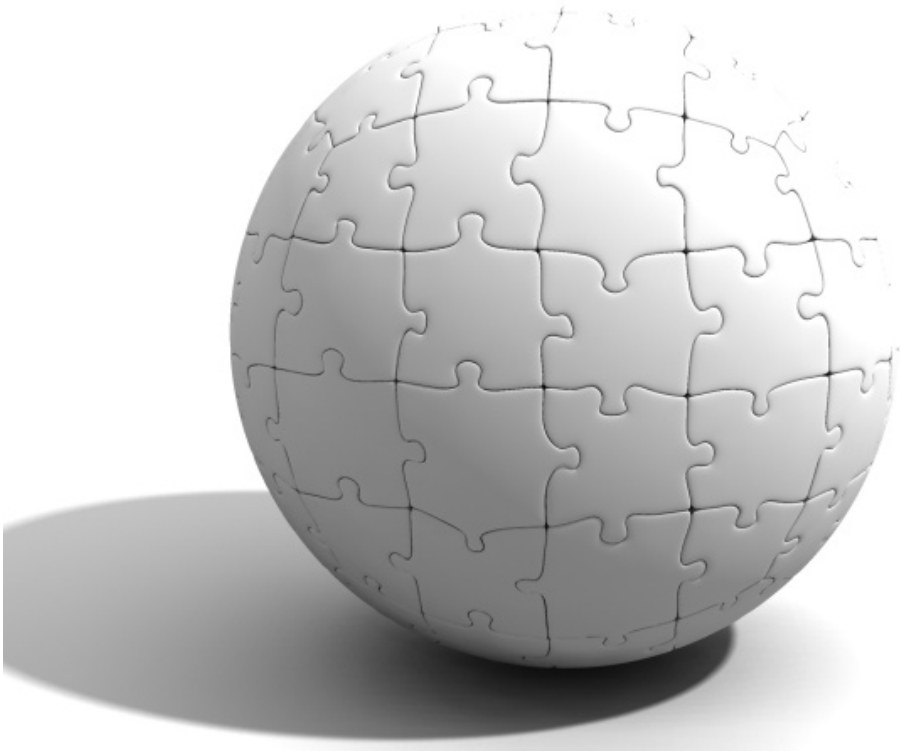
2009

Técnicas de Programação II



Professora Lucélia Oliveira

Sumário

I – REVISÃO DE LÓGICA DE PROGRAMAÇÃO

1. Faça um programa que permita entrar com a quantidade de pães e leite. Informe o valor da conta do cliente, sabendo que cada litro de leite custa R$2,50 e cada pão custa R$0,25.

2. Entrar com nome, sexo e idade de uma pessoa. Se a pessoa for do sexo masculino e tiver mais de 21 anos, exibir o nome e a mensagem: 'ACEITO'. Caso contrário, exibir o nome e a mensagem: 'NÃO ACEITO'.

3. Faça um programa que leia 50 números e informe, no final, a média aritmética dos números informados.

4. Faça um programa que leia o código de um produto do supermercado e informe que categoria o produto pertence de acordo com a relação a seguir:
   1 – limpeza;
   2 – alimentação;
   3 – bebidas;
   4 – utilidades para o lar;
   5 – têxtil;
   6 – açougue.

5. Criar um algoritmo que receba a idade e o estado civil (C – casado, S – solteiro, V viúvo, D – divorciado ou separado) de várias pessoas. Calcule e informe:
   - A quantidade de pessoas casadas;
   - A quantidade de pessoas solteiras;
   - A quantidade de viúvas com menos de 40 anos.

- A média de idades.

II - ESTRUTURA HOMOGÊNEA: VETORES

Um vetor é um arranjo de elementos armazenados na Memória Principal, um após o outro, todos com o mesmo nome. A idéia é a mesma de uma matriz linha da matemática, isto é, várias colunas e uma linha.

2	4	5	8	12	3	56	34
1	2	3	4	5	6	7	8

A [2 4 5 8 12 3 56 34]

Esse é um vetor de 8 elementos, isto é, tem 8 variáveis, todas com o mesmo nome e diferentes por sua posição dentro do arranjo que é indicada por um **índice.**

Quando se tem somente uma linha, podemos omiti-la e colocar somente a coluna.

$A_1 = 2$, $A_2 = 4$, $A_3 = 5$, $A_4 = 8$, $A_5 = 12$, $A_6 = 3$, $A_7 = 56$, $A_8 = 34$

Em algoritmos, representamos da seguinte forma:

A[1] = 2 A[2] = 4 A[3] = 5 A[4] = 8 A[5] = 12 A[6] = 3 A[7] = 56 A[8] = 34

Um algoritmo com vetor implica vários trechos para que possa funcionar corretamente. Esses trechos são **independentes.**

Trecho de Dimensionamento

Para dimensionar um vetor, usamos o seguinte comando na declaração de variáveis:

Português Estruturado	**Pascal**
Var	Var
Nome: arranjo [dimensão] de tipo;	Nome: array [dimensão] of tipo;

Onde dimensão, na prática, é o número de elementos:

No exemplo acima seria:

Português Estruturado – `A: arranjo [1..8] de Inteiro;`

Pascal - `A: array [1..8] of Integer;`

Trecho de Entrada de Dados

- normalmente, utiliza uma **estrutura de repetição.**
- se for a estrutura **Para (For),** deverá ter o **valor final** igual à **última posição** do vetor.
- se for a estrutura **Enquanto (While),** deverá ter uma variável que será incrementada e **nunca** poderá assumir um valor maior do que a **última posição** do vetor.

Português Estruturado

```
PARA L <- 1 ATÉ tamanho do vetor
FAÇA
INÍCIO
     ESCREVA (‘....’);
     LEIA (nome do vetor [ L ] );
FIM;
```

Pascal

```
FOR L := 1 TO tamanho do vetor DO
BEGIN
     WRITE (‘....’);
     READ (nome do vetor [ L ] );
END;
```

- Este trecho poderá ser incrementado com outros comandos como: estrutura de seleção (if), outro para (for), etc.

Trecho de Saída de Dados

- Normalmente, utiliza uma **estrutura de repetição.**
- Se for a estrutura **Para (For),** deverá ter o **valor final** igual à **última posição** do vetor.
- Se for a estrutura **Enquanto (While),** deverá ter uma variável que será incrementada e **nunca** poderá assumir um valor maior do que a **última posição** do vetor.

Português Estruturado

```
PARA L <- 1 até tamanho do vetor
FAÇA
INÍCIO
   ESCREVA (nome do vetor [ L ] );
FIM;
```

Pascal

```
FOR L := 1 TO tamanho do vetor DO
BEGIN
     WRITE (nome do vetor [ L ] );
END;
```

- O trecho anterior poderá ser incrementado com outros comandos;
- **Jamais** deverá aparecer um comando ***Leia (Read)*** no trecho de saída**;**
- Este trecho poderá ser incrementado com outros comandos como: estrutura de seleção (if), outro para (for), etc.

Exemplos:

1 - Criar um algoritmo que entre com dez nomes e imprima uma listagem contendo todos os nomes:

Deverá aparecer na tela o seguinte resultado:

Digite nome 1: MARIA
Digite nome 2: JOANA
Digite nome 3: MARCOS
Digite nome 4: FERNANDA
Digite nome 5: ANA
Digite nome 6: MARIANA
Digite nome 7: PEDRO
Digite nome 8: TEREZA
Digite nome 9: FILIPE
Digite nome 10: REGINA

OBS: após digitar esses dados e pressionar a tecla Enter deverá aparecer:

MARIA

JOANA

MARCOS

FERNANDA

ANA

MARIANA

PEDRO

TEREZA

FILIPE

REGINA

Português Estruturado

```
Programa lerimp;
var
L: inteiro;
nomes: arranjo [1..10] de literal;
INÍCIO
        Para L <- 1 até 10 faça
        Início
             Escreva (‘Digite nome ’, L, ‘ : ‘);
             Leia (nomes [L]);
          Fim;
          Para L <- 1 até 10 faça
           Início
              Escreva (nomes [ L]);
           Fim;
 FIM.
```

Pascal

```
Program lerimp;
var
L: Integer;
nomes: array [1..10] of string;
BEGIN
     For L := 1 to 10 do
     Begin
      Write ('Digite nome ', L, ' : ');
        Read (nomes [L]);
  End;
  For L := 1 to 10 do
  Begin
        Writeln (nomes [ L]);
End;
END.
```

Exemplo 2 – Criar um algoritmo que armazene nome e duas notas de 5 alunos e imprima uma listagem contendo nome, as duas notas e a média de cada aluno.

Português Estruturado

```
Programa imprimemedia;
Var
i: Inteiro;
nomes: arranjo [1..4] de literal;
pr1,pr2, media: arranjo [1..5] de real;
INÍCIO
          Para L <- 1 até 4 faça
          Início
               Escreva (‘Nome do aluno ‘,L, ‘ : ‘);
               Escreva(‘Nome do(a) aluno(a) ‘,i,’: ‘);
               Leia (nomes[L]);
               Escreva (‘Digite a 1ª nota: ‘);
               Leia (pr1[L]);
               Escreva (‘Digite a 2ª nota: ‘);
               Leia (pr2[L]);
               media [L] <- (pr1 [L] + pr2[L]) / 2;
          Fim;
          Escreva (‘ Relação Final’);
          Para L <- 1 até 4 faça
          Início
              Escreva (L, ‘ – ‘, nomes [L], ‘ 1ª nota: ‘, pr1[L],’ 2ª
nota: ‘, pr2[L], ‘ média: ‘,media[L]);
          Fim;
FIM.
```

Pascal

```
Program imprimemedia;
Var
i: Integer;
nomes: array [1..4] of String;
pr1,pr2, media: array [1..4] of real;
BEGIN
          For i := 1 to 4 do
          Begin
               Write (‘Nome do aluno ‘,I, ‘ : ‘);
               Write(‘Nome do(a) aluno(a) ‘,i,’: ‘);
               Read (nomes[I]);
               Write (‘Digite a 1ª nota: ‘);
               Read (pr1[I]);
               Write (‘Digite a 2ª nota: ‘);
               Read (pr2[I]);
               media [I] := (pr1 [I] + pr2[I]) / 2;
          End;
          Writeln (‘ Relação Final’);
          For i := 1 to 4 do
          Begin
              Writeln (I, ‘ – ‘, nomes [I], ‘ 1ª nota: ‘,
pr1[I]:0:1,’ 2ª nota: ‘, pr2[I] :0:1, ‘ média: ‘,media[I] :0:1);
          End;
END.
```

Veja o resultado deste programa:

Nome do aluno 1: MARCOS
Digite a 1ª nota: 10
Digite a 2ª nota: 5
Nome do aluno 2: PEDRO
Digite a 1ª nota: 8
Digite a 2ª nota: 6
Nome do aluno 3: MARIANA
Digite a 1ª nota: 5
Digite a 2ª nota: 7
Nome do aluno 4: ANA
Digite a 1ª nota: 10
Digite a 2ª nota: 9

Relação Final

1 – MARCOS	1ª nota: 10	2ª nota: 5	média: 7.5
2 – PEDRO	1ª nota: 8	2ª nota: 6	média: 7
3 - MARIANA	1ª nota: 5	2ª nota: 7	média: 6
4 – ANA	1ª nota: 10	2ª nota: 9	média: 9.5

EXERCÍCIOS

1. Armazenar 10 nomes em um vetor NOME e imprimir uma listagem numerada.

2. Armazenar 15 números inteiros em um vetor NUM e imprimir uma listagem contendo o número e uma das mensagens: par ou ímpar.

3. Armazenar 8 números em um vetor e imprimir todos os números. Ao final, teremos o total de números múltiplos de 3.

4. Armazenar nomes e notas da prova 1 e prova 2 de 15 alunos. Calcular e armazenar a média. Armazenar também a situação do aluno: AP ou RP. Imprimir uma listagem contendo nome, média e situação de cada aluno.

5. Armazenar nome e salário de 20 pessoas. Calcular e armazenar o novo salário sabendo-se que o reajuste foi de 8%. Imprimir uma listagem numerada com nome e novo salário.

6. Ler um vetor de 10 elementos e obter outro vetor, cujos componentes são o triplo dos respectivos componentes do primeiro vetor.

7. Entrar com números inteiros em um vetor A[50] e um vetor B[50]. Gerar e imprimir o veto C[50] que será a soma dos vetores A e B.

III – ORDENAÇÃO DE VETORES

Para trabalhar com vetores, é necessário obedecer três trechos distintos: trecho de dimensionamento, trecho de entrada e trecho de saída. Para ordenar um vetor é necessário acrescentar mais um trecho: o trecho de ordenação.

O trecho de ordenação irá comparar o elemento que está na 1ª posição com todos os seguintes a ele. Quando for necessário irá trocá-lo de lugar, fazendo uso **de uma variável auxiliar**. Essa operação será repetida até que tenha feito todas as comparações.

Trecho de Ordenação

```
For l:=1 to <qtde de nomes a ordenar> do
Begin
     For c:=l+1 to 5 do
     Begin
          If nome[l] > nome[c] then
          Begin
               aux := nome [l];
               nome[l]:=nome[c];
               nome[c]:=aux;
          end;
     end;
end;
```

Exemplo: Leia os nomes de 10 pessoas e exiba-os em ordem alfabética.

```
Program Exemplo1;
var
     nome:array[1..10] of string;
     l,c:integer;
     aux:string;
 Begin
     //Trecho de entrada
     for l:=1 to 10 do
     begin
          Write('Digite o nome ',l,': ');
          Read(nome[l]);
     end;

     //Trecho de ordenação
     For l:=1 to 10 do
     Begin
          For c:=l+1 to 10 do
          Begin
               If nome[l] > nome[c] then
               Begin
                    aux := nome [l];
                    nome[l]:=nome[c];
                    nome[c]:=aux;
               end;
          end;
     end;
```

```
     //trecho de saída
     for l:=1 to 10 do
     begin
          Writeln(nome[l]);
     end;
End.
```

Exemplo 2: Agora faça um programa que leia o nome e a idade de 8 pessoas e exiba os nomes e as idades ordenados pela idade.

```
Program Exemplo1;
var
     nome:array[1..8] of string;
     idade: array[1..8] of integer;
     l,c,auxi:integer;
     auxn:string;
 Begin
     //Trecho de entrada
     for l:=1 to 8 do
     begin
          Write('Digite o nome ',l,': ');
          Read(nome[l]);
          Write('Idade: ');
          Read(idade[l]);
     end;

     //Trecho de ordenação
     For l:=1 to 8 do
     Begin
          For c:=l+1 to 8 do
          Begin
               If idade[l] > idade[c] then
               Begin
                    auxn := nome [l];
                    nome[l]:=nome[c];
                    nome[c]:=auxn;

                    auxi := idade [l];
                    idade[l]:=idade[c];
                    idade[c]:=auxi;
               end;
          end;
     end;

     //trecho de saída
     for l:=1 to 8 do
     begin
          Writeln(l,'-',nome[l],' -> ',idade[l], ' anos');
     end;
End.
```

EXERCÍCIOS

1. Implementar um algoritmo que leia 10 números quaisquer e os imprima em ordem crescente.

2. Implementar um algoritmo que leia um conjunto de 50 elementos inteiros e os imprima em ordem contrária da que foi lida.

3. Faça um programa que leia o nome e as notas de 10 candidatos de um concurso e exiba o nome dos candidatos pela ordem de classificação no concurso (ou seja ordenados pelas notas).

4. Faça um programa que leia o nome e as notas de 15 candidatos de um concurso e exiba o nome dos candidatos em ordem alfabética.

5. Faça um programa que leia as quatro notas de 6 alunos, calcule a média desses alunos e informe no final a seguinte saída ordenada por ordem alfabética:
   - nome do aluno situação (aprovado ou reprovado) média

IV - MATRIZ

A estrutura da matriz é semelhante à do vetor, sendo que, pode possuir n dimensões. Desta forma para fazer referência aos elementos de uma matriz, precisaremos de tantos índices quanto for suas dimensões.

A declaração de uma matriz no Turbo Pascal é obrigatória. Por exemplo, para se declarar uma matriz MAT de uma única dimensão (vetor) composta de 50 números inteiros, seria feito da seguinte forma:

```
var
   MAT : array[1..50] of integer;
```

No caso de uma matriz bidimensional de 50 linhas e 100 colunas, composta de string de 10 caracteres, seria assim:

```
var
   MAT : array[1..50 , 1..100] of string[10];
```

Exemplo: Programa para receber via teclado os elementos da Matriz MAT, que possui 15 Linhas e 5 colunas e exibi-los no final:

```
program mat1;
var
     LINHA, COLUNA : integer;
     MAT : array[1.. 5 , 1..5] of integer;
begin
     for LINHA := 1 to 5 do
        begin
            for COLUNA := 1 to 5 do
                begin
                     writeln ('DIGITE ELEMENTO ',LINHA,'X',COLUNA,':
');
                     readln (MAT[LINHA,COLUNA]);
                end;
        end;

     for LINHA := 1 to 5 do
        begin
            for COLUNA := 1 to 5 do
                begin
                     write (MAT[LINHA,COLUNA]:4);
                end;
writeln;      writeln;
        end;

end.
```

Exemplo 2: Programa que lê o nome de 3 disciplinas e as quatro notas do alunos em cada disciplina, ao final será exibido o nome das quatro disciplinas e suas notas.

```
Program mat2;
var
mat: array [1..3,1..4] of integer;
notas: array [1..3,1..4] of real;
l,c:integer;
disc: array [1..3] of string;

begin
     for l:= 1 to 3 do
     begin
          write('Digite o nome da disciplina ',l,': ');
          read(disc[l]);
          for c:=1 to 4 do
          begin
               write('Informe a ',c,'§ nota de ',disc[l],': ');
               read(notas[l,c]);
          end;
     end;
          writeln;
          writeln ('-----------------------------------------------');
          writeln('Resultado');
          writeln;
          for l:= 1 to 3 do
          begin
               writeln('============= ',disc[l],'============== ');
               for c:=1 to 4 do
               begin
                    writeln(c,' -      ',notas[l,c]:0:2);
               end;
          end;


end.
```

EXERCÍCIOS

1. Entrar com valores em reais. Gerar e mostrar a matriz metade

2. Entrar com valores inteiros para duas matrizes 4 X 4. Gerar e mostrar a matriz SOMA.

3. Criar um algoritmo que leia os elementos de uma matriz 5 X 5 e mostre os elementos da diagonal principal. Dica: Elementos da diagonal principal linha = coluna.

4. Criar um algoritmo que leia os elementos de uma matriz 5 X 5 e mostre os elementos da abaixo da diagonal principal. Dica: Elementos da abaixo da diagonal principal linha > coluna.

5. Criar um algoritmo que leia os elementos de uma matriz 5 X 5 e mostre a soma dos elementos da diagonal principal.

6. Faça um programa que possa armazenar o nome de 5 atletas de 3 países que participarão dos jogos de verão. Informar os nomes dos países e seus respectivos jogadores.

7. Faça um Microssistema para realizar as seguintes operações:

→ Soma de duas matrizes;
→ Subtração de duas matrizes;
→ Multiplicação de duas Matrizes;
→ Mostrar os valores da diagonal principal

As regras são:
→ o Microssistema deve conter um menu de opções em que o usuário deve escolher a opção desejada;
→ As matrizes devem conter no máximo três linhas por três colunas; o usuário é quem determina as dimensões das matrizes;
→ Permitir realizar várias operações, isto é, após uma operação, voltar ao menu de opções.

V – ESTRUTURA HETEROGÊNEA: REGISTRO

Com a utilização dos vetores e matrizes, foi possível armazenar vários dados de uma só vez, porém, estes dados eram todos de um mesmo tipo; não era possível armazenar, por exemplo, dados do tipo string e inteiro dentro de uma mesma matriz. A estrutura de Registros trabalha com tipos de dados diferentes dentro de um mesmo registro, por isso é chamada de Estrutura Heterogênea.

Para armazenar os nomes e as quatro notas dos alunos de uma turma, por exemplo, era necessário um vetor que armazenasse o nome dos alunos por serem seus valores do tipo string e uma matriz para armazenar as notas por ser seus valores do tipo real. Imagine como seria mais fácil agrupar os dois tipos de dados em uma mesma estrutura. É exatamente isso que se consegue fazer com a estrutura de registro.

Exemplo de um layout de registro:

Nome.....:

Primeira Nota.....:
Segunda Nota.....:
Terceira Nota.....:
Quarta nota.....:

A declaração desse registro ficaria assim:

```
Type
   Cad_aluno = Record
         Nome: string;
         Nota1: real;
         Nota2: real;
         Nota3: real;
   End;
Var
   Aluno: cad_aluno;
```

Observe que é especificado um registro denominado cad_aluno, o qual é um conjunto de dados heterogêneos (um campo tipo string e quatro do tipo real). Desta forma é possível guardar em uma mesma estrutura vários tipos de dados distintos.

Leitura de Registros

A leitura de um registro é efetuada com a instrução **read** seguida do nome da variável registro e seu campo correspondente separado por “.” ponto. Observe os exemplos de leitura abaixo:

```
Read (aluno.nome);
Read (aluno.nota1);
Read (aluno.nota2);
```

Exemplo: Efetuar a leitura das 4 notas bimestrais de 8 alunos, apresentando no final os dados dos alunos classificados por nome.

```
Program Registro1;
type

     cad_aluno = record
          nome: string;
          nota: array [1..4] of real;
     end;
var
     aluno: array [1..8] of cad_aluno;
     l,c,i: integer;
     auxn:string;
     auxnota:real;
 Begin
     for l:=1 to 8 do
     begin
     write ('Informe o nome do aluno ',l,':');
     read(aluno[l].nome);
     for c:=1 to 4 do
     begin
          write('Infome a nota ',c,' de ',aluno[l].nome,': ');
          read(aluno[l].nota[c]);
     end;
     end;

     //Trecho de ordenação

     for l:=1 to 8 do
     begin
     for c:= 1+l to 8 do
     begin
          if aluno[l].nome > aluno[c].nome then
          begin
               auxn := aluno[l].nome;
               aluno[l].nome:= aluno[c].nome;
               aluno[c].nome:=auxn;

               for i:=1 to 4 do
               begin
               auxnota := aluno[l].nota[i];
               aluno[l].nota[i]:= aluno[c].nota[i];
               aluno[c].nota[i]:=auxnota;
               end;
          end;
     end;
     end;


//Trecho de Saída
    writeln ('        nome   1§ Nota   2§ Nota   3§ Nota   4§ Nota');
    for l:=1 to 8 do
```

```
    begin
     write(aluno[l].nome:10);
     for c:=1 to 4 do
     begin
          write (aluno[l].nota[c]:10:1);
    end;
    writeln;
    end;
End.
```

Exemplo 2: Deverá ser criado um programa que efetue a leitura de uma tabela de cargos e salários. Em seguida, o programa deverá solicitar que seja fornecido o código de um determinado cargo. Esse código deverá estar entre 1 e 10. O operador do programa poderá fazer quantas consultas desejar. Sendo o código válido, o programa deverá apresentar o cargo e o respectivo salário. Caso o código seja inválido, o programa deve avisar o operador da ocorrência. Para dar entrada de dados observe a seguinte tabela:

Código	Cargo	Salário
1	Secretária	500,00
2	Programador	1200,00
3	Faxineiro	500,00
4	Gerente Administrativo	1500,00
5	Gerente Financeiro	1400,00
6	Gerente de Pessoal	1400,00
7	Gerente de Treinamento	1400,00
8	Gerente Comercial	1400,00
9	Operador de Micro	500,00
10	Diretor	4000,00

Observe os passos que o programa deverá executar. Quanto à pesquisa será utilizado o método de pesquisa seqüencial (*Método de pesquisa seqüencial é o método mais simples de determinar a presença, ou não, de um elemento numa seqüência, é percorrê-la a partir do seu início, efetuando comparações, até que o elemento seja encontrado ou o fim da seqüência seja alcançado):

1. A tabela em questão é formada por três tipos de dados: o código como integer, o cargo como string e o salário como real. Criar então um registro neste formato;
2. Cadastrar os elementos da tabela. Para facilitar, o código será fornecido automaticamente no momento do cadastramento;
3. Criar uma estrutura de repetição para executar as consultas enquanto o operador desejar;
4. Pedir o código do cargo; se válido, apresentar o cargo e o salário;
5. Se o código for inexistente, apresentar mensagem;
6. Saber do usuário se ele deseja continuar a consulta; se sim, repetir os passos 3, 4 e 5; se não, encerrar o programa.

```
Program Registro2;
```

```
type
     dados = record
          codigo: integer;
          cargo: string;
          sal:real;
     end;
var
     tabela: array [1..10] of dados;
     i,cod: integer;
     resp:char;
     acha:boolean;
 Begin
     for i:=1 to 10 do
     begin
     tabela[i].codigo:=i;
     write ('Código.........:',i,' Cargo.............: ');
     read(tabela[i].cargo);
     write ('Salário de ',tabela[i].cargo, ' = R$....: ');
     read (tabela[i].sal);
     end;

     //Trecho de Pesquisa Seqüencial
      resp:='s';
      while (resp = 's') or (resp = 'S') do
      begin
           writeln('Digite o código que deseja pesquisar: ');
           read (cod);
           i:=1;
           acha:=false;
           while (i<=10) and (acha = false) do
           begin
                If (cod = tabela[i].codigo) then
                Begin
                     acha := true;
                     break;
               end
                else
                     acha := false;
                i:=i+1;
           end;
           if acha = true then
           Begin
                Writeln('Cargo.......: ', tabela[i].cargo);
                writeln('Salário.....: R$',tabela[i].sal:0:2);
           end
           else
                writeln('Cargi Inexistente!');
           writeln('--------------------------------');
           Writeln('Deseja continuar a pesquisa? S/N: ');
           read (resp);
     end;
 End.
```

EXERCÍCIOS

1. Considerando a necessidade de desenvolver uma agenda que contenha nomes, endereços e telefones de 10 pessoas, crie a codificação de um programa que por meio do uso de um menu de opções, execute as seguintes etapas:
    a) Cadastrar os 10 registros;
    b) Pesquisar um dos 10 registros de cada vez pelo campo nome (usar o método seqüencial);
    c) Classificar por nome os registros cadastrados;
    d) Apresentar todos os registros
    e) Sair do programa

2. Elaborar um programa que armazene o nome e a altura de 15 pessoas, por meio do uso de registros. O programa deverá ser manipulado por meio de um menu que execute as seguintes etapas:
    f) Cadastrar 10 registros;
    g) Apresentar os registros (nome e altura) das pessoas menores ou iguais a 1.5m;
    h) Apresentar os registros (nome e altura) das pessoas que sejam maiores que 1.5;
    i) Apresentar os registros (nome e altura) das pessoas que sejam maiores que 1.5 e menores que 2.0m;
    j) Apresentar todos os registros com média extraída de todas as alturas armazenadas.
    k) Sair do programa.

VI – SUB-ROTINAS

Vamos estudar os dois tipos de sub-rotinas: Funções e Procedimentos. Entre estes dois tipos de sub-rotinas existem algumas diferenças, mas, o conceito é o mesmo para ambas. O importante no uso prático destes dois tipos de sub-rotinas é distinguir as diferenças entre elas e como utiliza-las no momento mais adequado.

Procedimento (Procedure)

Um procedimento é um bloco de programa contendo início e fim e será identificado por um nome, por meio do qual será referenciado em qualquer parte do programa principal. Quando uma sub-rotina é chamada dentro do programa principal, ela é executada e ao seu término o controle de processamento retorna para a próxima linha de instrução após a linha que efetuou a chamada da sub-rotina.

Com relação à criação da rotina, será idêntica a tudo o que foi estudado sobre programação. A sintaxe em Pascal é definida da seguinte forma:

```
PROCEDURE <nome do procedimento>;
   Declaração das variáveis locais;
BEGIN
   Comandos;
END;
Ou

PROCEDURE <nome do procedimento> (x,y: tipo dos dados);
   Declaração das variáveis locais;
BEGIN
   Comandos;
END;
```

No primeiro caso, a procedure é chamada escrevendo apenas seu nome no programa principal, assim a execução é desviada até a procedure para que esta seja executada, esta sintaxe representa a procedure sem parâmetros.

No segundo caso, a procedure é chamada escrevendo o nome da procedure e seus parâmetros no programa principal, assim a execução é desviada até a procedure para que esta seja executada, esta sintaxe representa a procedure com parâmetros.

Exemplo 1: Este programa lê vários números e informa quantos são pares e quantos são ímpares (Exemplo de procedure sem parâmetros).

```
   Program procedure1;
   var
```

```
     i, p, num: integer;

Procedure par;
begin

     if num mod 2 = 0 then
          if num <> 0 then
          p:=p + 1;
end;

Procedure impar;
begin

     if num mod 2 <> 0 then
          i:=i + 1;
end;

 Begin //Programa principal
     p:=0;
     i:=0;
     num:=1;
     while num <> 0 do
     begin
          write('digite um número qualquer ou zero para encerrar');
          read (num);
          par;
          impar;
     end;
     writeln('Quantidade de números pares = ',p);
     writeln('Quantidade de números ímpares = ',i);

 End.
```

Exemplo 2: Programa que lê dois números e calcula a soma dos números lidos (Exemplo de procedure com parâmetro).

```
Program procedure2;
var
     soma,x,y:integer;

procedure somar (a, b: integer);
begin
     soma:= a + b;
     writeln('Resultado da soma = ',soma);
end;

Begin //Programa principal
     write('Digite os valores que deseja soma: ');
     read(x,y);
     somar (x,y);

end.
```

Funções

Uma function tem a mesma função de uma procedure, que é desviar a execução do programa principal para realizar uma tarefa específica, com uma única diferença: uma function sempre

retorna um valor através do seu próprio nome, por isso ao declarar uma function, declaramos também que tipo de dados esta função irá retornar.

Sintaxes:

```
FUNCTION <nome da function>: tipo de dado do valor retornado;
   Declaração das variáveis locais;
   BEGIN
        Comandos;
   END;
```

Ou

```
FUNCTION <nome da function> (x,y: tipo dos dados) : tipo de dado do valor retornado;
   Declaração das variáveis locais;
   BEGIN
        Comandos;
   END;
```

Exemplo 1: Programa que lê um valor e informa a raiz do número digitado (exemplo sem parâmetros):

```
   Program exemplo1;
   var
        calc, x: real;

   function raiz: real;
   begin
        raiz:= sqrt(x);
   end;

   Begin //Programa principal
        writeln('Digite um valor para calcular a raiz: ');
        read(x);
        if x < 0 then
             writeln('Não existe raiz quadrada de número negativo! ')
        else
        begin
             calc:=raiz;
             writeln('Raiz de ',x:0:2,' = ',calc:0:2);
        end;
   end.
```

Exemplo 2: Programa que lê um valor e informa a raiz do número digitado (exemplo com parâmetros):

```
Program exemplo2;
var
     calc, x: real;

function raiz (num:real): real;
begin
     raiz:= sqrt(num);
end;

Begin //Programa principal
     writeln('Digite um valor para calcular a raiz: ');
     read(x);
     if x < 0 then
          writeln('Não existe raiz quadrada de número negativo! ')
     else
     begin
          calc:=raiz(x);
          writeln('Raiz de ',x:0:2,' = ',calc:0:2);
     end;
end.
```

EXERCÍCIOS

1. Criar um programa que efetue o cálculo de uma prestação em atraso através de uma sub-rotina. Para tanto, utilize a fórmula PREST = VALOR + (VALOR * (TAXA/100) *TEMPO.

2. Elaborar um programa que possua uma sub-rotina que efetue e permita apresentar o somatório dos N primeiros números inteiros, definidos pelo usuário. (1 + 2 + 3 + 4 + ... + N).

3. Elaborar um programa que efetue a leitura de três valores dentro de uma procedure, implementar uma function que retorne a soma dos quadrados dos três valores. Mostrar o resultado através do programa principal.

4. Faça um programa com uma procedure que calcula o fatorial de um número, o número deve ser lido dentro de uma outra procedure.

5. Faça um programa com uma function que calcula o fatorial de um número, o número deve ser lido dentro de uma procedure.

6. Faça um programa calculadora que apresente um menu de opções no programa principal. Este menu deverá dar ao usuário a possibilidade de escolher uma entre quatro operações aritméticas. Escolhida a opção desejada, deverá ser solicitada a entrada de dois números, e processada a operação deverá ser exibido o resultado.

7. Considerando a necessidade de desenvolver uma agenda que contenha nomes, endereços e telefones de 10 pessoas, defina a estrutura de registro apropriada e a codificação de um programa que, por meio do uso do conceito de sub-rotinas, apresente um menu e suas respectivas rotinas para a execução das seguintes etapas:
    a) Cadastrar os 10 registros
    b) Pesquisar os dados de uma pessoa através do nome digitado pelo cliente
    c) Classificar por nome os registros cadastrados
    d) Apresentar todos os registros
    e) Sair do programa

8. Considerando os registros de 8 funcionários, contendo os campos matrícula, nome e salário, desenvolver um programa que utilizando conceito de sub-rotinas apresente um menu e suas respectivas rotinas para a execução das seguintes etapas:
    a) Cadastrar os 8 empregados
    b) Classificar os registros por matrícula
    c) Pesquisar um determinado funcionário por ordem de matrícula
    d) Apresentar os registros dos empregados que recebem salários abaixo de R$1000,00
    e) Apresentar os registros dos empregados que recebem salários acima de R$1000,00
    f) Sair do programa.

Funções Pré-Definidas

FUNÇÕES E PROCEDIMENTOS PRÉ-DECLARADOS PARA COTROLE DE TELA

O Turbo Pascal possui algumas funções e procedimentos pré-declarados para controle da tela:

clrscr - Comando para limpar a tela.
clreol - Este comando apaga todos os caracteres da linha que estão à direito do cursor.
delline - Apaga a linha onde está o cursor e causa scroll nas linhas seguintes de tal modo que preencha a linha deletada.
insline - Este comando funciona de modo oposto ao delline. Insere uma linha vazia na posição onde está o cursor. Provoca scroll nas linhas seguintes.
gotoxy(X,Y) - Movimenta o cursor para coluna X, linha Y da tela. A contagem das colunas e linhas começa no 1, ou seja, o canto superior esquerdo possui coordenadas (1 ,1).
lowvideo - Diminui a luminosidade dos caracteres na tela.
highvideo - Aumenta a luminosidade dos caracteres na tela.
norrnvideo - Retoma a luminosidade normal.

OUTROS RECURSOS:

delay(X) - Provoca uma pausa no processamento equivalente a X millissegundos. X deve ser definido como número inteiro.
exit - Abandona o bloco corrente. Estando numa sub-rotina, retoma ao bloco onde foi feita a sua chamada. Se usado no programa principal, termina a execução do mesmo.
Halt - Interrompe o processamento do programa e retoma ao nível de sistema operacional.
randomize - Inicializa o gerador de números aleatórios com um valor aleatório.

Exemplo1 - uso do comando gotoxy:

```
program exemplo1;
var

      I: integer;

begin

     clrscr;

     for I := 1 to 20 do

     begin
          gotoxy ( 25,1);
          writeln('** LÓGICA AVANÇADA **');
          gotoxy ( 25,3);
          writeln('-- Professora Lucélia --');
     end;
end.
```

Exemplo2- uso do comando gotoxy:

```
program Exemplo2;

begin

     clrscr;
     gotoxy(30,08); writeln('1] Incluir ');
     gotoxy(30,10); writeln('2] Alterar ');
     gotoxy(30,12); writeln('3] Excluir ');
     gotoxy(30,14); writeln('4] Pesquisar');
     gotoxy(30,16); writeln('s] Finalizar');
     gotoxy(30,20); writeln('opção ? ');

end.
```

Exemplo1 - uso do comando type

```
program exemplo3;

type
    Estacoes = (VERAO, OUTONO, INVERNO, PRIMAVERA);

var
    EST : Estacoes;

begin

     for EST := PRIMAVERA downto VERAO do
           begin

                case EST of

                     VERAO: begin
                                 Writeln('MUITO CALOR');
                                end;

                        OUTONO: begin
                                    Writeln('AS FOLHAS CAEM');
                                  end;

                        INVERNO: begin
                                     Writeln('MUlTO FRIO');
                                    end;

                         PRIMAVERA: begin
                                      Writeln('MUITAS FLORES');
                                     end;

               end;

          end;

end.
```

Outras Funções/Rotinas

Rotina : ABS( )
Função : Retorna o valor absoluto de um valor numérico.
Sintaxe : Resultado: =ABS(Valor)
Exemplo:

```
PROGRAM Teste;
VAR
     X1    : REAL;
     X2    : INTEGER;
BEGIN
     X1: =ABS( -2.3 );     ( 2.3 )
     X2: =ABS( -157 );     ( 157 )
END.
```

Rotina : CHR( )
Função : Retorna um caracter da tabela ASCII de acordo com um determinado valor numérico
Sintaxe : Resultado: =CHR(Valor)
Exemplo:

```
PROGRAM Teste;
VAR
     X1    : CHAR;
BEGIN
     X1: =CHR( 65 ); ( 'A' )
END.
```

Rotina : CLRSCR
Função : Limpa a tela de vídeo
Sintaxe : CLRSCR
Exemplo:

```
PROGRAM Teste;
USES CRT;
BEGIN
     CLRSCR;
END.
```

Rotina : CONCAT( )
Função : Concatena (Junta) uma seqüência de STRING's
Sintaxe : Resultado: =CONCAT(s1,s2,…,sn)
Exemplo:

```
PROGRAM Teste;
VAR
     s1,s2 : STRING;
BEGIN
     s1: =CONCAT('João','Mineiro');  ('João Mineiro' )
     s2: = CONCAT('ABC','DEFG','HIJ');     ('ABCDEFGHIJ')
END.
```

Rotina : COPY( )
Função : Copia n caracteres de uma STRING a partir de uma posição específica
Sintaxe : Resultado: =COPY(s1, posição , quantidade)

Exemplo:

```
PROGRAM Teste;
VAR
     s1   :STRING;
BEGIN
     s1: =COPY('ABCDEFGH',2,3);      ('BCD')
END.
```

Rotina : COS( )
Função : Retorna o cosseno de um valor numérico
Sintaxe : Resultado: = COS(Valor)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : REAL;
BEGIN
     x: =COS(10);
END.
```

Rotina : DEC( )
Função : Decrementa uma variável numérica
Sintaxe : DEC(Valor)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : INTEGER;
BEGIN
     x: =10;
     DEC(x)     ;     ( 9 )
END.
```

Rotina : DELAY( ) (CRT)
Função : Interrompe o processamento por um número especificado de milisegundos
Sintaxe : DELAY(Tempo)

Exemplo:

```
PROGRAM Teste;
USES CRT;
BEGIN
     DELAY(200);
END.
```

Rotina : DELETE( )
Função : Deleta n caracteres de uma STRING, a partir de uma posição inicial
Sintaxe : DELETE(s ,posição, quantidade)

Exemplo:

```
PROGRAM Teste;
VAR
     s    : STRING;
BEGIN
     s: ='João da Silva';
     DELETE (s,5,3); ('João Silva')
END.
```

Rotina : EXP( )
Função : Retorna “e” elevado a um determinado valor numérico
Sintaxe : Resultado: =EXP(Valor)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : REAL;
BEGIN
     x: : =EXP(10);
END.
```

Rotina : FRAC( )
Função : Retorna a parte fracionária de um valor numérico
Sintaxe : Resultado: =FRAC(Valor)

Exemplo:

```
PROGRAM Teste
VAR
     x: REAL
BEGIN
     x: =FRAC(2.345) ( 345 )
END
```

Rotina : GOTOXY( ) (CRT)
Função : Posiciona o cursor no vídeo em uma determinada coluna (x) e linha (y)
Sintaxe : GOTOXY(coluna, linha)

Exemplo:

```
PROGRAM Teste;
USES CRT;
BEGIN
     GOTOXY (10,20);
END.
```

Rotina : INC( )
Função : Incrementa uma variável numérica
Sintaxe : INC(Valor)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : INTEGER;
BEGIN
     x: =10;
     INC (x);        (11)
END.
```

Rotina : INSERT( )
Função : Insere uma STRING dentro de outra STRING a partir de uma determinada posição
Sintaxe : INSERT(STRING_Fonte, STRING_Destino, posição)

Exemplo:

```
PROGRAM Teste;
VAR
     s    : STRING;
```

```
BEGIN
     s: =‘João Silva’;
     INSERT(‘Da ’,s,6);    (‘João Da Silva’)
END.
```

Rotina : INT( )
Função : Retorna a parte inteira de um valor numérico
Sintaxe : REsultado: =INT(Valor)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : REAL;
BEGIN
     x: =INT(2.345);       (2)
END.
```

Rotina : KEYPRESSED (CRT)
Função : Retorna TRUE se uma tecla foi pressionada, FALSE caso contrário
Sintaxe : KEYPRESSED

Exemplo:

```
PROGRAM Teste;
USES CRT;
BEGIN
     REPEAT UNTIL KEYPRESSED;
END.
```

Rotina : LENGTH( )
Função : Retorna o número de caracteres de uma STRING
Sintaxe : Resultado: = LENGTH(s)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : INTEGER;
BEGIN
     x: = LENGTH(‘João’);       ( 4 )
END.
```

Rotina : LN( )
Função : Retorna o logaritmo natural de um determinado valor numérico
Sintaxe : Resultado: =LN(Valor)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : REAL;
BEGIN
     x :=EXP(LN(2)*3);     (* 2**3 ( 8 ) *)
END.
```

Rotina : ODD( )
Função : Retorna TRUE se uma determinado valor numérico FOR impar, FALSE caso contrário
Sintaxe :

Exemplo:

```
PROGRAM Teste;
VAR
     x    : Boolean;
BEGIN
     c: =odd(65);    ( TRUE )
END.
```

Rotina : ORD ( )
Função : Retorna qual o número na tabela ASCII de um determinado caracter
Sintaxe : Resultado: =ORD(Caracter)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : INTEGER;
BEGIN
     x: =ORD('A');   ( 65 )
END.
```

Rotina : PI
Função : Retorna o valor de PI
Sintaxe : Resultado: =PI

Exemplo:

```
PROGRAM Teste;
VAR
     x    : REAL;
BEGIN
     x: =PI;    ( 4.1415926535…)
END.
```

Rotina : POS( )
Função : Retorna a posição da primeira ocorrência dos caracteres de uma STRING dentro de outra STRING
Sintaxe : Resultado: =POS(Caracteres,STRING_A_SER_PROCURADA)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : INTEGER;
BEGIN
     x :=POS('ABC','DEABCDFG');      ( 3 )
END.
```

Rotina : READKEY (CRT)
Função : Faz a leitura de um caracter do teclado, não sendo necessário pressionar ENTER
Sintaxe : Resultado: =READKEY
Exemplo:

```
PROGRAM Teste;
USES CRT;
VAR
     tecla:     CHAR;
BEGIN
     Tecla :=READKEY;
END.
```

Rotina : SIN( )
Função : Retorna o seno de um valor numérico
Sintaxe : Resultado: =SIN(Valor)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : REAL;
BEGIN
     x := SIN(10);
END.
```

Rotina : SQR( )
Função : Retorna um valor numérico elevado ao quadrado
Sintaxe : Resultado: =SQR(Valor)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : INTEGER;
BEGIN
     x: =SQR(3);     ( 9 )
END.
```

Rotina : SQRT( )
Função : Retorna a raiz quadrada de um valor numérico
Sintaxe : Resultado: =str(Valor)

Exemplo:

```
PROGRAM Teste
VAR
     x: INTEGER
BEGIN
     x: =SQRT(9)     ( 3 )
END
```

Rotina : STR( )
Função : Converte um valor numérico para a sua representação em STRING
Sintaxe : STR(Valor, STRING_Resultante)

Exemplo:

```
PROGRAM Teste;
VAR
     s    : STRING;
BEGIN
     STR(2.345,s);
END.
```

Rotina : TEXTBACKGROUND() (CRT)
Função : Altera a cor de fundo nas operações de E/S
Sintaxe : TEXTBACKGROUND(Cor)

Exemplo:

```
PROGRAM Teste;
USES CRT;
BEGIN
     TEXTBACKGROUND(0);    (Preto)
     TEXTBACKGROUND(1);    (Azul)
     TEXTBACKGROUND(2);    (Verde)
     TEXTBACKGROUND(3);    (Ciano)
     TEXTBACKGROUND(4);    (Vermelho)
     TEXTBACKGROUND(5);    (Mangenta)
     TEXTBACKGROUND(6);    (Marrom)
     TEXTBACKGROUND(7);    (Cinza)
END.
```

Rotina : TEXTCOLOR( ) (CRT)
Função : Altera a cor das letras nas operações de E/S
Sintaxe : TEXTCOLOR(Cor)

Exemplo:

```
PROGRAM Teste;
USES CRT;
BEGIN
     TEXTCOLOR (0);  (Preto)
     TEXTCOLOR(1);   (Azul)
     TEXTCOLOR (2);  (Verde)
     TEXTCOLOR(3);   (Ciano)
     TEXTCOLOR(4);   (Vermelho)
     TEXTCOLOR(5);   (Mangenta)
     TEXTCOLOR(6);   (Marrom)
     TEXTCOLOR(7);   (Cinza)
END.
```

Rotina : TRUNC( )
Função : Trunca um valor REAL para um valor Inteiro
Sintaxe : Resultado: =TRUNC(Valor)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : INTEGER;
BEGIN
     x: =TRUNC(2.345);    ( 2 )
END
```

Rotina : UPCASE( )
Função : Converte um caracter minúsculo para maiúsculo
Sintaxe : Resultado: =UPCASE(Caracter)

Exemplo:

```
PROGRAM Teste;
VAR
     x    : CHAR;
BEGIN
     x :=UPCASE('a');( 'A' )
END.
```

Rotina : VAL( )
Função : Converte uma STRING para a sua representação numérica
Sintaxe : VAL(STRING_ORIGEM,VALOR_RESPOSTA,CODIGO_ERRO)

Exemplo:

```
PROGRAM Teste;
VAR
     x, e : INTEGER;
BEGIN
     VAL('12345',x,e);    ( 12345 )
END
```