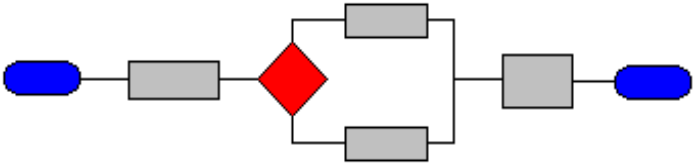
celta informática

Lógica de Programação

Sumário



INTRODUÇÃO À LÓGICA DE PROGRAMAÇÃO

Para um computador executar operações é necessário que ele seja programado, e esta programação é feita através de uma linguagem que “ele” entenda. Os computadores só executam instruções expressas em linguagem de máquina (forma numérica) e isto gera vários inconvenientes para o ser humano. Diante de tais problemas, surgiram as linguagens simbólicas de fácil compreensão para o homem, que são traduzidas para a linguagem de máquina por um programa chamado compilador.

A primeira destas linguagens foi o Fortran, com característica técnicas e científicas, é ainda hoje utilizada. Com o passar dos tempos, muitas outras linguagens surgiram, especificamente voltadas para problemas de natureza comercial e administrativa.

Para elaborar um programa é necessário ter toda a seqüência de passos e operações estabelecidas de modo formal para a resolução do problema. Para isso, encontraremos, nas próximas páginas, uma técnica muito útil e eficiente que é o algoritmo.

ALGORITMO

Um algoritmo é uma estratégia para resolver um desejado problema. Ele está associado ao desenvolvimento de uma técnica capaz de encontrar uma solução para este problema.

Os dados do problema são a entrada do algoritmo e a solução corresponde à saída deste algoritmo. Considera-se a entrada como uma variável independente básica, a partir da qual são produzidas as saídas do algoritmo.

Os algoritmos são escritos quase que livremente em língua portuguesa, apresentando a estratégia desejada, contendo algumas sentenças especiais com formato predefinido.

A apresentação de um algoritmo é formada por uma seqüência de instruções ordenadas em linhas de maneira a dar em seu decurso a solução do problema. Cada linha possui uma margem correspondente ao seu início. Sejam *r*, *s* duas linhas de um algoritmo, r antecedendo s. Diz-se que a linha r contém a linha s, quando a margem de s é mais à direita do que a de r, a isto chamamos de **edentação**.

ⓠ ..... Calcular subtração entre X e Y.
ⓡ ..... Se resultado for negativo
ⓢ .... inverta o sinal do resultado.
ⓣ ..... Exibir o resultado.

Além da estrutura de edentação, um algoritmo possui também sentenças especiais, que podem ou não ser utilizadas. Estas estruturas são: comentários, declaração e atribuição, desvios, condiçõesl e repetições.

Exemplo de um algoritmo

Rotina {este exemplo ilustra um algoritmo}
    Invente um problema
    Enquanto existir erro
        análise do problema
        tentativa de resolução
    FimEnquanto
    Se gostou da resolução **Então**
        apresente aos amigos
    SeNão
        invente outro problema
    FimSe
FimRotina

Todo algoritmo deverá iniciar com a palavra **Rotina** e terminar com **FimRotina**.

ITENS FUNDAMENTAIS DO ALGORITMO

Como vimos no exemplo anterior, os algoritmos são simples e bem objetivos, possuem palavras chaves - em negrito, que nos auxiliam na ordenação das instruções. No desenvolvimento de um algoritmo, precisamos analisar todas as possibilidades e ordená-las da maneira mais simples possível a fim de atingir o objetivo. Este processo requer paciência e persistência, por sinal, estas são qualidades importantes para um programador, principalmente no início da carreira.

Variáveis

Uma variável corresponde a uma posição de memória cujo conteúdo poderá variar durante a execução de um programa. Em uma equação matemática (A+B=7), A e B representam as posições de memória que conterão as parcelas da equação.

As variáveis são identificadas por um nome que é composto de um ou mais caracteres, o primeiro caractere deve ser uma letra e não poderá conter caracteres especiais (?/']^&%$#@) e nem ser uma palavra reservada da linguagem utilizada.

Exemplos:

valor; B2; matriz; código, A

Os nomes das variáveis deverão ser claros e objetivos. Se em um algoritmo for necessário armazenar o total de vários números, um bom nome para esta variável seria soma ou total.

As variáveis só armazenam valores de um mesmo tipo, isto é, precisaremos declarar no algoritmo, qual o tipo de dado que a variável irá trabalhar, e não poderemos atribuir a esta variável valores diferentes do tipo declarado.

Constantes

Uma constante corresponde a uma variável que não pode ter o seu valor modificado durante a execução do algoritmo. Existem constantes numérica, de caractere ou lógica.

Exemplos:

a) Numérica - 5; 7,5 -
b) Lógica - Falso (F); Verdadeiro (V); Sim (S); Não (N)
c) Caractere - “Renato”; “X5”; “27” - (as constantes caracteres são representadas entre aspas)

Expressões

Uma expressão pode ser a combinação de uma ou mais constantes, variáveis, operadores e/ou funções. As expressões mais comuns são as aritméticas que contém os operandos (constantes, variáveis e/ou funções) ligados por um ou mais operadores (+,-, *, /).

Exemplos:

```
valor * 10
salário - imposto
total - (total*0,2)
“Sr. “+ nome
```

Operadores

Os operadores são representados por símbolos, funcionando como relacionamentos que criam um único resultado. Existem os operadores aritméticos, relacionais, lógicos e de strings (cadeia de caracteres).

Operadores aritméticos

Os operadores matemáticos a seguir estão em ordem de prioridade (cima para baixo) que são efetuadas as operações:

Operação	Operador
exponenciação	^
multiplicação	*
divisão	/

adição	+
subtração	-
inteiro da divisão	div
resto da divisão	mod

Operadores relacionais

Os operadores relacionais servem para comparar dois valores de mesmo tipo, e nesta relação poderemos encontrar constantes, variáveis ou expressões. Os operadores relacionais são:

Operação	Operador
igual a	=
diferente de	<>
menor que	<
maior que	>
menor que ou igual a	<=
maior que ou igual a	>=
membro de	in

O resultado de uma comparação obtido utilizando-se um operador relacional, sempre será um valor lógico.

Exemplos:

6 > 7 resulta em Falso

A + B = 5 o resultado poderá ser Verdadeiro ou Falso, dependendo do valor da expressão A+B.

5,73 **in** N resulta em Falso, pois 5,73 não é membro do conjunto dos números naturais.

Operadores lógicos

Os operadores lógicos também comparam valores de mesmo tipo para criar uma lógica verdadeira ou falsa, sim ou não, utilizando a lógica booleana. Os operadores lógicos são:

Operação	Operador
conjunção	E
disjunção	OU
negação	NÃO
exclusão	XOU

Exemplo:

Admitindo que a variável A armazene o valor 5 e a variável B o valor 3, temos que:

a) (A > 4 **E** B > 4) expressão Falsa
b) (A > 4 **OU** B > 4) expressão Verdadeira
c) (**NÃO** A > 4) expressão Falsa
d) (A > 4) expressão Verdadeira
e) (A > 4 **XOU** B < 4) expressão Falsa
f) (A > 4 **XOU** B > 4) expressão Verdadeira

No item **a**, obterei Verdadeiro se, somente se, as duas questões forem verdadeiras. O operador lógico **OU** retornará Verdadeiro se pelo menos uma das duas operações for Verdadeira. Para o operador de negação, o que teremos é o inverso da operação analisada. O operador **XOU** retorna Verdadeiro se um operando for verdadeiro e o outro falso.

Operador de caracter

O operador de strings é representado pelo sinal de adição (+), é utilizado para combinar duas ou mais séries de caracteres. Supondo que A e B são variáveis do tipo caractere, a expressão A+B resultaria um único literal formado pelo conteúdo de A seguido do conteúdo de B.

Exemplo:

Supondo que:

A = “Charles Babbage é “
B = “conhecido como “
C = “o pai do computador.”

Teremos:

A + C = “Charles Babbage é o pai do computador”
A + B + C = “Charles Babbage é conhecido como o pai do computador”

FUNÇÕES

As funções são utilizadas para a execução de operações mediante a passagem ou não de argumentos e, sempre retornam um valor.

Exemplos:

TRUNCA (C)	retorna a parte inteira de um nº fracionário (C)
COS(X)	retorna o cosseno do valor X
ARREDONDA(Z)	retorna, por arredondamento, um nº inteiro do valor fracionário (Z)
RAIZ(V)	retorna a raiz quadrada do valor V
HOJE	retorna a data atual

Existem diversas funções e elas variam de acordo com a linguagem de programação. Em nossos algoritmos, utilizaremos estas funções como exercícios, neste caso, elas são palavras-chave que não podem ser usadas como nome de variável, e serão identificadas com letras maiúsculas.

COMENTÁRIOS

Todo programador deve ter a preocupação com a clareza de seu algoritmo, ou seja, a facilidade com que outras pessoas poderão entender a lógica da resolução de um problema, para entendê-lo e/ou alterá-lo.

Os comentários em um algoritmo devem estar entre chaves “{ } “, ou barra dupla “//” no início da linha.

Exemplo:

{O texto de comentário deve estar entre chaves}
//Ou com duas barras no início da linha.

DECLARAÇÕES E ATRIBUIÇÕES

Toda variável ou constante deve ser declarada com um nome seguido de um tipo, conforme o seu conteúdo. Se declararmos uma variável como numérica, não poderemos armazenar textos ou valores lógicos. Utilizaremos as seguintes formas para declararmos variáveis e constantes.

Declare lista-de-nomes **tipo** {*declaração de variáveis*}
Constante tipo nome = valor {*declaração de constantes*}

Declare/Constante	palavra-chave do algoritmo
lista-de-nomes	nomes das variáveis
tipo	um dos tipos permitidos

Exemplo:

Declare valor, total **Numérico** {*declara as variáveis valor e total como sendo numéricas*}
Declare nome **Caractere**
Declare retorno **Lógico**
Constante Numérica Inflação = 0,4
Constante Caractere Cabeçalho = "Meu Nome"

O comando de atribuição é descrito como de uma ação a ser executada em um dado momento. O comando de atribuição fornece um valor a uma certa variável do mesmo tipo com que ela foi declarada. O sina de igual (=) será o nosso símbolo de atribuição.

Identificador = expressão

Exemplos:

A = 5	o valor cinco foi atribuído à variável A
Total = (a+b) * 3	a expressão à direita da seta é calculada e o resultado é atribuído à variável total.

COMANDOS DE ENTRADA E SAÍDA

O fluxo de dados em um computador poderia ser representado, basicamente, pelo diagrama abaixo:

O momento da entrada poderá ser iniciado com dados obtidos externamente, como exemplo, a entrada da teclado ou a leitura do código gravado em um cartão magnético. Muitos programas iniciam o processamento mediante o recebimento de um dado (entrada) e geram, após o processamento, um ou vários resultados (saída).

Em nossos algoritmos utilizaremos comandos para receber algum dado, e enviar para o usuário o resultado do processamento. O meio através do qual receberemos ou enviaremos os dados, é irrelevante em nossos estudos sobre lógica de programação.

O comando de entrada terá a seguinte forma:

Receba lista-de-variáveis (separadas por vírgula)
Receba "<informação>" : variável

Exemplo:

Receba código, senha {*recebe código e senha*}
Receba "Entre com a idade" : Idade {*mostra a mensagem e recebe o valor para Idade*}

Para exibir uma saída, a sintaxe será:

Escreva lista-de-variáveis e/ou constantes

Exemplo:

código = 20
senha = "13A"
Escreva "Valor ", código, " senha ", senha {neste caso, teremos as variáveis e constantes impressas uma ao lado da outra}

Será exibido o seguinte: Valor 20 senha 13^{A}

EXERCÍCIOS RESOLVIDOS

1) Faça um algoritmo que receba duas notas e imprima a média destes valores.

Solução:

```
Rotina
    Declare nota1, nota2, média Numérico
    Receba nota1, nota2
    média = (nota1+nota2) / 2
    Escreva média
FimRotina
```

2) Crie um algoritmo que receba três valores e escreva apenas a parte inteira referente ao cálculo do cosseno dos três valores somados.

Solução:

```
Rotina
    Declare valor1, valor2, valor3, cálculo, resultado Numérico
    Receba valor1, valor2, valor3
    cálculo = COS(valor1+valor2+valor3)
    resultado = TRUNCA(cálculo)
    Escreva "resultado = ", resultado
FimRotina
```

EXERCÍCIOS PROPOSTOS

1) Dados 5 números, monte um algoritmo que permita obter como resultado a soma, produto e média dos valores recebidos.

2) Crie um algoritmo que calcule o valor de Y a partir do valor de X recebido, para a sentença matemática: $Y = \frac{X^2}{3} - X + 7$, e imprima o resultado.

3) Num triângulo retângulo, segundo Pitágoras, o quadrado da hipotenusa (a) é igual a soma dos quadrados dos catetos (b e c), isto é, $a^2 = b^2 + c^2$. Num algoritmo, receba os valores dos catetos e imprima o valor da hipotenusa.

4) Receba dois valores nas variáveis A e B respectivamente, troque o valor contido na variável A pelo valor em B, e o valor em B pelo valor em A, isto é, imprimiremos A e B com os valores trocados.

5) Receba o código, nome e salário bruto de um determinado funcionário, logo após, calcule o salário líquido, sabendo-se que será deduzido 15% de imposto de renda. No final, imprima o nome do funcionário e o salário a receber.

ESTRUTURAS CONDICIONAIS

Até agora, nossos algoritmos foram executados numa seqüência linear, seguindo-se as ações de cima para baixo. A estrutura condicional irá permitir a escolha de ações e estruturas a serem executadas quando determinada condição for ou não satisfeita.

Temos dois tipos de estruturas condicionais, a estrutura **Se** ... **Então** e a estrutura **Caso** ... .

SE ... ENTÃO ... SENÃO

A estrutura condicional **Se** ... **Então**, tem as seguintes formas:

Condicional Simples

Se condição **Então**
    seqüência de comandos para condição verdadeira
FimSe

Condicional Composta

Se condição **Então**
    seqüência de comandos para condição verdadeira
SeNão
    seqüência de comandos para condição falsa
FimSe

CASO ...

A estrutura Caso ... , contém uma expressão (o seletor) e uma lista de declarações, cada declaração é anteposta por uma ou mais constantes (chamadas constantes de caso) ou com a palavra SeNão. O seletor deve ser de um tipo ordinal, ou seja, possua uma ordem. Todas as constantes de caso devem ser diferentes e de um tipo ordinal compatível com o tipo do seletor.

Esta estrutura executa a seqüência de comandos precedida pela constante igual ao valor do seletor. Se nenhuma constante for igual ao valor do seletor, a parte Senão será executada. Se não houver a parte SeNão, a execução continuará com o próxima comando que segue a estrutura Caso... .

Caso expressão
    Seja valor1 **Faça** comando
    Seja valor2 **Faça** comando
    Seja valor3 **Faça** comando
    Senão comando
FimCaso

Caso Resposta
    Seja 1 **Faça** Número = “Um”
    Seja 2 **Faça** Número = “Dois”
    Seja 3 **Faça** Número = “Três”
    SeNão Número = “Maior que três”
FimCaso

O exemplo anterior testa a variável Resposta, se ela for igual a 1 a variável caractere Número será “Um”, de modo semelhante acontecerá para Resposta igual a 2 ou 3. Se Resposta não for nenhum destes valores, a variável Número será “Maior que três”.

EXERCÍCIOS RESOLVIDOS

1) Crie um algoritmo que receba o nome e a nota de um determinado aluno. Caso a nota seja maior ou igual a sete, imprima aprovado, em caso negativo, imprima reprovado.

Solução:

```
Rotina
    Declare nota Numérico
    Declare nome, Caractere
    Receba nome, nota
    Se nota >= 7 Então
        Escreva nome, "aprovado"
    SeNão
        Escreva nome, "reprovado"
    FimSe
FimRotina
```

2) Faça um algoritmo que após receber dois valores numéricos, calcule a média e imprima o resultado se, e somente se, for maior que 33.

Solução:

```
Rotina
    Declare valor1, valor2, média Numérico
    Receba valor1, valor2
    média = (valor1+valor2) / 2
    Se média > 33 Então
        Escreva média
    FimSe
FimRotina
```

3) Escreva um algoritmo que receba dois números e o nome da operação matemática desejada entre eles (soma, subtração, multiplicação), exibindo o resultado.

Solução:

```
Rotina
    Declare X, Y Numérico
    Declare Operação Caractere
```

```
        Receba X, Y, "soma(1), subtração(2) ou multiplicação(3)" : Operação
        Caso Operação
                Seja 1 Faça Escreva  X + Y
                Seja 2 Faça Escreva  X - Y
                Seja 3 Faça Escreva  X * Y
        FimCaso
FimRotina
```

4) A partir da entrada de qualquer número, o algoritmo deverá informar o seguinte: "Número par" se a entrada for um número par menor que dez; "Número impar" se a entrada for um número impar menor que dez; "Entre 10 e 100" se a entrada for maior que dez e menor ou igual a cem e finalmente "Negativo ou maior que 100" caso contrário.

Solução:

```
Rotina
        Declare X Numérico
        Receba X
        Caso X
                Seja 0, 2, 4, 6, 8 Faça Escreva "Número par"
                Seja 1, 3, 5, 7, 9 Faça Escreva "Número impar"
                Seja 10..100 Faça Escreva "Entre 10 e 100"
        SeNão Escreva  "Negativo ou maior que 100"
        FimCaso
FimRotina
```

EXERCÍCIOS PROPOSTOS

1) Encontre os valores de saída escritos pelo algoritmo a seguir, se fossem recebidos os valores 7, 3 e 5, relativos as variáveis X, Y e Z respectivamente.

```
Rotina
        Declare X, Y, Z, M Numérico
        Receba X, Y, Z
        M = TRUNCA (Z-X)/(X-Y)
        Se M <= 7 Então
                M = X
        X = Y
                Y = M
        SeNão
```

```
                M = ARREDONDA (Z/X+2)
                Y = Z
                Z = M
        FimSe
        Escreva X, Y, Z
FimRotina
```

2) Fazer um algoritmo que após receber três valores, imprima apenas os números pares. Um número par quando dividido por 2, o resto será igual a zero.

3) Dados três valores A, B, C, determine e imprima o menor deles.

4) Faça um algoritmo que ao receber três valores, verifique se eles podem ser os comprimentos dos lados de um triângulo. Se não for um triângulo, imprima uma mensagem.
   Propriedade: O comprimento de um lado do triângulo é menor do que a soma dos comprimentos dos outros dois lados.

5) Receber três valores distintos e colocá-los em ordem crescente. No final, escreva-os na nova ordem. Utilize o operador lógico E.

6) Recebidos valores numéricos entre zero e cinco, escreva-os na forma literal.

7) A partir do exercício anterior, pergunte ao usuário se deseja os numerais em inglês ou português.

ESTRUTURAS DE REPETIÇÃO

A estrutura de repetição permite que uma seqüência de comandos seja executada um determinado número de vezes ou enquanto uma determinada condição for satisfeita, ou então, encontre um comando de interrupção.

Temos três tipos de estruturas de repetição: **Para** ... **Até** ... **Faça**, **Repita** ... **Até que** e **Enquanto** ... **Faça**.

PARA ... ATÉ ... FAÇA

A estrutura **Para** ... **Até** ... **Faça** repete uma seqüência de comandos um determinado número de vezes, enquanto o valor de uma variável de controle é incrementado.

Para I = 1 **Até** 15 **Faça**
    seqüência de comandos {*esta seqüência será repetida quinze vezes*}
FimPara

Exemplo:

```
Para I = 1 Até 5 Faça
    Se I mod 2 =0 Então        {condição verdadeira se resto = 0}
        Número = "Par"         {atribui "Par" à Número}
        Escreva Número
    FimSe
FimPara
```

A saída deste algoritmo será: 2 e 4.

REPITA ... ATÉ QUE ...

A estrutura **Repita ... Até que...**, contém uma condição que controla a execução repetidas vezes de uma seqüência de comandos. Esta condição tem que produzir um resultado de tipo Booleano - verdadeiro ou falso. A seqüência de comandos será executada até que a condição seja satisfeita.

A seqüência de comandos será executada pelo menos uma vez, porque a condição é avaliada somente depois da execução de cada seqüência de comandos.

Repita
    seqüência de comandos
Até que condição

Exemplo:

```
Repita
```

```
        Leia A
        A = A + 1
    Até que A > 19      {avalia se A >19, se for, para o loop}
    Escreva A           {escreverá somente valores maiores que 19}
```

ENQUANTO ... FAÇA

Uma estrutura **Enquanto ... Faça**, contém uma condição que controla a execução repetidas vezes de uma seqüência de comandos. A condição que controla a repetição deve ser de tipo Booleana. Ela é avaliada antes da seqüência de comandos ser executada.

A seqüência de comandos será executada repetidamente contanto que a condição seja Verdadeira. Se a condição no princípio for Falsa, a seqüência de comandos não será executada em nada, passando o controle para a linha seguinte a **FimEnquanto**.

```
Enquanto condição1 Faça
    seqüência de comandos
    Se condição2 Então      {se a condição2 for verdadeira interrompe loop}
        Interrompa          {comando de interrupção opcional}
    FimSe
    seqüência de comandos
FimEnquanto
```

Nesta estrutura, verificamos que é possível termos outros tipos de estruturas dentro da estrutura de repetição, esta situação é comum para qualquer tipo de estrutura. O comando **Interrompa**, força um salto, na seqüência, para os comandos que estão logo após a expressão **FimEnquanto**, e seu uso não é obrigatório.

EXERCÍCIOS RESOLVIDOS

1) Faça um algoritmo que some os números compreendidos entre os valores 10 e 100, e no final imprima o valor total da soma.

Solução:

```
Rotina
    Declare soma, valor Numérico
    valor = 11
    soma = 0
    Enquanto valor <= 100
        soma = soma + valor
        valor = valor + 1
    FimEnquanto
    Escreva "Total = ", soma
FimRotina
```

2) Crie um algoritmo que receba um número inteiro diferente de zero, e calcule o fatorial deste número. (Fatorial de um número é igual ao produto dos números 1 ao número desejado - inclusive. Ex: 3! (fatorial de 3 é igual a: 1x2x3 = 6)

Solução:

```
Rotina
    Declare valor, contador, total Numérico
    total = 1
    Receba valor
    Para contador = 1 Até valor
        total = total * contador
    FimPara
FimRotina
```

3) Escreva um algoritmo que repita a soma dos números recebidos até que o total seja maior que cem.

Solução:

```
Rotina
    Declare X ,Total Numérico
    Total = 0
    Repita
        Receba X
        Total = Total + X
    Até que Total > 100
    Escreva Total
FimRotina
```

EXERCÍCIOS PROPOSTOS

1) Faça um algoritmo que receba 100 valores inteiros e positivos, e resolva as seguintes questões:

    a) Calcule a raiz quadrada dos valores menores que 100;
    b) Para todo valor maior que 100, verifique se o mesmo é par, caso positivo, imprimir o valor e a mensagem “valor par”;
    c) Ao final, imprimir a quantidade de números pares.

2) A conversão de graus Farenheit para centígrados é obtida por C = 5(F - 32)¸9. Fazer um algoritmo que calcule e escreva uma tabela de graus centígrados em função de graus Farenheit, variando um a um de 50 a 150 graus Farenheit,

3) Foi feita uma pesquisa de canal de TV em várias casas de uma certa cidade, num determinado dia. Para cada casa visitada, foi preenchido uma ficha contendo o número do canal (2,4,5,13) e o número de pessoas que estavam assistindo naquela casa. Faça um algoritmo que:

    a) Receba um número indeterminado de fichas, sendo que a última ficha contém o número do canal igual a zero;
    b) Calcule a porcentagem de audiência para cada emissora;
    c) Escreva o número do canal e sua respectiva porcentagem.

4) Fazer um algoritmo que calcule e escreva o valor de S onde:

5) Fazer um algoritmo que calcule e escreva o número de grãos de milho que se pode colocar num tabuleiro de xadrez, colocando 1 na primeira casa e nas casa seguintes o dobro da casa anterior. Sabe-se que um tabuleiro de xadrez tem 64 casas.

DESVIOS

Por desvios entendemos todo comando que desvia a execução do programa de sua linha principal, podendo ser chamados também de sub-rotinas. Muitas vezes é possível dividirmos um problema grande em problemas pequenos, onde teremos algoritmos menores de fácil resolução. Estes algoritmos menores também possuem entrada, processamento e saída. Como desvios temos o comando *ir para*, procedimentos e funções.

IR PARA

Uma declaração *Ir para* transfere a execução do programa para a linha marcada por um nome específico. Quando todas as declarações deste desvio forem executadas, volta-se para a linha seguinte a que chamou o comando *Ir para*.
Quando utilizar o comando *Ir para*, observe as seguintes regras:

A linha referida por uma declaração *ir para* devem estar no mesmo bloco com a declaração ir para. Em outras palavras, não é possível saltar para ou fora de um procedimento ou função.

Boas práticas de programação recomendam que você utilize o comando *ir para* menos possível.

Ir para nome-do-bloco

Exemplo:

```
Rotina
    Declare A, SOMA Numérico
    Ir Para Desvio
    SOMA = A + 20

Desvio:
    A = 10
FimRotina
```

PROCEDIMENTOS

Utilizamos os *procedimentos* para evitarmos a repetição de uma seqüência de comandos que é utilizada em várias partes do programa, dividindo e estruturando o algoritmo em partes fechadas e coerentes.

Uma declaração de procedimento ativa um procedimento especificado por um identificador de procedimento. Se a declaração de procedimento correspondente contém uma lista de parâmetros formais, então a chamada do procedimento tem que ter uma lista de parâmetros atuais (parâmetros listados em definições são parâmetros formais; na declaração de chamada, eles são parâmetros atuais). Os parâmetros atuais são passados aos parâmetros formais como parte da chamada.

Procedimento nome (parâmetros formais)

declaração dos objetos locais

seqüência de comandos

FimProcedimento

Exemplo:

```
Rotina
    Declare A, B, C, D, RESULT Numérico
    Procedimento Soma ( X, Y )          {X e Y são parâmetros formais}
        RESULT = X + Y
    FimProcedimento
    Leia A, B, C, D
    Soma (A, C)         {executa o procedimento atribuindo A a X e C a Y}
                                {A e C são parâmetros reais}
    Se RESULT > 20 Então
        Soma ( B, D)
    FimSe
    Escreva RESULT
FimRotina
```

Funções

Uma Função define um bloco do algoritmo que computa e devolve um valor, semelhante às funções matemáticas vistas anteriormente. A declaração de uma função é semelhante a de um procedimento, com exceção da especificação do tipo de dado retornado por esta função.

Função tipo nome (parâmetros){*tipo = tipo de dado de retorno*}
    declaração dos objetos locais
    seqüência de comandos
FimFunção

Uma função é ativada utilizando-se o seu nome junto com os parâmetros requeridos por ela. A função então é executada, e o valor de retorno deverá ser atribuído a uma variável com o nome desta função ou à variável Resultado, que armazena o retorno da função.

Exemplo:

```
Rotina
    Declare A, B, Nota Numérico
      Função numérico Média (A, B)
          Se A < 1 Então
              Média = ((A ^ 2) *( B ^ 2)) ^ 0,5        {média geométrica, √(A².B²) }
          SeNão
              Média = (A + B) / 2                      {média aritmética}
          FimSe
      FimFunção
    Leia A, B
    Nota = Média (A, B)
    Escreva Nota
FimRotina
```

A função acima retornará a média entre A e B, atribuída à variável Média. Será calculada média geométrica se A<1 e média aritmética caso contrário.

EXERCÍCIOS RESOLVIDOS

1) Faça um algoritmo que escreva o resultado e a paridade das seguintes funções:

$Y = X^3 - 5$

$Y = 7X^3$

Solução:

```
Rotina
    Declare X, Y numérico
    Declare Número caractere
    Procedimento Verificar          {procedimento sem parâmetros}
        Se Y mod 2 = 0 Então
            Número = "Par"
        SeNão
            Número = "Impar"
        FimSe
    FimProcedimento
    Receber X
    Y = X³ - 5
    Verificar
    Escrever Número; Y
    Y = 7X³
    Verificar
    Escrever Número; Y
FimRotina
```

2) Construa uma tabela, com 0 <= X <= 10, para a seguinte função:

$Y = X^2 + 2X + 3$; para $X < 5$

$Y = (X + 1)^2 + 2(X + 1) + 3$; para $5 <= X < 7$

$Y = (X / 2)^2 + X + 3$; para $X >= 7$

Solução:

```
Rotina
    Declare X, Y numérico
    Função numérico Func (X)
        Resultado = X² + 2X + 3
    FimFunção
    Para X = 0 Até 10 Faça
        Se X < 5 Então
```

```
                    Y = Func (X)
                SeNão
                    Se (X >= 5) E (X < 7) Então
                        Y = Func (X + 1)
                    SeNão
                        Y = Func ( X / 2)
                    FimSe
                FimSe
            FimPara
FimRotina
```

EXERCÍCIOS PROPOSTOS

1) Elabore um algoritmo que calcule o valor de uma prestação em atraso utilizando a seguinte fórmula: ValorAtrasado = Prestação + (Prestação * (1 + TaxaJuros/100) ^ Tempo).

2) Desenvolva um programa que calcule uma potência qualquer, enviando-se para um procedimento os valores da base e do expoente.

3) Faça um programa de calculadora com as funções de soma e subtração, que escreva no início as opções que o usuário deverá escolher para definir o operando, a seguir, peça os valores dos operandos e apresentando o resultado.

ARQUIVOS

Verificamos que as variáveis são meios de armazenamento de informações, mas esta entidade é transitória, isto é, desaparece quando o aplicativo é fechado perdendo-se todos os dados nelas contidos. E quando o aplicativo é reiniciado, as variáveis são recalculadas. Para armazenar dados permanentes que possam ser acessados e recuperados, as informações destas variáveis devem ser salvas em arquivos.

Arquivo é um conjunto de dados armazenados em um dispositivo permanente que pode ser qualquer meio magnético (disco rígido, disquete) ou óptico (CD-ROM).

Basicamente existem dois tipos de organização de arquivos, a organização seqüencial, na qual os registros são obtidos ou inseridos no arquivo em ordem seqüencial, e a organização randômica, em que o acesso do registro é feito em ordem aleatória, isto é, poderemos encontrar qualquer registro entre vários outros, mediante uma chave de acesso, este recurso assemelha-se a um índice de livro que nos indica exatamente onde encontrar o assunto que desejamos estudar.

Em nossos algoritmos, vamos introduzir o uso de arquivos para entendermos a lógica deste recurso tão utilizado nos programas de computador; neste caso, os arquivos do tipo seqüencial serão melhor entendidos durante o estudo da linguagem de programação, por isto, iremos exercitar o uso deste recurso com apenas arquivos de acesso randômico.

REGISTRO

Entende-se por registro, um conjunto de componentes de vários tipos relacionados em um único identificador.

Considere a ficha de inscrição a seguir:

Ficha de Inscrição

Código:	23
Nome:	Marcos do Campo
Endereço:	Rua Mirian, 37 - São Paulo
Telefone:	478-2334
Curso:	Linguagem de Programação
Cliente:	S

Podemos considerar esta ficha como sendo um registro com os componentes Código, Nome, Endereço, Telefone, Curso e se é ou não Cliente, sendo estes chamados de campos. O conteúdo destes campos completa o conjunto de informações do nosso registro. A relação lógica entre estas informações é a sua associação a uma pessoa. Um conjunto destes registros formará um arquivo.

Observe os campos na ficha de inscrição, note que esta ficha possui dados de vários tipos:

Dado	Tipo
Código	numérico
Nome	caractere
Endereço	caractere
Telefone	caractere
Curso	caractere
Cliente	lógico

O campo Telefone é considerado do tipo caractere porque nenhuma operação matemática é realizada com ele, e o campo Cliente indica se esta pessoa é cliente ou não.

Declaramos um registro da seguinte forma:

Declare nome **Registro** (declaração dos campos)

Exemplo :

Declare ficha **Registro** (Código **numérico**
Nome **caractere**
Endereço **caractere**
Telefone **caractere**
Curso **caractere**
Cliente **lógico**)

Para ter acesso a um campo do registro utilize;

nome-do-registro **.** nome-do-campo

Por exemplo, atribuímos o valor do campo **nome** do registro **ficha** à variável Cliente utilizando:

Cliente = ficha**.**nome

DECLARAÇÃO

Para termos acesso a um arquivo dentro do algoritmo, é necessário que ele esteja declarado e aberto. Ao final ou quando necessário, deveremos fechar o arquivo. A declaração de um arquivo é feita da seguinte forma:

Declare identificador **arquivo** organização **de** nome-registro **chave** campo

Onde:

Declare	palavra chave.
identificador	nome do arquivo usado no algoritmo.
arquivo	palavra chave.
organização	seqüencial ou randômica.
de	palavra chave.
nome-registro	nome do tipo de registro declarado, contido no arquivo.
Chave	palavra chave
campo	nome do campo utilizado como chave de identificação do registro

Exemplo:

Declare clientes **arquivo** randômico **de** ficha **chave** código

Neste exemplo, temos que: o nome do arquivo é clientes organizado de forma randômica, com registros do tipo ficha, utilizando como identificador de registros o campo código.

ABERTURA E FECHAMENTO DE ARQUIVO

No momento que declaramos um arquivo, estamos identificando os campos à estrutura de dados, para associá-la ao arquivo físico basta utilizar o comando de abertura:

Abra identificador utilização

Onde:

Abra	palavra-chave.
identificador	nome do arquivo usado no algoritmo.
utilização	especifica se o arquivo será utilizado para leitura ou escrita.

Caso o arquivo que está sendo aberto não exista, automaticamente ele será criado, isto é, será criado um arquivo vazio com as características determinadas na declaração.

Em arquivo de acesso randômico, não é necessário especificar a utilização, pois quando aberto ele será utilizado tanto para leitura quando gravação.

Para desfazer a associação, utilize o comando de fechamento:

Feche identificador

COMANDO DE LEITURA

Em uma organização randômica, a disposição dos registros no arquivo não é necessariamente a mesma encontrada em arquivos seqüenciais. Existem funções internas no computador que, para cada registro, será determinada a posição mediante a chave escolhida. A grande vantagem de acessar um registro de forma randômica (chave ou índice), é que não será feita a leitura de outros registros, senão o que determinamos.

O acesso a este registro é feito através do comando de leitura que possui a seguinte forma:

Leia item [chave] identificador**.**registro

Onde:

Leia	palavra-chave

item	palavra-chave
chave	valor do campo utilizado como chave, para endereçar o registro
identificador	nome do arquivo usado no algoritmo
registro	nome do registro declarado para o arquivo, usado para armazenar o registro após a leitura ou antes da escrita.

Exemplo:

Leia item [23] clientes**.**ficha {ler dados do cliente com código = 23}

Neste exemplo, após a declaração do arquivo, das variáveis e abertura do arquivo, será feita a leitura do registro que possua o campo código com igual valor ao atribuído para **item**. Em caso negativo, isto é, não existir o registro com o código desejado, será retornado uma condição de erro que poderá ser testada através de uma estrutura condicional.

```
Se ERRO Então
    Escreva "código inválido"
FimSe
```

A palavra ERRO é uma palavra-chave, usada apenas para verificação de erro de leitura em arquivos, não sendo necessária a sua declaração.

Outra palavra-chave que iremos utilizar, quando necessário, é a marca de Fim De Arquivo (FDA). Esta condição irá ocorrer quando for feita a leitura do último registro contido em um arquivo, desta forma saberemos quando encerrará a leitura de um arquivo.

Exemplo:

```
Enquanto Não FDA
    Instruções a serem realizadas até o fim do arquivo
    Leia item [ I ] arquivo.campo
    I = I +1
FimEnquanto
```

Nesta estrutura, a cada comando de leitura é feito um salto para o próximo registro.

COMANDO DE GRAVAÇÃO

Vimos que ao abrir um arquivo não existente, ele será criado como um arquivo novo, neste caso, o comando de gravação, quando for utilizado, irá incluir um novo registro, isto também ocorrerá quando gravarmos um registro em arquivo já existente e que possua chave inexistente, caso contrário, irá regravar o registro ou atualizá-lo.

Para o comando de gravação utilizaremos a seguinte forma:

Grave item [chave] identificador**.**registro

EXERCÍCIOS RESOLVIDOS

1) Uma determinada empresa decidiu dar um aumento de 15% aos funcionários que possuem salários inferiores a 1.300,00. Sabendo-se que os campos do registro são: matrícula, nome, salário e que o arquivo esta organizado por matrícula, faça um algoritmo que calcule o novo salário e imprima o registro destes funcionários. Deve ser feito também, a atualização do arquivo de funcionário.

Solução:

```
Rotina
    Declare Mat numérico
    Declare funcionário arquivo randômico de regfun chave
                                                matrícula
    Declare regfun registro  (matrícula numérico
                              nome  caractere
                              salário  numérico)
    Abra funcionário
    Mat = 1
    Leia item [ Mat ] funcionário.regfun
    Enquanto NÃO funcionário.FDA
        Se regfun.salário < 1300
            regfun.salário = regfun.salário * 1,15
            Escreva regfun
```

```
                    Grave item [ Mat ] funcionário.regfun
                FimSe
                Mat = Mat + 1
                Leia item [ Mat ] funcionário.regfun
            FimEnquanto
            Feche funcionário
        FimRotina
```

2) Crie um arquivo cujo registro é composto pelos campos: código, nome, idade, sexo. Acrescente ao novo arquivo, cem (100) registros. Logo após, leia este arquivo e imprima quantas pessoas são do sexo feminino (F).

Solução:

```
Rotina
    Declare totsexo, contador Numérico
    Declare clientes arquivo randômico de regcli chave código
    Declare regcli registro   (código numérico
                               nome caractere
                               idade numérico
                               sexo caractere)
    totsexo = 0
    Abra clientes
    Para contador = 1 Até 100
        Receba regcli
        Grave item [ contador ] clientes.regcli
    FimPara
    contador = 1
    Enquanto NÃO clientes.FDA
        Leia item [ código ] clientes.regcli     {cada operação
de leitura incrementa o valor de código}
        Se sexo = "F"
            totsexo = totsexo + 1
        FimSe
    FimEnquanto
    Escreva totsexo
    Feche clientes
FimRotina
```

EXERCÍCIOS PROPOSTOS

1) Certa empresa possui um arquivo de clientes com n registros. Este arquivo contém informações de clientes que realizaram algum tipo de compra ou apenas solicitaram informações dos produtos comercializados. Foi decidido que este arquivo será dividido em dois, um para clientes que já compraram mercadorias e outro para clientes que não compraram. Sabendo-se que os campos que compõem o registro são: código, nome, endereço, pedido, e que o campo pedido, quando conter alguma informação, indicará que o cliente já comprou algum tipo de mercadoria; crie um algoritmo que satisfaça a questão.

2) Você, na condição de programador, recebeu a seguinte tarefa: criar um arquivo que contenha registros com os seguintes campos para uma agenda: nome, endereço, UF, fone, data nascimento. Após criar o arquivo, deverá inserir 50 registros.

3) Supondo que o arquivo criado no exercício existem apenas dois estados (SP, MG) cadastrados, faça um algoritmo que imprima todos os nomes e telefones das pessoas por estado, isto é, teremos como resultado uma relação de nomes e telefones agrupados por estado.

4) Faça um algoritmo que receba o nome de um funcionário e, logo após, consulte o cadastro a procura deste funcionário, caso não encontre, envie uma mensagem, em caso afirmativo, some 500,00 ao salário, atualize o cadastro e liste todos as informações sobre o funcionário. Os campos que pertencem ao registro do cadastro de funcionário são: matrícula, nome, endereço, salário.

5) Uma instituição de pesquisa recolheu amostras em três regiões a respeito do nível de vida da população destas regiões. Cada amostra constitui um registro com os seguintes componentes: sexo, idade, salário, estado civil (S, C, D), número de dependentes, cidade. Diante de tais informações, o governo do estado decidiu enviar para as prefeituras destas regiões, verbas correspondentes as seguintes condições: 300,00 para cada pessoa do sexo feminino com salário inferior a 500,00 e que tenham dois ou mais dependentes, as pessoas que não atendem estas condições, receberão 150,00. Escreva um algoritmo que ajude ao Governo a calcular quanto deverá enviar a cada prefeitura.

6) A seção de controle de produção de uma fábrica mantém um arquivo de registros de produção por máquinas. Cada registro contém o número da máquina e número de peças produzidas em um dia. Supondo que a fábrica possua três máquinas (1, 2, 3), escreva um algoritmo que calcule a quantidade produzida por cada máquina num período de trinta dias.

7) Uma agência bancária cria diariamente um arquivo com as operações de crédito e débito. O arquivo possui as seguintes informações: número da conta, código da operação (D - débito ou C - crédito) e valor da operação. Escreva um algoritmo que ao ler o arquivo de movimento, atualize o arquivo de correntistas que possui as seguintes informações: número da conta, nome, endereço, saldo em conta.

8) Dado o arquivo cadastro contendo registro com os campos: nome, sexo, cor-de-olhos, altura, peso e data-de-nascimento. Separá-los em dois arquivos: um chamado Homens, com registros cujo campo sexo apresente o valor 1 (sexo masculino) e outro chamado Mulheres, com registros cujo campo sexo seja igual a 2. Os registros dos novos arquivos deverão possuir os seguintes campos: nome, cor-de-olhos, peso e data-de-nascimento.

9) Uma determinada empresa registra as vendas por ela efetuadas em um arquivo contendo em cada registro a descrição e valor das vendas. Elaborar um algoritmo que determine o valor médio das vendas efetuadas e imprimir a descrição da mercadoria e o valor. No final, devem ser impressos o valor médio das vendas, o nome da mercadoria que obteve a maior venda.

10) Uma Empresa de vendas mantém dois arquivos contendo o nome do vendedor e o valor de cada venda por ele efetuada. A comissão de cada vendedor é calculada segundo a tabela:

Até 5.000,00	=>	10%
Acima de 5.000,00	=>	20%

Elabore um algoritmo que determine a comissão de cada vendedor, deverão ser impressos o nome do vendedor, o total por ele vendido e sua comissão. No final, devem ser impressos o total vendido pela empresa, o valor médio das vendas da empresa, o nome do vendedor que vendeu mais e o nome do vendedor que vendeu menos.

OBS: Os arquivos estão classificados pelo código do vendedor, e o valor máximo de vendas por vendedor é 1.000.000,00.

LÓGICA DE OBJETOS

O tempo de empregar um programa grande e monolítico para executar tarefas em sistemas computacionais, está chegando ao fim com a utilização cada vez maior da lógica de objetos que trabalha com pequenos programas agrupados em várias unidades inter-relacionadas. A lógica de objetos combina as diferentes tecnologias de resolução de problemas, fazendo uma interconexão coerente de diversas regras, tais como: lógica tradicional (Se..Então), funções, redes neurais, técnica estruturada, SQL, cliente-servidor, processamento paralelo (Threads) e assim por diante.

A tecnologia baseada em objetos trabalha com modelos semelhantes ao que vemos no nosso mundo real. Em nosso mundo interagimos com diversos objetos, como por exemplo uma caneta. Esta caneta possui propriedades (cor, tamanho, forma) e operações (mudar de cor, tampar, destampar, esconder a ponta), esta mesma visão foi transferida para a computação com a Análise, Projeto e Programação baseada em objetos.

Nas formas de programação anteriores, como programação estruturada, tinhamos dados e operações tratados separadamente pelo programador. Com a orientação a objetos, os dados e as operações são agrupados em uma entidade chamada objeto.

A programação orientada a objeto (OOP) é uma extensão natural da programação estruturada. A OOP permite a utilização de boas práticas de programação, dando como resultado um código limpo de fácil compreensão e manutenção.
Uma vez criado, um objeto poderá ser utilizado por outros programadores, diminuindo o tempo de desenvolvimento do aplicativo.

A tecnologia de objetos baseia-se nos seguintes princípios, que explicaremos mais adiante:

- Classes
- Herança
- Encapsulamento
- Polimorfismo
- Métodos
- Eventos

DEFINIÇÃO DE OBJETO

Um objeto é qualquer coisa sobre a qual armazenamos dados e operações em um único pacote. Por exemplo:

- Uma nota fiscal
- Uma bicicleta
- Um crachá
- Um boleto bancário
- Um robô

Observe a tabela a seguir para o objeto Nota Fiscal:

Dados	Operações
número da nota	emitir
identificação do cliente	calcular impostos
itens vendidos	buscar itens vendidos
valor total	calcular valor total
data de emissão	imprimir

Nos referimos a um objeto pelo seu nome seguido por um dado ou operação:

NomeObjeto.Dado = valor ou variável = NomeObjeto.Dado

NomeObjeto.Operação

Utilizando o exemplo da nota fiscal temos:

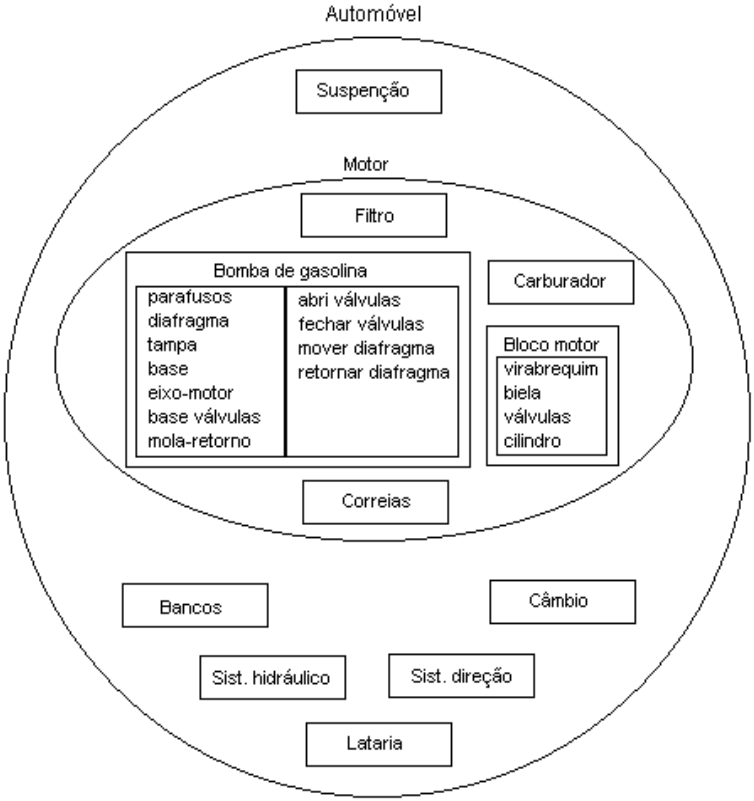
```
NotaFiscal.Emitir                    {emitir nota fiscal}
monTotal = NotaFiscal.ValorTotal     {atribui o valor total à variável
                                     monTotal}
NotaFiscal.DataEmissão = 09/07/98    {atribui à nota uma data}
```

Poderemos atribuir valores aos dados ou atribuir o valor do dado a uma variável. Já as operações não necessitam de valores, bastando apenas chamá-las para serem executadas.

Aos dados damos o nome de propriedades e às operações poderemos chamar de métodos ou eventos.

Visualizamos a utilidade dos Objetos quando queremos realizar uma viagem e precisamos construir o automóvel necessário para tal. Primeiro teríamos de montar um automóvel, tendo à disposição todas as peças separadas e um manual de montagem. Isto é análogo a escrever um programa para Windows sem utilizar objetos.

Ao invés de todo este trabalho. As peças individuais do automóvel são fabricadas e agrupadas em componentes pelas autopeças, a montadora agrupa estes componentes formando o automóvel, nós vamos até a revenda e compramos o automóvel inteiro para realizarmos a viagem. Este processo de obtenção do automóvel utiliza a abordagem de objetos, aproveitando componentes desenvolvidos por outros para se construir um sistema ou outro objeto maior.



Semelhante aos Registros um objeto contém uma coleção relacionada de elementos de vários tipos. Mas diferente de Registros, os objetos contém código - procedimentos e funções - que acionam os dados contidos nos campos do objeto, e dados - propriedades.

Um objeto deve: Ter características essenciais para o projeto;
Possuir uma representação visual;
Reagir a eventos gerados pela interação do usuário;
Realizar algumas ações simples;

Classes

Continuando com o exemplo anterior, podemos definir o automóvel como classe e cada automóvel com seu número de chassi diferente, um objeto. Temos então que o carro com chassi nº 234.345.34 é uma instância da classe automóvel.

Uma classe é uma máquina de estados finitos, sendo isolada de causa e efeito, agindo sem saber porque recebeu uma entrada ou como sua saída será utilizada.

A nossa nota fiscal pode ser considerada uma classe, e cada tipo de nota emitida, um objeto diferente. Instanciamos um objeto da seguinte forma:

NomeObjeto : **classe** (CClasse)

Exemplo:

NotaVenda : **classe** (CNotaFiscal) e;
NotaCompra : **classe** (CNotaFiscal)

Herança

Herança é a característica que o objeto tem de transferir aos seus descendentes algumas ou todas as suas características: propriedades, métodos e eventos.

Por exemplo, temos o objeto Pessoa com suas propriedades e métodos, existindo vários tipos de pessoas. O Jardineiro e o Escriturário possuem todas as características herdadas de Pessoa e em cada um são acrescentadas novas características que os distinguem. Por sua vez os objetos Auxiliar de Escritório e Contador herdam características do objeto Escriturário, acrescentando ou não utilizando algumas destas.

Encapsulamento

Encapsulamento é a propriedade que um objeto tem de ocultar o seu interior, escondendo de outras pessoas e objetos, os seus dados e operações. O objetivo disto é proteger o objeto de corrupção que pode alterar o seu funcionamento. Quando um programador utiliza um determinado objeto, ele não precisa saber como este objeto trata as informações recebidas.

O encapsulamento separa duas coisas: como o objeto se comporta e como ele é construído, pois o programador deve saber somente como o objeto reage as determinadas solicitações e como utilizá-las e nada mais. Da mesma forma que não precisamos saber o funcionamento do motor ou dos freios ABS para dirigirmos um automóvel.

O encapsulamento pode ser dos seguintes tipos:

Privado
Protegido
Público

Privado - todos os itens declarados como privados só poderão ser utilizados pelo próprio objeto. Este é o nível máximo de segurança.

Protegido - somente o próprio objeto e seus descendentes poderão utilizar os itens protegidos.

Público - são os itens visíveis a todos os outros objetos e que permitem a interação com o programador.

No exemplo da Nota Fiscal poderemos construir a nossa classe CNotaFiscal da seguinte forma:

Classe
CNotaFiscal : **classe** (CObjeto)
Privado
Calcular impostos
Buscar itens
Protegido
Número da nota
Itens

Valor Total
Calcular valor total
Público
Identificação do cliente
Data de emissão
Emitir
Imprimir
FimClasse

Polimorfismo

Polimorfismo representa a característica de diferentes objetos responderem à mesma operação de formas diferentes. Todas as pessoas andam, mas cada uma anda de uma forma diferente da outra, ou seja, a classe Pessoa possui a operação Andar, sendo que cada uma responderá de uma forma diferente.

No exemplo de Nota Fiscal, a NotaVenda herdará todas as características da CNotaFiscal ao contrário, a NotaCompra não herdará as operações de cálculo, e deverá implementar de uma forma diferente a Identificação.

OPERAÇÕES

Operações são funções ou procedimentos contidos no objeto que manipulam os seus dados, retornando ao sistema uma informação ou mudando o estado do objeto.

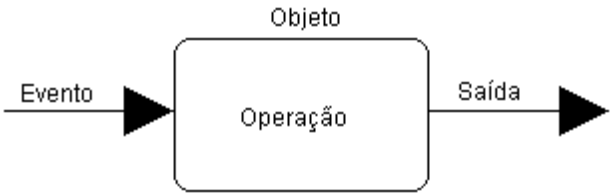
Uma operação pode ser um Método ou um Evento.

Método é toda operação quando implementada no código do programa, são procedimentos e funções construídas na estrutura de uma classe.

Um evento é um vínculo entre uma ocorrência no sistema (como uma ação de usuário ou uma mudança de foco) e um pedaço de código que responde àquela ocorrência. Evento é toda operação digna de nota que modifica significativamente o estado do objeto.

Voltando ao objeto NotaFiscal ele possui vários estados: Solicitado, Cancelado, Impresso e Arquivado, para que cada mudança de estado tenha ocorrido foi necessária uma operação.

A operação Emitir pode ser considerada um evento que muda o estado do NotaFiscal para Solicitado, já as operações de cálculo são métodos executados a partir do evento Emitir.



A saída do objeto tanto pode ser um outro evento quanto uma mudança nos estados de outros objetos.

Criando Objetos

A criação de um objeto consiste em duas partes principais: declaração e implementação. Na declaração, indicamos a forma como será o objeto, suas propriedades, métodos e eventos que ele responderá. Na parte de implementação, são escritos os códigos que dão funcionalidade ao objeto, referentes às características relacionadas na seção de declaração.

Todo objeto possui um ancestral, o ancestral primário é a classe CObjeto, ou seja, CObjeto é o Adão dos Objetos no computador.

Vamos criar uma classe de objeto CEmpregado, com quatro propriedades (Nome, Telefone, CustoHora e AlterarFunção) e dois métodos (Salário e MudarFunção). Quando for chamado o método Salário, será calculado o salário mensal do empregado, e o método MudarFunção verifica o nível de custo do empregado e altera a propriedade AlterarFunção.

```
Classe
Declaração
        CEmpregado : classe (CObjeto)
          Público
                Nome caractere
                Telefone caractere
                CustoHora numérico
                AlterarFunção lógico
                Função numérico Salário
                Procedimento MudarFunção
FimDeclaração

Implementação
        Função numérico Salário
                Resultado = CustoHora * 220
        FimFunção
        Procedimento MudarFunção
                Se CustoHora > 20 Então
                        AlterarFunção = verdadeiro
                FimSe
        FimFunção
FimClasse
```

A classe CEmpregado está criada, para criarmos um objeto desta classe utilizamos o método Criar da classe CObjeto. O exemplo abaixo mostra como utilizar um objeto no algoritmo.

```
Rotina
        Declare Maria : CEmpregado
                Cláudio : CEmpregado
                Ricardo : CEmpregado

        Maria = CEmpregado.Criar
        Cláudio = CEmpregado.Criar
        Ricardo = CEmpregado.Criar
        Receba  Maria.Nome
                Maria.Telefone
                Maria.CustoHora
        Com Cláudio
                Receba Nome, Telefone, CustoHora
```

```
        FimCom
        Com Ricardo
              Receba Nome, Telefone, CustoHora
        FimCom
        Escreva Maria.Salário, Cláudio.Salário. Ricardo.Salário
          Se Maria.MudarFunção Então
              Escreva "É necessário mudar a função da Maria"
          FimSe
FimRotina
```

No algoritmo anterior, foi introduzida a estrutura **Com ... FimCom**, que permite economia na digitação do código. Eliminando a identificação do objeto com o qual estamos alterando suas propriedades. Neste exemplo foram criados três objetos - Maria, Cláudio e Ricardo - da classe CEmpregado, foram recebidos os seus dados, emitida uma lista com os salários e verificado o nível de salário da Maria.

Agora vamos incluir o evento AumetarSalário, que acontecerá quando o Salário for menor que 100, para a classe CEmpregado.

Primeiro declaramos uma variável de controle - do tipo **notificação** - que notificará ao objeto se um evento ocorreu, está variável deve ser "vista" somente pelo objeto, depois declaramos uma propriedade dependente desta variável.

```
Classe
Declaração
        CEmpregado : classe (CObjeto)
          Público
              Nome caractere
              Telefone caractere
              CustoHora numérico
              AlterarFunção lógico
              AumentarSalário notificação Aumentar
              Função numérico Salário
              Procedimento MudarFunção
          Privado
              Aumentar notificação
FimDeclaração
```

```
Implementação
        Função numérico Salário
            Resultado = CustoHora * 220
            Se (Resultado < 100) E (atribuído (Aumentar)) Então
                AumentarSalário
            FimSe
        FimFunção
        Procedimento MudarFunção
            Se CustoHora > 20 Então
                AlterarFunção = verdadeiro
            FimSe
        FimFunção
FimClasse
```

A variável Aumentar do tipo notificação, funciona como um ponteiro do método AumentarSalário, ela contém uma mensagem que informa se existe uma rotina no algoritmo que trabalha com o evento AumentarSalário. A função **atribuído** é a encarregada de receber esta mensagem.

No caso do exemplo, se o Salário for menor que 100 e retorno da função **atribuído** for verdadeiro, então será executado o evento AumentarSalário.

Agora implementando este evento no exemplo anterior, teremos:

```
Rotina
        Declare Maria : CEmpregado
                Cláudio : CEmpregado
                Ricardo : CEmpregado
        Procedimento CEmpregado.AumentarSalário {este procedimento
será executado toda vez que o evento AumentarSalário ocorrer}
                CustoHora = CustoHora * 1,10
        FimProcedimento
        Maria = CEmpregado.Criar
        Cláudio = CEmpregado.Criar
        Ricardo = CEmpregado.Criar
        Receba  Maria.Nome
                Maria.Telefone
                Maria.CustoHora
```

```
        Com Cláudio
              Receba Nome, Telefone, CustoHora
        FimCom
        Com Ricardo
              Receba Nome, Telefone, CustoHora
        FimCom
        Escreva Maria.Salário, Cláudio.Salário. Ricardo.Salário
          Se Maria.MudarFunção Então
              Escreva "É necessário mudar a função da Maria"
          FimSe
FimRotina
```

EXERCÍCIOS RESOLVIDOS

1) Como seria um Rádio visto por um programador em objetos? Descreva algumas propriedades e métodos.

```
Classe
Declaração
        CRádio : classe (CObjeto)
        Público
              Marca caractere
              Modelo caractere
              Cor caractere
              Volume numérico
              Sintonia numérico
              Procedimento BuscaMemória (freqüência numérico)
              {envia um parâmetro}
        Protegido
              NívelRuído numérico
        Privado
              Procedimento AutoAjuste          {ajuste fino de sintonia}
FimDeclaração
```

```
Implementação
    Procedimeto BuscaMemória (freqüência numérico)
        Enquanto Sintonia <> freqüência Faça
            Sintonia = Sintonia + 5
        FimEnquanto
    FimProcedimento
    Procedimento AutoAjuste
        Declaração Temp numérico    {a variável Temp só existirá
                                     dentro deste procedimento}
        Repita
            Temp = Ruído
            Sintonia = Sintonia + 1
            Se Temp < Ruído Então
                Sintonia = Sintonia - 2
            FimSe
        Até Que Ruído < 2
    FimProcedimento
FimClasse
```

2) E um MicroSystem?

```
Classe
Declaração
    CMicroSystem : classe (CRádio)    {herda todas as características
                                       da CRádio}
    Público
        FaixaSintonia numérico
        Procedimento Equalizar (Tipo numérico)
        Grave numérico
        Agudo numérico
FimDeclaração

Implementação
    Procedimento Equalizar (Tipo numérico)
        Caso Tipo
            Seja 1 Faça Grave = Grave + 5
            Seja 2 Faça Agudos = Agudo + 5
            Seja 3 Faça Grave = Grave - 2, Agudo - 1
        FimCaso
```

FimClasse

1) Crie uma classe Automodelo, com as seguintes características:

propriedades: côr, tipo, velocidade, direção
métodos: andar, parar, acender_luzes
evento: bateu

Com uma análise mais detalhada percebemos que é necessária a criação de mais alguns componentes.

Classe
Declaração
CAutomodelo : **classe** (CObjeto)
Público
Cor **caractere**
Tipo **caractere**
Velocidade **numérico**
Direção **numérico**
Procedimento Andar
Procedimento Parar
Procedimento AcenderLuzes
Procedimento AlterarVelocidade (Delta **numérico**)
Bateu **notificação** FBateu
Privado
FBateu **notificação**
MotorLigado **lógico**
EnergiaMotor **numérico**
Luzes **lógico**
FimDeclaração

Implementação
Procedimento Andar {*movimenta o automodelo devagar*}
MotorLigado = Verdadeiro
EnergiaMotor = 2
FimProcedimento
Procedimento Parar
MotorLigado = Falso
FimProcedimento
Procedimento AcenderLuzes
Luzes = Verdadeiro

```
    Pausa 200                    {executa uma pausa de 2 segundos}
    Luzes = Falso
  FimProcedimento
  Procedimento AlterarVelocidade (Delta numérico)
    EnergiaMotor = EnergiaMotor * Delta
    Se (Velocidade = 0) E (MotorLigador) E (atribuído(FBateu)) Então
        Bateu
    FimSe
  FimProcedimento
FimClasse
```

2) Faça uma rotina de controle do automodelo criado anteriormente.

```
Rotina
  Declare Bug : CAutomodelo
  Com Bug
    Receba Cor, Tipo
  FimCom
  Procedimento Bug.Bateu     {executa este procedimento para o even
                              to Bateu}
    Bug.Parar
    Para i = 1 Até 10 Faça
        Bug.AcenderLuzes
        Pausa 400
    FimPara
  FimProcedimento
  Bug.Andar
  Bug.Direção = 0⁰
  Pausa 300
  Bug.AlterarVelocidade (1,1)
  Bug.Direção = 20⁰
    Pausa 300
  Bug.AlterarVelocidade (1)
  Bug.Parar
FimRotina
```

EXERCÍCIOS PROPOSTOS

1) Métodos podem ser considerados propriedades? Podemos substituir alguns métodos por alterações em propriedades.

2) Desenvolva um objeto semelhante a uma pessoa com três propriedades e dois métodos.

3) A partir do objeto anterior, crie um objeto Jardineiro com pelo menos mais duas propriedades e um método.

4) O Jardineiro criado anteriormente recebeu um bilhete, onde podem haver uma das três mensagens:
   a) Ir embora
   b) Regar plantas
   c) Pintar cerca

   Duas destas mensagens serão tratadas como Eventos e uma delas será desprezada pois não pertence ao Jardineiro.

5) Utilizando o polimorfismo altere a rotina de tratamento dos eventos para um outro Jardineiro que execute as tarefas anteriores.

RESOLUÇÃO DOS EXERCÍCIOS

ALGORITMO

1) **Rotina**
   Declare A, B, C, D, E, Soma, Produto, Média **numérico**
   Receba A, B, C, D, E
   Soma = A + B + C + D + E
   Produto = A * B * C * D * E
   Média = Soma / 5
   Escreva "Soma=", Soma; "Produto=", Produto; "Média=", Média
   FimRotina

2) **Rotina**
   Declare X, Y **numérico**
   Receba X
   Y = ((X ^ 2) / 3) - X + 7
   Escreva Y
   FimRotina

3) **Rotina**
   Declare A, B, C **numérico**
   Receba B, C
   A = ((B ^ 2) + (c ^ 2)) ^ 0,5 { ou, A = RAIZ ((B ^ 2) + (c ^ 2)) }
   Escreva A
   FimRotina

4) **Rotina**
   Declare A, B, Temp **numérico**
   Receba A, B
   Temp = A
   A = B
   B = Temp
   Escreva A, B
   FimRotina

5) **Rotina**
Declare Código, Salário **numérico**
Declare Nome **caractere**
Receba Código, Nome, Salário
Salário = Salário * 0,85
Escreva "Nome: ",Nome; "Salário=", Salário
FimRotina

ESTRUTURAS CONDICIONAIS

1) X = 3, Y = 7 e Z = 5

2) **Rotina**
Declare A, B, C **numérico**
Receba A, B, C
Se (A mod 2) = 0 **Então**
Escreva A
FimSe
Se (B mod 2) = 0 **Então**
Escreva B
FimSe
Se (C mod 2) = 0 **Então**
Escreva C
FimSe
FimRotina

3) **Rotina**
Declare A, B, C **numérico**
Receba A, B, C
Se (A < B) E (A < C) **Então**
Escreva A
FimSe
Se (B < A) E (B < C) **Então**
Escreva B
FimSe
Se (C < A) E (C < B) **Então**
Escreva C
FimSe
FimRotina

4) **Rotina**
Declare A, B, C **numérico**
Receba A, B, C
Se (A + B) < C **Então**
Escreva "Estes valores não formam um triângulo"
FimSe
Se (B + C) < A **Então**
Escreva "Estes valores não formam um triângulo"
FimSe
Se (C + A) < B **Então**
Escreva "Estes valores não formam um triângulo"
FimSe
FimRotina

5) **Rotina**
Declare A, B, C **numérico**
Receba A, B, C
Se (A < B) E (A < C) E (B < C) **Então**
Escreva A, B, C
FimSe
Se (A < B) E (A < C) E (B > C) **Então**
Escreva A, C, B
FimSe
Se (A > B) E (A < C) E (B < C) **Então**
Escreva B, A, C
FimSe
Se (A > B) E (A > C) E (B < C) **Então**
Escreva B, C, A
FimSe
Se (A < B) E (A > C) E (B > C) **Então**
Escreva C, A, B
FimSe
Se (A > B) E (A > C) E (B > C) **Então**
Escreva C, B, A
FimSe
FimRotina

6) **Rotina**
Declare X **numérico**
Receba X
Caso X
0 : **Escreva** “Zero”
1 : **Escreva** “Um”
2 : **Escreva** “Dois
3 : **Escreva** “Três”
4 : **Escreva** “Quatro”
5 : **Escreva** “Cinco”
FimCaso
FimRotina

7) **Rotina**
Declare X **numérico**
Declare Língua **caractere**
Receba X, “Português ou Inglês” : Língua
Se Língua = “Português” **Então**
Caso X
0 : **Escreva** “Zero”
1 : **Escreva** “Um”

5 : **Escreva** “Cinco”
FimCaso
FimSe
Se Língua = “Inglês” **Então**
Caso X
0 : **Escreva** “Zero”
1 : **Escreva** “One”

5 : **Escreva** “Five”
FimCaso
FimSe

ESTRUTURAS DE REPETIÇÃO

1)
```
Rotina
    Declare Valor, TotPar, I numérico
    TotPar = 0
    Para I = 1 Até 100 Faça
        Receba Valor
        Se Valor < 100 Então
            Valor = RAIZ( Valor )
            Escreva Valor
        SeNão
            Se (100 mod 2) = 0 Então
                Escreva "Valor par ", Valor
                TotPar = TotPar + 1
            FimSe
        FimSe
    FimPara
    Escreva "Total de números pares = ", TotPar
FimRotina
```

2)
```
Rotina
    Declare C, F  numérico
    Para F = 50 Até 150 Faça
        C = 5 * ( F - 32) / 9
        Escreva F, " = ", C
    FimPara
FimRotina
```

3)
```
Rotina
    Declare Canal, Pessoas, TotPessoas  numérico
    Declare Can2, Can4, Can5, Can13  numérico
    Can2 = 0
    Can4 = 0
    Can5 = 0
    Can13 = 0
    Repita
        Receba Canal, Pessoas
        Caso Canal
            2 : Can2 = Can2 + Pessoas
            4 : Can4 = Can4 + Pessoas
```

```
                    5 : Can5 = Can5 + Pessoas
                    13 : Can13 = Can13 + Pessoas
                FimCaso
                TotPessoas = TotPessoas + Pessoas
            Até que Canal = 0
            Can2 = Can2 / TotPessoas
            Can4 = Can4 / TotPessoas
            Can5 = Can5 / TotPessoas
            Can13 = Can13 / TotPessoas
            Escreva "Canal 2 - ", Can2; "Canal 4 - ", Can4; "Canal 5 - ",
Can5;                   "Canal 13 - ", Can13
    FimRotina
```

4)

```
    Rotina
        Declare S, Num, Den numérico
        Num = 1
        Para Den = 1 Até 50
            S = S + (Num / Den)
            Num = Num +2
        FimPara
```

5)

```
    Rotina
        Declare Total, Casa, Atual  numérico
        Total = 0           {total de grãos de milho}
        Atual = 1           {número de grãos na casa atual}
        Para Casa = 1 Até 64 Faça
            Total = Total + Atual
            Atual = Atual * 2
        FimPara
        Escreva Total
    FimRotina
```

DESVIOS

1)
```
Rotina
Declare ValorAtrasado, Prestação, TaxaJuros, Tempo numérico
Procedimento Calcular (P, TJ, T)
        ValorAtrasado = P + (P * (1 + TJ/100) ^ T)
FimProcedimento
Escreva "Entre com o valor da prestação"
Receber Prestação
Escreva "Entre com o valor da Taxa de Juros"
Receber TaxaJuros
Escreva "Entre com o tempo de atraso"
Receber Tempo
Calcular (Prestação, TaxaJuros, Tempo)
Escreva "O total a pagar é: ";ValorAtrasado
FimRotina
```

2)
```
Rotina
Declare Base, Expoente, Potência numérico
Procedimento Calcular (B, E)
        Potência = B
        Para I = 1 Até (E -1) Faça
                Potência = Potência * B
        FimPara
FimProcedimento
Escreva "Entre com a base"
Receber Base
Escreva "Entre com o expoente"
Receber Expoente
Calcular (Base, Expoente)
Escrever "A potência é: ";Potência
FimRotina
```

3) Faça um programa de calculadora com as funções de soma e subtração, que escreva no início as opções que o usuário deverá escolher para definir o operando, a seguir, peça os valores dos operandos, apresentando o resultado.

```
Rotina
    Declare A, B, Operação numérico
    Função numérico Soma(A, B)
        Resultado = A + B
    FimFunção
    Função numérico Subtração (A, B)
        Resultado = A - B
    FimFunção
    Escreva "Escolha uma das opções abaixo:"
    Escreva " 1 - Soma"
    Escreva " 2 - Subtração"
    Escreva " 3 - Sair"
    Receber Operação
    Receber A, B
    Caso Operação
        Seja 1 Faça Escreva Soma (A, B)
        Seja 2 Faça Escreva Subtração (A, B)
        SeNão FimRotina
    FimCaso
FimRotina
```

ARQUIVOS

1)
```
Rotina
    Declare Clientes arquivo randômico de RegCli chave Código
    Declare CliCompras arquivo randômico de RegCli chave
Código
    Declare CliDados arquivo randômico de RegCli chave Código
    Declare RegCli registro (Código numérico
                             Nome caractere
                             Endereço caractere
                             Pedido caractere)
    Abra Clientes, CliCompras, CliDados
    Repita
        Leia item [ Código ] Clientes.RegCli
        Se RegCli.Pedido = "" Então
            Grave item [ Código ] CliDados.RegCli
        SeNão
```

```
                    Grave item [ Código ] CliCompras.RegCli
                FimSe
            Até que Clientes.FDA
            Feche Clientes, CliCompras, CliDados
        FimRotina
```

2)
```
        Rotina
            Declare I  numérico
            Declare Agenda arquivo randômico de RegAg chave Nome
            Declare RegAg registro (Nome caractere
                                    Endereço caractere
                                    UF  caractere
                                    Fone  caractere
                                    Data-nascimento caractere)
            Abra Agenda
            Para I = 1 Até 50 Faça
                Receba RegAg
                Grave item [ Nome ] Agenda.RegAg
            FimPara
            Feche Agenda
        FimRotina
```

3)
```
        Rotina
            Declare Agenda arquivo randômico de RegAg chave Nome
            Declare RegAg registro (Nome caractere
                                    Endereço caractere
                                    UF  caractere
                                    Fone  caractere
                                    Data-nascimento caractere)
            Abra Agenda
            Escreva "Clientes de São Paulo"
            Enquanto Não Agenda.FDA Faça
                Leia item [ Nome ] Agenda.RegAg
                Se RegCli.UF = "SP"  Então
                    Escreva RegCli.Nome, RegCli.Fone
                FimSenão
            FimEnquanto
            Nome = ""          {posição inicial}
            Escreva "Clientes de Minas Gerais"
```

```
        Enquanto Não Agenda.FDA Faça
            Leia item [ Nome ] Agenda.RegAg
            Se RegCli.UF = "MG"  Então
                Escreva RegCli.Nome, RegCli.Fone
            FimSenão
        FimEnquanto
    FimRotina
```

4)
```
    Rotina
        Declare NovoNome  caractere
        Declare Funcionários arquivo randômico de RegFu chave
                                                    Matrícula
        Declare RegFu registro (Matrícula numérico
                                Nome caractere
                                Endereço caractere
                                Salário numérico)
        Abra Funcionários
        Receba NovoNome
        Enquanto Não Funcionários.FDA  Faça
            Leia item [ Matrícula ] Funcionários.RegFu
            Se RegFu.Nome = Nome Então
                RegFu.Salário = RegFu.Salário + 500
                Grave item [ Matrícula ] Funcionários.RegFu
            SeNão
                Escreva "Funcionário não encontrado"
            FimSe
        FimEnquanto
        Feche Funcionários
    FimRotina
```

5)
```
    Rotina
        Declare VerbaÁrea1, VerbaÁrea2, VerbaÁrea3  numérico
        Declare Cond lógico
        Declare População arquivo randômico de RPes  {este arquivo
                                                    não possui chave}
        Declare RPes registro   (Sexo caractere
                                 Idade numérico
                                 Salário numérico
                                 EstadoCivil caractere
```

```
                                        Dependentes numérico
                                        Cidade caractere)
        Abra População
        Enquanto Não População.FDA Faça
            Leia População.RPes {após a leitura, move para o
                                 próximo registro}
            Cond = (RPes.Sexo = F) E (RPes.Salário<500) E
            (RPes.Dep>=2)
            Se Cond Então
                Caso Rpes.Cidade
                    "Área1" : VerbaÁrea1 = VerbaÁrea1 + 300
                    "Área2" : VerbaÁrea2 = VerbaÁrea2 + 300
                    "Área3" : VerbaÁrea3 = VerbaÁrea3 + 300
                FimCaso
            SeNão
                Caso Rpes.Cidade
                    "Área1" : VerbaÁrea1 = VerbaÁrea1 + 150
                    "Área2" : VerbaÁrea2 = VerbaÁrea2 + 150
                    "Área3" : VerbaÁrea3 = VerbaÁrea3 + 150
                FimCaso
            FimSe
        FimEnquanto
        Feche População
        Escreva "Verba Área 1 = ", VerbaÁrea1; "Verba Área 2 = ",
        VerbaÁrea2;        "Verba Área 3 = ", VerbaÁrea3
    FimRotina
```

6)

```
    Rotina
        Declare DiaInicial, I numérico
        Declare Produção arquivo randômico de ProdMaq
        Declare ProdMaq registro (NumMaq numérico
                                  Peças numérico
                                  Dia numérico)
        Abra Produção
        Receba DiaInicial
        Para I = DiaInicial Até DiaInicial + 30
            Enquanto Não Produção.FDA Faça
                Leia Produção.ProdMaq
                Se ProdMaq.Dia = I Então
```

```
                    Caso ProdMaq.NumMaq
                    1 : TotMaq1 = TotMaq1 + ProdMaq.Peças
                    2 : TotMaq2 = TotMaq2 + ProdMaq.Peças
                    3 : TotMaq3 = TotMaq3 + ProdMaq.Peças
                    FimCaso
                FimSe
            FimEnquanto
        FimPara
        Feche Produção
        Escreva "Máquina 1 = ", TotMaq1; "Máquina 2 = ", TotMaq2
        Escreva "Máquina 3 = ", TotMaq3;
FimRotina
```

7)

```
Rotina
    Declare Movimento arquivo randômico de RegMov chave
                                                NConta
    Declare Conta arquivo randômico de RegConta chave NConta
    Declare RegMov registro (NConta numérico
                             CodOper lógico
                             Valor numérico)
    Declare RegConta registro (NConta numérico
                             Nome caractere
                             Endereço caractere
                             Saldo numérico)
    Abra Movimento, Conta
    Enquanto Não Movimento.FDA
        Leia item [ NConta ] Movimento.RegMov
        Leia item [ NConta ] Conta.RegConta
        Se RegMov.CodOper Então    {débito = Verdadeiro}
            NConta.Saldo = NConta.Saldo - RegMov.Valor
        SeNão
            NConta.Saldo = NConta.Saldo + RegMov.Valor
        FimSenão
        Grave item [ NConta ] Conta.RegConta
    FimEnquanto
    Feche Movimento, Conta
FimRotina
```

8) **Rotina**

Declare Maior, Média, NVendas, TotVendas **numérico**
Declare MaiorDesc **caractere**
Declare Vendas **arquivo** randômico **de** RegVend
Declare RegVend **registro** (Descrição **caractere**
Valor **numérico**)
Maior = 0
NVendas = 0
TotVendas = 0
Abra Vendas
Enquanto Não Vendas.FDA **Faça**
Leia Vendas.RegVend
Se Maior < RegVend.Valor **Então**
Maior = RegVend.Valor
MaiorDesc = RegVend.Descrição
FimSe
NVendas = NVendas + 1
TotVendas = TotVendas + RegVend.Valor
Escreva "Descrição: ", RegVend.Descrição; "Valor = ",
RegVend.Valor
FimEnquanto
Feche Vendas
Média = TotVendas / NVendas
Escreva "O produto mais vendido foi: ", MaiorDesc
Escreva "A média das vendas foi: ", Média
FimRotina

9) **Rotina**

Declare MaiorVend, MenorVend **caractere**
Declare Maior, Menor, Comissão, NVendas, TotVenda, Total **numérico**
Declare Vendedores **arquivo** randômico **de** RegPes **chave** Código
Declare Vendas **arquivo** randômico **de** RegVend
Declare RegPes **registro** (Código numérico
Nome caractere
Fone caractere)
Declare RegVend **regisro** (Código caracter
Venda numérico)

```
        Abra Vendedores, Vendas
        Total = 0                    {total das vendas da loja}
        Maior = 0                    {maior valor vendido por vendedor}
        Menor = 1000000              {menor valor vendido por vendedor}
        Enquanto Não Vendedores.FDA Faça
            Leia item [ Código ] Vendedores.RegPes
            TotVenda = 0             {total vendido por vendedor}
            Enquanto Não Vendas.FDA Faça
                Leia Vendas.RegVend
                Se RegVend.Código = RegPes.Código Então
                    TotVendas = TotVendas + RegVend.Venda
                    NVendas = NVendas + 1
                FimSe
            FimEnquanto
            Se TotVenda > 5000 Então
                Comissão = TotVenda * 0,1
            SeNão
                Comissão = TotVenda * 0,2
            FimSe
            Total = Total + TotVendas
            Escreva "Nome: ", RegPes.Nome
            Escreva "Total Vendido = ", TotVendas, "Comissão = ",
                                                    Comissão
            Se Maior < TotVendas Então
                MaiorVend = RegPes.Nome
                Maior = TotVendas
            FimSe
            Se Menor > TotVendas Então
                MenorVend = RegPes.Nome
                Menor = TotVendas
            FimSe
        FimEnquanto
        Escreva "Total Vendido = ", Total; "Média Vendas = ", Total /
                                                        NVendas
        Escreva "Vendedor que vendeu menos: ", MenorVend
        Escreva "Vendedor que vendeu mais: ", MaiorVend
FimRotina
```

OBJETOS

1) R. Métodos podem ser considerados propriedades, pois poderemos substituir alguns métodos por variações nos valores destas propriedades.

2) Classe
CPessoa : **classe** (CObjeto)
Público
Nome **caractere**
Idade **numérico**
Altura **numérico**
Procedimento Ler (mensagem **notificação**)
Privado
Procedimento Equilibrar
FimClasse

3) Classe
Declaração
CJardineiro : **classe** (CPessoa)
Público
GostarPlantas **lógico**
Visão **caractere**
Procedimento Cortar
Privado
Procedimento PodarFlores
Procedimento CortarGrama
FimDeclaração

Implementação
Procedimento Cortar
Se Visão = Flores **Então**
PodarFlores
FimSe
Se Visão = Grama Então
CortarGrama
FimSe
FimProcedimento
Procedimento PodarFlores
Ficar em pé
Escolher melhor flor

```
            Acionar alicate de corte
            Apanhar flor
        FimProcedinento
        Procedimento CortarGrama
            Ficar agachado
            Acionar tesoura de corte
            Avançar tesoura
        FimProcedimeto
FimClasse


4) Classe
Declaração
    CJardineiro : classe (CPessoa)
    Público
        GostarPlantas lógico
        Visão caractere
        Procedimento Cortar
    Privado
        Ir notificação
        Regar notificação
        Procedimento PodarFlores
        Procedimento CortarGrama
        IrEmbora notificação Ir
        RegarPlantas notificação Regar
FimDeclaração

Implementação
        Procedimento Ler (mensagem notificação)
            Caso Ler
                Seja Ir Faça Se atribuído(Ir) Então IrEmbora
                Seja Regar Faça Se atribuído (Regar) Então RegarPlantas
            FimCaso
        FimProcedimeto
FimClasse
```

5) Rotina
Declare João : CJardineiro
Rafael : CJardineiro
Procedimento João.IrEmbora
Agrupar ferramentas
Guardar todas as ferramentas
Pegar bicicleta
Pedalar até sua casa
FimProcedimento
Procedimento Rafael.IrEmbora
Virar as costas
Ir ao banheiro
Andar até o bar
FimProcedimento
João = CJardineiro.Criar
Rafael = CJardineiro.Criar

João.Ler (mensagem)
Rafael.Ler (mensagem) {*nesta ordem as ferramentas do Rafael ficarão largadas*}
FimRotina