada como um elemento e uma estrutura de dados como um conjunto. As variáveis compostas homogêneas são formadas de variáveis com o mesmo tipo primitivo.

Os agregados homogêneos correspondem a posições de memória, identificadas por um mesmo nome, referenciadas por um índice, e que possuem um mesmo tipo.

Variáveis Compostas Homogêneas

Suponha que você está dentro de um elevador, para ir a um determinado andar você deve especificar o seu número, isto é, em um vetor, você deve se referenciar por seu índice para acessar qualquer um de seus elementos.

Declaração

Tipo

IDENTIFICADOR = **vetor** [LI .. LF] **de** <tipo>;

variáveis

variável : **IDENTIFICADOR**;

Onde:

LI - representa o limite inicial do vetor; LF - representa o limite final do vetor; <tipo> - qualquer um dos tipos primitivos, ou ainda um outro tipo que pode ser construído, o qual veremos adiante.

Primeiramente criaremos um novo tipo e lhe daremos um nome, após isso podemos usá-lo para declarar as variáveis que serão utilizados dentro do programa.

Exemplo:

Utilizaremos o exercício de média das notas:

tipo

v_notas = **vetor** [1..4] de real;

variáveis

notas : v_notas;

índice: **inteiro**;

média: **real**;

início

para índice \leftarrow 1 **até** 4 **faça**

leia(notas[índice]);

fim;

média \leftarrow 0;

para índice \leftarrow 1 **até** 4 **faça**

 média \leftarrow média + notas[índice]; média \leftarrow média / 4;

escreva(média);

fim;

fim.

Exercício

1. Elabore um algoritmo que leia, some e imprima o resultado da soma, entre dois vetores inteiros de 50 posições.
2. Construa um algoritmo que preencha um vetor de 100 elementos inteiros colocando 1 na posição correspondente a um quadrado perfeito e 0 nos demais casos.

Agregados Multidimensionais

Suponha que, além do acesso pelo elevador até um determinado andar, tenhamos também a divisão desse andar em apartamentos. Para chegar a algum deles não basta só o número do andar, precisamos também do número do apartamento.

Os vetores têm como principal característica a necessidade de apenas um índice para endereçamento - estruturas unidimensionais. Uma estrutura que precisasse de mais de um índice, como no caso específico, dividido em apartamentos, seria então denominada estrutura composta multidimensional (agregado), neste caso, de duas dimensões (bidimensional).

Declaração

Tipo

IDENTIFICADOR = **matriz** [LI1 .. LF1, LI2 .. LF2, ..., LIN .. LFN] **de** <tipo>;

variáveis

variável : **IDENTIFICADOR**;

Onde:

LIN, LFN - são os limites dos intervalos de variação dos índices da variável, onde cada par de limites está associado a um índice. O número de dimensões é igual ao número de intervalos;

<tipo> - qualquer um dos tipos primitivos, ou ainda um outro tipo que pode ser construído, o qual veremos adiante.

Primeiramente criaremos um novo tipo e lhe daremos um nome, após isso podemos usá-lo para declarar as variáveis que serão utilizados dentro do programa.

Exemplo:

Construa um algoritmo que efetue a leitura, a soma e a impressão do resultado, entre duas matrizes inteiras que comportem 25 elementos;

tipo

m = **matriz** [1 .. 5, 1 .. 5] **de** **inteiros**;

variáveis

ma, mb, mc : **m**;

i,j : **inteiro**;

início

i \leftarrow 1;

enquanto i <= 5 **faça**

início

j \leftarrow 1;

enquanto j <= 5 **faça**

início

conheça (ma[i,j] , mb[i,j]); mc[i,j] \leftarrow ma[i,j] + mb[i,j]; j \leftarrow j + 1;

fim;

i \leftarrow i + 1;

fim;

j \leftarrow 1;

enquanto j <= 5 **faça**

início

i \leftarrow 1;

enquanto i <= 5 **faça**

início

informe (mc[i,j]);

i \leftarrow i + 1;

fim;

$j \leftarrow j + 1;$

fim;

fim.

Exercícios

1. O tempo que um determinado avião leva para percorrer o trecho entre duas localidades distintas está disponível através da seguinte tabela:

	A	B	C	D	E	F	G
A	XX	02	11	06	15	11	01
B	02	XX	07	12	04	02	15
C	11	07	XX	11	08	03	13
D	06	12	11	XX	10	02	01
E	15	04	08	10	XX	05	13
F	11	02	03	02	05	XX	14
G	01	15	13	01	13	14	XX

- Construa um algoritmo que leia a tabela acima e informe ao usuário o tempo necessário para percorrer duas cidades por ele fornecidas, até o momento em que ele fornecer duas cidades iguais (fonte e destino).
- Elabore um algoritmo que imprima a tabela sem repetições (apenas o triângulo superior ou o triângulo inferior).
- Desenvolva um algoritmo que permita ao usuário informar várias cidades, até inserir uma cidade "0", e que imprima o tempo total para cumprir todo o percurso especificado entre as cidades fornecidas.
- Escreva um algoritmo que auxilie um usuário a escolher um roteiro de férias, sendo que o usuário fornece quatro cidades: a primeira é sua origem, a última é seu destino obrigatório, e as outras duas caracterizam a cidade de descanso (no meio da viagem). Por isso, o algoritmo deve fornecer ao usuário qual das duas viagens (origem para descanso, descanso para destino) seja o mais próximo possível.

Agregados Heterogêneos

Já sabemos que um conjunto homogêneo de dados é composto por variáveis do mesmo tipo primitivo; porém, se tivéssemos um conjunto em que os elementos não são do mesmo tipo, teríamos então um conjunto heterogêneo de dados.

Registros

Uma das principais estruturas de dados é o registro. Para exemplificar, imagine uma passagem de ônibus, que é formada por um conjunto de informações logicamente relacionadas, porém de tipos diferentes, tais como: número da passagem (inteiro), origem e destino (caracter), data (caracter), horário (caracter), poltrona (inteiro), distância (real), fumante (lógico), que são subdivisões (elementos de conjunto) do registro, também chamadas de campos. Logo, um registro é composto por campos que são partes que especificam cada uma das informações.

Número: _____ Fumante: _____
 De: _____ Para: _____
 Data: ____ / ____ / ____ Horário: ____ : ____
 Poltrona: _____ Distância: _____

Declaração**Tipo IDENTIFICADOR = registro**

campo1 : tipo;

campo2 : tipo;

" "

campoN : tipo;

fimregistro;**variáveis**variável : **IDENTIFICADOR;**

Onde:

IDENTIFICADOR : representa o nome associado ao tipo registro construído.

Campo1, campo2, campoN: são os nomes associados a cada campo do registro.

Da mesma forma que na declaração de vetores e matrizes, primeiramente devemos criar um tipo, para então declararmos as variáveis desse tipo.

Exemplo:

Conhecer um passageiro e imprimir.

Tipo**RegPassagem = registro**número, poltrona : **inteiro;**origem, destino, data, horário : **caracter;** distância : **real;**fumante : **lógico;****fimregistro;****variáveis**passagem : **RegPassagem;****início****leia**(passagem.número, passagem.poltrona);**leia**(passagem.origem, passagem.destino);**leia**(passagem.data, passagem.horário);**leia**(passagem.distância, passagem.fumante);**escreva**(passagem.número, passagem.poltrona);**escreva**(passagem.origem, passagem.destino);**escreva**(passagem.data, passagem.horário);**escreva**(passagem.distância, passagem.fumante);**fim.**

Conjunto de Registros

Nas estruturas compostas heterogêneas (vetores e matrizes) utilizamos tipos de dados primitivos como sendo os elementos dessas estruturas. Agora utilizaremos como componente dessa estrutura não apenas um tipo primitivo, mas sim os tipos construídos, neste caso os registros. Supondo que quiséssemos manter um registro de informações relativas a passagens rodoviárias de todos lugares de um ônibus, utilizaríamos um registro diferente para cada poltrona, e para agrupar todos eles utilizaríamos um conjunto desses registros.

Como possuímos 44 lugares num ônibus, numerados sequencialmente de 1 até 44, podemos, para uni-los, criar um vetor no qual cada posição é um elemento de tipo construído registro (passagem).

Como possuímos um vetor composto por registros, não podemos declara esse vetor sem antes ter declarado seus elementos; devemos então declarar primeiro o tipo construído registro e depois o vetor.

Tipo

```
RegPassagem = registro  
    número, poltrona : inteiro;  
    origem, destino, data, horário : caracter; distância : real;  
    fumante : lógico;  
fimregistro;  
VetPassagem = vetor [1 .. 44] de RegPassagem;
```

variáveis

```
passagem : RegPassagem;  
vetor : VetPassagem;
```

início

```
    para i <— 1 até 44 faça início  
        conheça (vetor[i].número);  
        conheça (vetor[i].poltrona);  
        conheça (vetor[i].origem);  
        conheça (vetor[i].destino);  
        conheça (vetor[i].data);  
        conheça (vetor[i].horário);  
        conheça (vetor[i].distância);  
        conheça (vetor[i].fumante);  
    fim;  
fim.
```

Exercício

1. Defina um registro para cheque bancário.
2. Construa um algoritmo que, usando um campo saldo de um registro conta bancária, imprima um relatório dos cheques sem fundos.
3. Uma determinada biblioteca possui obras de ciências exatas, ciências humanas e ciências biomédicas, totalizando 1.500 volumes, sendo 500 volumes de cada área. O proprietário resolveu informatizá-la e para tal agrupou as informações sobre cada livro do seguinte modo:

Código de Catalogação: _____ Doador: _____

Nome da Obra: _____

Nome do Autor: _____

Editora: _____ Nº de páginas: _____

- a) Construir um algoritmo que declare tal estrutura e que reúna todas as informações de todas as obras em três volumes distintos para cada área.
 - b) Elabore um trecho de algoritmo que, utilizando como premissa o que foi feito no item a, faça uma consulta às informações. O usuário fornecerá código da obra e sua área: existindo tal livro, informa seus campos; do contrário; envia mensagem de aviso. A consulta se repete até que o usuário introduza como código finalizador o número -1.
 - c) Idem ao item b, porém o usuário só informa o nome e a área do livro que deseja consultar.
 - d) Escreva um trecho de algoritmo que liste todas as obras de cada área que representam livros doados.
 - e) Idem ao item d, porém, obras que sejam livros comprados e cujo número de páginas se encontram entre 100 e 300.
 - f) Elabore um trecho de algoritmo que faça a alteração de um registro; para tal, o usuário fornece o código, a área e as demais informações sobre o livro. Lembre-se de que somente pode ser alterado um livro que exista.
 - g) Construa um trecho de algoritmo que efetue a exclusão de algum livro; o usuário fornecerá o código e a área.
-
2. Dado um conjunto de tamanho N, calcular a somatória de seus elementos.
 3. Calcular a média de um conjunto de números inteiros.
 4. Encontrar o maior elemento, e sua respectiva posição, de um vetor A de 15 elementos.
 5. Dados dois conjuntos ordenados A e B de N posições, gerar a partir destes o conjunto C, sabendo-se que o conjunto C deverá conter os elementos comuns entre A e B.
 6. Dado um conjunto A de N elementos, separá-lo em outros dois conjuntos B e C, contendo B os elementos pares e C os elementos ímpares.
 7. Dado um conjunto A de N elementos, separar os elementos pares e ímpares, usando apenas um vetor extra.
 8. Fazer um algoritmo para corrigir provas de múltipla escolha. Cada prova tem 10 questões e cada questão vale 1 ponto. O primeiro conjunto de dados a ser lido será o gabarito para a correção da prova. Os outros dados serão os números dos alunos e suas respectivas respostas, e o último número, do aluno fictício, será 9999.
 9. O programa deverá calcular e imprimir:
 - 10.a) para cada aluno, o seu número e nota;
 - 11.b) a porcentagem de aprovação, sabendo-se que a nota mínima de aprovação é 06;

- 12.c) a nota que teve maior frequência absoluta, ou seja, a nota que apareceu maior número de vezes.
13. Leia uma frase de 80 letras (incluindo brancos).
14. Escrever um algoritmo para:
- 15.a) contar quantos brancos existem na frase;
- 16.b) contar quantas vezes aparece a letra 'A';
- 17.c) contar quantas vezes ocorre o mesmo par de letras na frase e quais são elas.
18. Para o controle dos veículos que circulam em uma determinada cidade, a Secretária dos Transportes criou o seguinte registro padrão:

Onde:

Proprietário: _____ Combustível: _____

Modelo: _____ Cor: _____

Nº chassi: _____ Ano: _____ Placa: _____

- Combustível pode ser álcool, diesel ou gasolina;
 - Placa possui os três primeiros valores alfabéticos e os quatro restantes numéricos.
- Sabendo-se que o número máximo de veículos da cidade é de 5.000 unidades e que os valores não precisam ser lidos.

- a) Construa um algoritmo que liste todos os proprietários cujos carros são do ano de 1980 e acima e que sejam movidos a diesel.
- b) Escreva um algoritmo que liste todas as placas que comecem com a letra A e terminem com 0, 2, 4 ou 7 e seus respectivos proprietários.
- c) Elabore um algoritmo que liste o modelo e a cor dos veículos cujas placas possuem como segunda letra uma vogal e cuja soma dos valores numéricos fornece um número par.
- d) Construa um algoritmo que permita a troca do proprietário com o fornecimento do número do chassi apenas para carros com placas que não possuem nenhum dígito igual a zero.

11. Escrever um algoritmo para a formação da seguinte matriz:

1	1	1	1	1	1
1	2	2	2	2	1
1	2	3	3	2	1
1	2	3	3	2	1
1	2	2	2	2	1
1	1	1	1	1	1

Procedimentos e Funções

Ao desenvolvermos um programa, muitas vezes precisamos utilizar uma pequena rotina repetidamente em mais de um local do mesmo, quando fazemos isso, estamos ocupando mais espaço de memória, aumentando a complexidade do algoritmo, aumentando o tamanho do programa. Para evitar tudo isso, podemos modularizar o programa, isto é, criar procedimentos e funções (módulos) uma vez apenas, e chamá-los repetidamente de várias partes do programa, por um nome que os identifica.

Procedimentos

Aqui aplicamos efetivamente o termo “Dividir para Conquistar”, quando nos depararmos com um problema, e formos resolvê-lo por meio de algoritmos, devemos dividi-lo em vários problemas menores, se necessário, dividimos novamente em vários outros problemas menores, resolvendo estes problemas estaremos conseqüentemente resolvendo nosso problema inicial.

Como exemplo, faremos um algoritmo que imprima o extenso do número 1 ou 2.

escreva

num : inteiro;

Procedimento mostre_um;

início

escreva('Um');

fim;

Procedimento mostre_dois;

início

escreva('Dois');

fim;

início

Repita

leia(num);

se num = 1 **então**

mostre_um;

se num = 2 **então**

mostre_dois;

Até num >= 2;

fim.

O que acontece neste algoritmo é o seguinte, os blocos que começam com a palavra **Procedimento**, são módulos que somente serão executados se chamados do principal ou outro módulo do algoritmo. Sendo assim o primeiro comando a ser executado é o comando **leia**, dentro da estrutura de repetição **Repita**, que irá conhecer o valor da variável num, o valor da variável é então testado, se for igual a 1 o procedimento mostra_um é chamado, quando isto acontece os comandos que estão contidos dentro do procedimento, neste caso somente o comando **informe**. Quando a ultima linha do procedimento é executada e encontra-se o comando **fim**, a execução do algoritmo volta exatamente ao comando após a chamada do procedimento, neste caso, após a linha mostra_um ou mostra_dois, é executado o teste da estrutura de repetição, se num for maior que 2 o algoritmo é encerrado, caso contrário, o algoritmo é executado novamente a partir do comando **leia**.

Escopo de Variáveis

Escopo de variáveis seria o mesmo que domínio sobre variáveis.

Quando uma variável é declarada no início do algoritmo, dizemos que ela é global, isto é, todos os módulos do algoritmo reconhecem esta variável. Quando uma variável é declarada dentro de um módulo, dizemos que ela é local, isto é, somente este módulo reconhece a variável. Variáveis declaradas no cabeçalho do módulo (parâmetros), são variáveis locais deste módulo.

Olhe para estes quadros e imagine que seja um algoritmo. O quadro maior é o módulo principal, dentro deles existem mais quatro quadros (1, 2, 3, 4) que são os módulos secundários (procedimentos, funções), sendo que ainda temos o quadro 2 dentro do quadro 1 - isto significa que o quadro 2 só pode ser chamado pelo quadro 1.

Observe que as variáveis K e Y são globais, elas são visíveis a todos os módulos; as variáveis J e H são visíveis apenas ao quadro 1 onde estão declaradas, e ao quadro 2 que está declarado dentro do quadro 1; as variáveis O e P são visíveis somente ao quadro 2; as variáveis O, J, Y são visíveis somente ao quadro 3; e as variáveis O, J, L, H, I e A são visíveis somente ao quadro 4.

Note que a variável Y dentro do quadro 3 não é a mesma variável Y do quadro principal, então, se Y é usado dentro de 3, é o Y mais interno que será usado. Isto é uma prática desaconselhável por tornar o programa confuso e com tendência a comportamentos indesejados.

Parâmetros

Agora imagine se pudéssemos mandar mensagens aos procedimentos, isto é, se pudéssemos passar parâmetros aos procedimentos, desta forma eles ganhariam muito mais poder e generalização. Vamos usar o exemplo anterior e ver como podemos diminuir o número de procedimentos.

Variáveis

num : inteiro;

Procedimento mostre_num (numero : caracter); **início**

 escreva(numero);

fim;

início

Repita

 leia(num);

se num = 1 ou num = 2 **então** mostre_num(num);

Até num >= 2;

fim.

Observe como nosso algoritmo ficou menor e mais legível, conseguimos isto com a passagem de parâmetros para o procedimento.

Funções

Como você pôde perceber, os procedimentos são blocos de comandos que executam determinada tarefa, mas não retornam nenhum valor, ao contrário das **Funções**, que são especificamente desenvolvidas para retornarem valores aos módulos que as chamam.

Como exemplo vamos construir um algoritmo que retorne o valor de uma expressão matemática:

variáveis

X : real;

Função Expressão : Real;

início

 expressão \leftarrow 8 * X ;

fim;

início

 leia (X);

 X \leftarrow expressão;

 escreva(X);

fim.

Passagem de Parâmetros por Referência e Valor

Como já falamos, somente funções podem retornar valores e procedimentos não. Mas existe uma forma de fazer com que uma variável que foi passada como parâmetro, tanto para um procedimento

como para uma função, volte com o valor alterado. Para isso você deve conhecer o conceito de passagem de parâmetros por Referência e Valor.

Na **passagem por valor**, você apenas passa o conteúdo da variável e não seu endereço físico, isto é, se o conteúdo da variável do cabeçalho, que está *recebendo* o valor, for alterado dentro do procedimento ou função, não irá alterar o conteúdo da variável que foi usada para *passar* o valor.

Já na **passagem por referência**, você não apenas passa apenas o conteúdo da variável e sim seu endereço físico, isto é, se o conteúdo da variável do cabeçalho, que está *recebendo* o valor, for alterado dentro do procedimento ou função, irá alterar o conteúdo da variável que foi usada para *passar* o valor.

Exemplo:

Função quadrado (X : Real) : Real;

início

expressão \leftarrow X * X;

fim;

Procedimento teste (val str : Caracter, X : Real); **início**

Se quadrado(X) > 20 **então**

str \leftarrow 'Maior que 20';

senão

str \leftarrow 'Menor que 20';

fim;

Procedimento informe_teste;

variáveis

i : inteiro;

mens : caracter;

início

leia(i);

teste (mens, i);

escreva(mens);

fim;

início

informe_teste;

fim.

Exercícios

- 1.Reconstrua o algoritmo de cálculo fatorial, modularizando com procedimento e função.
- 2.Faça um algoritmo que conheça as quatro notas bimestrais e, informe a média do aluno e se ele passou; média para aprovação 6.

3. A conversão de graus Fahrenheit para centígrados é obtida pela fórmula $C = 5/9 (F-32)$. Escreva um algoritmo que calcule e escreva uma tabela de graus centígrados em função de graus Fahrenheit que variem de 50 a 150 de 1 em 1.
4. Uma rainha requisitou os serviços de um monge e disse-lhe que pagaria qualquer preço. O monge, necessitando de alimentos, indagou à rainha sobre o pagamento, se poderia ser feito com grãos de trigo dispostos em um tabuleiro de xadrez, de tal forma que o primeiro quadro deveria conter apenas um grão e os quadros subsequentes, o dobro do quadro anterior. A rainha achou o trabalho barato e pediu que o serviço fosse executado, sem se dar conta de que seria impossível efetuar o pagamento. Faça um algoritmo para calcular o número de grão que o monge esperava receber.

Recursividade

Um objeto recursivo é aquele que consiste parcialmente ou é definido em termos de si próprio. Ex.: Números naturais, estruturas em árvores, certas funções matemáticas, fatorial, fibonacci, etc... A ferramenta para codificar programas de modo recursivo é o subalgoritmo.

Recursividade Direta

Quando um subalgoritmo (procedimento ou função) possuir uma referência explícita a si próprio através de seu nome.

Exemplo:

Procedimento A;

Início

:

A;

:

Fim;

Recursividade Indireta

Quando um subalgoritmo possuir uma chamada a um segundo subalgoritmo e este possuir uma referência direta ou indireta ao primeiro.

Exemplo:

Procedimento A;

Procedimento B;

Início

Início

:

:

B;

A;

:

:

Fim

Fim

- Deve-se declarar variáveis, constantes e tipos como locais, pois a cada chamada recursiva do subalgoritmo um novo conjunto de variáveis locais é criado, como também para os parâmetros passados. (sempre por valor).

- Em uma chamada recursiva deve-se considerar a possibilidade da ocorrência da não terminação dos cálculos, considerando-se então o critério de parada.
- Evitar níveis de recursão muito profundos, devido a ocupação de memória (variáveis locais, estado corrente do processamento, valor de retorno, valores de parâmetros de entrada e saída).
- Um algoritmo recursivo é empregado quando o problema ou os dados a que se destina for definido de forma recursiva.
- É preferível evitar o uso da recursão sempre que existir uma solução não recursiva (iterativa) para o problema.

Nem sempre um programa recursivo é rápido.

O método para transformar um programa não-recursivo num programa recursivo é a criação de uma pilha definida pelo usuário.

A técnica de recursão consiste em transformar uma parte do programa, que será alguma estrutura de repetição.

Não recursivo

$i := N$

Enquanto $i > 1$ **Faça**

$i := i-1;$

fimenquanto;

Recursivo

Função Dec (i : Inteiro);

Início

Se $i = 1$ **então** Dec $:= i;$

senão

início

$i := i-1;$

Dec (i)

fim;

fim;

Arquivos

Chegamos em um ponto, onde você com certeza já percebeu que é totalmente inútil ficar fazendo algoritmos, algoritmos e algoritmos, se não podemos gravar nossas informações. Imagine uma biblioteca informatizada, onde todos os livros e sócios estão cadastrados; é impossível cadastrar tudo, toda vez que liga-se o computador; da mesma forma que é impossível manter o computador 24 horas por dia ligado para não perder as informações.

Para evitar isso, nós vamos aprender a usar os arquivos, onde poderemos guardar nossas informações, e recuperá-las a qualquer momento, mesmo desligando o computador.

Para você ter uma idéia de como tratamos os arquivos em computadores, faremos uma comparação com um arquivo de ferro:

Quanto à estrutura: Imagine um arquivo de ferro, com uma gaveta apenas; dentro do arquivo existem várias fichas numeradas, estes números são as posições no arquivo; e todas as fichas contêm informações padrões, estas informações são os registros.

Quanto à manipulação: Logicamente para você operar um arquivo, primeiramente ele deve existir, você então compra este arquivo - no nosso caso criamos o arquivo. Toda a vez que você irá guardar (gravar) ou pegar (ler) uma ficha (registro em determinada posição), você deve abrir o arquivo. Cuidado, se o arquivo já estiver aberto, não tente abri-lo novamente. Após aberto o arquivo, se vai guardar uma nova ficha, você deve numerá-la na sequência das fichas já existentes. Se vai pegar uma ficha, você deve achar o número da ficha que quer pegar. Depois de feito isto, não esqueça de fechar o arquivo.

Definição

Um arquivo é um conjunto de registros (ou seja, é uma estrutura de dados), no qual cada registro não ocupa uma posição fixa dentro da estrutura, não possuindo portanto tamanho pré estabelecido. Os registros são formados por unidades de informação, denominadas campos. Este conjunto de registros é armazenado em um dispositivo de memória secundária (disco).

•Organização dos Arquivos

- As operações básicas que podem ser feitas em um arquivo através de um algoritmo são: leitura, gravação, alteração e exclusão de registros. Basicamente, existem duas possibilidades de organização:

•Seqüencial

Na organização seqüencial os registros são lidos ou gravados no arquivo em ordem seqüencial. Deste modo, a localização de qualquer um dos registros armazenados é indeterminada, ou seja, para acessar um registro específico é necessário obedecer a sua ordem de gravação, o que implica percorrer todos os registros que o antecedem.

•Direta ou Aleatória

- Na organização direta, o acesso aos registros é feito em ordem aleatória, podendo-se acessar um registro específico diretamente, sem se preocupar com os seus antecessores, utilizando neste acesso o mesmo campo que determinou sua posição no arquivo no instante da gravação. Num

arquivo de acesso direto, os registros não ficam localizados na ordem em que são gravados, o que leva a entender que cada registro possui “um lugar reservado” para ser armazenado.

Podemos comparar o trabalho com arquivos seqüenciais com a utilização de fitas K7. E o trabalho com arquivos diretos com a utilização de LP's

•Declaração

- Sendo que um arquivo é um conjunto de registros, é necessário definir o registro que compõe o arquivo primeiro, para somente então, definir o arquivo.

Tipo

- Ficha = **Registro**
- código : **inteiro**;
- nome : **caracter**;
- **Fim registro**;
- ArqCli = **Arquivo de** Ficha;

Variável

- cliente : **arq-cli**;
- Reg_cli : **Ficha**;
-

•Manipulação de Arquivos

Generalizando, pode-se admitir que todo arquivo possui maneiras semelhantes de ser manipulado, independentemente de como foi concebido.

-

•Abertura

Todo arquivo deve ser aberto para que as informações contidas, possam ser recuperadas. Isto é, através do comando **Abra** (nome-do-arquivo).

Ex.: **Abra** (clientes);

- Após a execução do comando abertura, o registro que estará a disposição será sempre o que está na primeira posição.
-

•Fechamento

- Após o uso do arquivo, é necessário que ele seja fechado, pois o seu conteúdo poderá estar exposto a agentes externos que podem danificar sua integridade. Para tal utiliza-se o comando **Fech** (nome-do-arquivo);

Ex.: **Fech** (clientes);

-

•Leitura

- Num arquivo não se deve retirar informações, pois se assim fosse este se tornaria vazio

rapidamente. Deve-se ter em mente que, um arquivo não deve ser “consumido” e sim consultado, e para isso, deve-se copiar o conteúdo desejado para um outro registro na memória, utilizando o comando **Leia** (nome-do-arquivo, variável);

•Ex.: **Leia** (clientes, reg_cli);

•

•**Gravação**

- Para a gravação de um registro no arquivo, faz-se necessário que este possua estruturação de campos idêntica a dos registros já armazenados, e ao mesmo tempo esteja completamente preenchido. Para tal, utiliza-se o comando **Grave** (nome-do-arquivo, variável);

•Ex.: **Grave** (clientes, reg_cli);

•

•**Posicionamento e verificação de fim e início de arquivo**

•

•**Avance (nome-arquivo):**

Posiciona o arquivo na posição consecutiva, ou seja, no próximo registro. Se utilizado repetidas vezes permite percorrer o arquivo passando por uma série consecutiva de registros.

Ex.: **Avance** (clientes);

•

•**FDA (nome-arquivo)**

- Esta instrução retorna verdadeiro quando a posição corrente é o fim do arquivo e falso caso contrário.

•Ex.:

•**Enquanto não FDA** (clientes) **Faça**

Leia (clientes, reg_cli);

Avance (clientes);

•**Fim enquanto**

•

•**Posicione (nome-arquivo, chave):**

Comando utilizado para que a posição corrente do arquivo seja aquela correspondente ao valor da variável chave.

Ex.:

Abra (clientes);

Escreva (“Qual Posição?”);

Leia (Reg_cli.posição);

Leia (Reg_cli.cod.);

Leia (Reg_cli.nome);

Posicione (clientes, reg_cli.posição);

Grave (clientes, reg_cli);

Feche (clientes);

Estudo de Concepções

Apesar de um arquivo ser concebido direta ou seqüencialmente, isso não obriga necessariamente que ele seja sempre manipulado como foi concebido. Em outras palavras, um arquivo concebido randomicamente pode ser acessado seqüencialmente, sob certas condições especiais, assim como um arquivo seqüencial pode ser acessado diretamente se for necessário.

Arquivo Direto Acessado Seqüencialmente

Um arquivo concebido randomicamente passou por um certo planejamento, que antecedeu sua criação, onde foi definida a chave de acesso do arquivo. Caso esse planejamento tenha sido inadequado, pode permitir que surja a necessidade de obter informação do arquivo a partir de um campo não-chave, forçando uma busca seqüencial do arquivo randômico.

Arquivo Seqüencial Acessado Randomicamente: Arquivo Indexado

Um arquivo concebido seqüencialmente foi preenchido na mesma seqüência em que as informações foram surgindo. Isto por definição condena-o a gravar ou ler informações seqüencialmente, o que se tornaria inconveniente se este arquivo crescesse muito e se a quantidade de acessos fosse muito freqüente. Esta situação pode ser controlada através de um arquivo concebido seqüencialmente (ou não) de menor porte, facilitando assim a busca seqüencial. Este arquivo possui arquivada em cada registro a posição onde encontrar cada registro original, ou seja, é como se fosse um índice de uma enciclopédia. Desta forma, um novo arquivo criado é chamado arquivo de índices, e o arquivo original (seqüencial) é chamado de arquivo indexado, que através do índice fornecido pelo arquivo de índices pode ser acessado diretamente.

Exercícios:

1. Faça um algoritmo que cadastre as informações sobre as confecções de um loja. Cada registro terá a seguinte forma: Código, descrição, tamanho, cor e quantidade.
2. Utilizando o arquivo do algoritmo anterior, elabore uma consulta que poderá ser feita pela descrição ou pelo tamanho.
3. Crie outros três arquivos que armazenarão os registros com tamanhos P, M e G, respectivamente, lidos do arquivo original.