CURSO DE INFORMÁTICA

Algoritmos

Ricardo José Cabeça de Souza

Parte 6

Sumário

UNIDADE II – ESTRUTURAS DE CONTROLE

2.3 ESTRUTURAS DE CONTROLE

2.3.5 Repetição – Para

A estrutura de repetição **para** tem uma natureza diferente das instruções enquanto. A execução deste laço é controlado por uma variável, obrigatoriamente um inteiro enumerável, onde as instruções são executadas um número bem definido de vezes.

Formato geral da estrutura de repetição para:

```
...
inteiro: i;
para i de valor_inicial até valor_final passo valor_passo faça
  comandos;
      ...
fim para;
...
```

A variável **i** é utilizada para controlar a execução das repetições (número de vezes que as operações são executadas).

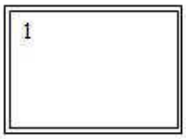
Exemplo da estrutura de repetição para:

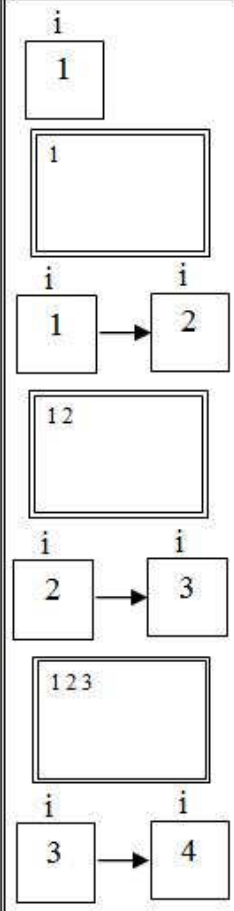
```
inicio
 inteiro: i;
  para i de 1 até 3 passo 1 faça
    imprimir (i);
  fim para;
fim.
```

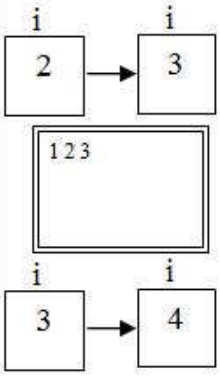
Vamos analisar agora como essa estrutura funciona, executando o teste de mesa com o exemplo mostrado acima.

Observe que o **valor_inicial** foi definido como igual a **1** e **valor_final** igual a **3**, como o **valor_passo** igual a **1**.

TESTE DE MESA		
ALGORITMO	**SIMULAÇÃO**	**COMPUTADOR VIRTUAL**
a) Declaração da variável **i**;	Na memória principal do computador é criada uma variável com o nome **i**, representada pela figura ao lado;	i 

ALGORITMO	SIMULAÇÃO	COMPUTADOR VIRTUAL
b) Execução da primeira linha (para) pela primeira vez;	Quando a estrutura é lida pela primeira vez, o valor_inicial definido na estrutura é armazenado na variável **i** e é realizado o teste para saber se já atingiu o valor_final. Como o valor_inicial é **1** e o valor_final é igual a **3**, ocorre a entrada no laço para execução da operação definida na estrutura, ou seja, a exibição do valor de **i** na tela;	i  

ALGORITMO	SIMULAÇÃO	COMPUTADOR VIRTUAL
c) Chegada no final da estrutura e retorno a linha inicial;	Após a execução da instrução no interior do laço, chegamos ao fim_para. Com a chegada ao fim_para, retornamos a leitura novamente da primeira linha, **ocorrendo o seguinte**: o valor_passo é incrementado a variável **i** (**i ← i + 1**), e é realizado novo teste no valor de **i** se chegou ao valor_final. Como o valor de **i** agora é **2** e o valor_final é igual a **3**, ocorre a entrada no laço novamente para execução da operação definida na estrutura, ou seja, a exibição do valor de **i** na tela.	

ALGORITMO	SIMULAÇÃO	COMPUTADOR VIRTUAL
d) Término do laço quando o valor de **i** ultrapassa o valor_final;	As operações de repetição são executadas enquanto o valor de **i** for menor ou igual ao valor_final. Quando **i** ultrapassar o valor_final definido na estrutura, a execução do laço é **interrompida**.	

O valor_passo apresentado no exemplo é **1**, ou seja, toda vez que a linha de instrução para é lida, o valor da variável de controle **i** aumenta em **1**, ocorrendo a instrução **i ← i + 1**. Nem sempre o valor do passo varia de **1 em 1**, sendo possível ocorrer o incremento do passo de **2 em 2** (**i ← i + 2**) ou até mesmo na forma de **decremento** (diminuição) do valor da variável, como por exemplo **passo -1** (**i ← i – 1**).

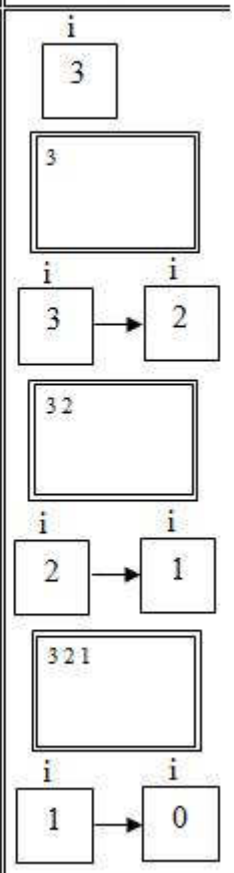
Exemplo de estrutura para com decremento da variável de controle:

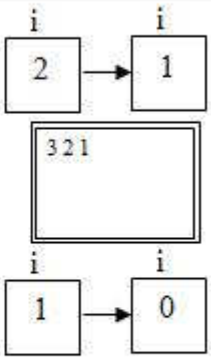
```
inicio
 inteiro: i;
   para i de 3 até 1 passo -1 faça
      imprimir (i);
   fim para;
fim.
```

Realizando o teste de mesa neste exemplo, temos:

TESTE DE MESA

ALGORITMO	SIMULAÇÃO	COMPUTADOR VIRTUAL
a) Declaração da variável **i**;	Na memória principal do computador é criada uma variável com o nome **i;**	i
b) Execução da primeira linha (para) pela primeira vez;	Quando a estrutura é lida pela primeira vez, o valor_inicial definido na estrutura é armazenado na variável **i** e é realizado o teste para saber se já atingiu o valor_final. Como o valor_inicial é **3** e o valor_final é igual a **1**, ocorre a entrada no laço para execução da operação definida na estrutura, ou seja, a exibição do valor de **i** na tela;	i 3 3

ALGORITMO	SIMULAÇÃO	COMPUTADOR VIRTUAL
c) Chegada no final da estrutura e retorno a linha inicial;	Após a execução da instrução no interior do laço, chegamos ao fim_para. Com a chegada ao fim_para, retornamos a leitura novamente da primeira linha, **ocorrendo o seguinte**: o valor_passo é decrementado a variável **i** (**i ← i - 1**), e é realizado novo teste no valor de **i** se chegou ao valor_final. Como o valor de **i** agora é **2** e o valor_final é igual a **1**, ocorre a entrada no laço novamente para execução da operação definida na estrutura, ou seja, a exibição do valor de **i** na tela.	

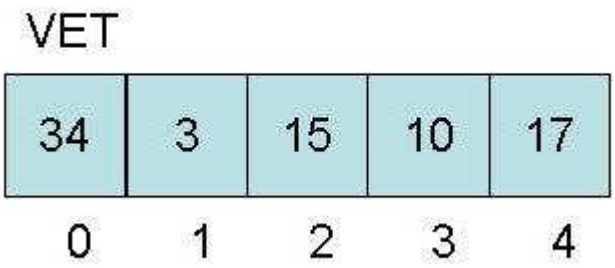
ALGORITMO	SIMULAÇÃO	COMPUTADOR VIRTUAL
d) Término do laço quando o valor de **i** ultrapassa (agora de forma invertida) o valor_final;	As operações de repetição são executadas enquanto o valor de **i** for maior ou igual ao valor_final. Quando **i** ultrapassar (inversamente) o valor_final definido na estrutura, a execução do laço é **interrompida**.	

2.3.6 Estrutura Vetor

Estamos trabalhando até o momento com variáveis onde você armazena um único valor, seja ele real, inteiro ou caractere. Contudo, existem algumas situações em que se precisa referenciar um grupo de variáveis do mesmo tipo usando um único nome simbólico. Esse tipo de variável consistindo em um conjunto de elementos de mesmo tipo sendo referenciados por uma única variável é chamado **vetor**.

A figura 2.1 representa um vetor do tipo inteiro com valores armazenados em cada posição do vetor e essas posições são identificadas por um índice, variando de 0 até 4.

Figura 2.1 - Vetor



Para usar esse tipo de estrutura, inicialmente precisamos declarar o vetor. A **declaração do vetor** segue a mesma regra para declaração de variável, diferenciando dos demais tipos pela inclusão do número de posições a serem utilizadas no vetor, como a seguir:

Declaração do vetor mostrado na figura 2.1:

inteiro: VET[5];

Identificamos na declaração o tipo de dados possível de ser armazenado em cada posição do vetor (no exemplo, inteiro), definimos também um nome válido para o vetor (VET) e o número de posições a serem utilizadas no vetor (5). Cuidado para não confundir o número de posições do vetor com os valores atribuídos aos índices do vetor, usados para identificar a posição de cada elemento do vetor.

Para realizar o armazenamento de dados do vetor (entrada), com valores digitados pelo usuário, utilizamos a estrutura de **repetição para**,

fazendo com que os valores da estrutura de repetição coincidam com os valores dos índices do vetor.

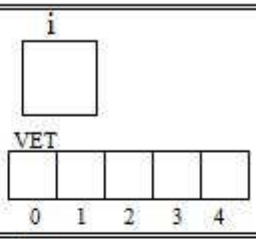
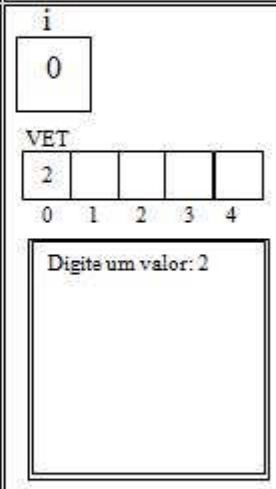
Exemplo de estrutura para entrada de dados em variáveis tipo vetor:

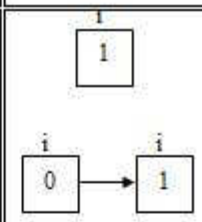
```
...
inteiro: i, VET[5];
Para i de 0 até 4 passo 1 faça
    imprimir ("Digite um valor:");
    ler ( VET[i]);
fim para;
...
```

Observe que obrigatoriamente necessitamos definir uma variável auxiliar (**i**) usada para controlar a estrutura de repetição e também identificar o local (posição) a ser armazenado o valor digitado.

Vamos realizar o teste de mesa na estrutura mostrada no exemplo.

TESTE DE MESA

ALGORITMO	SIMULAÇÃO	COMPUTADOR VIRTUAL
a) Declaração do vetor e da da variável **i**;	Na memória principal do computador é criada uma variável com o nome **i;**	
b) Leitura da estrutura **para**;	Com a primeira leitura, o valor de i recebe o valor inicial (0). Como o valor_final é 4, os comando dentro do laço são executados; O primeiro comando é executado, exibindo o texto na tela; Após a digitação do valor pelo usuário, a informação é lida e armazenada na variável VET na posição i (0);	

ALGORITMO	SIMULAÇÃO	COMPUTADOR VIRTUAL
c) Chegada ao fim_para e incremento da variável **i**;	Chegando ao fim_para, retorna e faz a leitura novamente da estrutura para, incrementando o valor de i (i ← i+1) e realizando o teste do valor_final;	
d) Leitura novamente da estrutura **para**;	Os comandos dentro do laço são executados novamente, exibindo o texto na tela; Após a digitação do valor pelo usuário, a informação é lida e armazenada na variável **VET** na posição **i** (1); Essas operações são repetidas até que o valor de **i** ultrapasse o limite estabelecido;	

ALGORITMO	SIMULAÇÃO	COMPUTADOR VIRTUAL
		

Para a exibição dos dados do vetor (saída), executamos operação semelhante a entrada de dados, como a seguir:

```
...
 inteiro: i, VET[5];
 Para i de 0 até 4 passo 1 faça
    imprimir ("Digite um valor:");
    ler ( VET[i]);
 fim para;
 //Saída de dados
 Para i de 0 até 4 passo 1 faça
    imprimir ( VET[i]);
 fim para;
...
```

É possível a realização de diversas operações com estruturas do tipo vetor. Para exemplificar, vamos realizar as seguintes operações: cálculo

da soma dos elementos do vetor, cálculo da média dos elementos do vetor e identificação de quantos valores ímpares existe no vetor.

Para realização da **soma**, criamos uma variável para acumular o resultado da soma, atribuindo um valor inicial igual a zero. Em seguida, dentro do laço, inserimos uma estrutura para realizar a acumulação dos valores armazenados. Essa estrutura é chamada **acumulador**.

```
...
inteiro: i, VET[5], S;
S ← 0;
Para i de 0 até 4 passo 1 faça
    imprimir ("Digite um valor:");
    ler ( VET[i]);
    S ← S + VET[i];
fim para;
//Saída de dados
Para i de 0 até 4 passo 1 faça
    imprimir ( VET[i]);
fim para;
//Exibição da soma
imprimir("Soma=",S);
...
```

Realizada a soma, podemos facilmente agora realizar a próxima operação, o cálculo da **média** dos valores do vetor, pois a média nada mais é do que a soma dividida pela quantidade de elementos do vetor. A soma já foi calculada no item anterior e o número de elementos do vetor já é conhecido, 5 elementos.

```
...
 inteiro: i, VET[5], S;
 S ← 0;
 Para i de 0 até 4 passo 1 faça
    imprimir ("Digite um valor:");
    ler ( VET[i]);
    S ← S + VET[i];
 fim para;
 //Saída de dados
 Para i de 0 até 4 passo 1 faça
    imprimir ( VET[i]);
 fim para;
 //Exibição da soma
 imprimir("Soma=",S);
 //Cálculo e Exibição da Média
 imprimir("Média=",S/5);
 ...
```

A próxima operação é a verificação de **quantos valores ímpares existem no vetor**. Para realizar essa tarefa, preciso analisar cada posição do vetor e verificar se o valor armazenado é um número ímpar. Se isso for verdade, preciso de uma estrutura para contar cada valor identificado como sendo ímpar. Esse tipo de estrutura é chamado **contador**, devendo, da mesma forma que o acumulador, ser declarado e atribuído valor inicial igual a zero.

```
inicio
 inteiro: i, VET[5], S,CONT;
 S ← 0;
 CONT←0;
 Para i de 0 até 4 passo 1 faça
    imprimir ("Digite um valor:");
    ler ( VET[i]);
    S ← S + VET[i];
    Se (VET[i] mod 2 = 1 )
       então
         CONT ← CONT + 1;
    fim se;
 fim para;
 //Saída de dados
 Para i de 0 até 4 passo 1 faça
    imprimir ( VET[i]);
 fim para;
 //Exibição da Soma
 imprimir ("Soma=",S);
 //Cálculo e Exibição da média
 imprimir ("Media=",S/5);
 //Identificação dos ímpares
 imprimir ("Total de Ímpares=",CONT);
 fim.
```

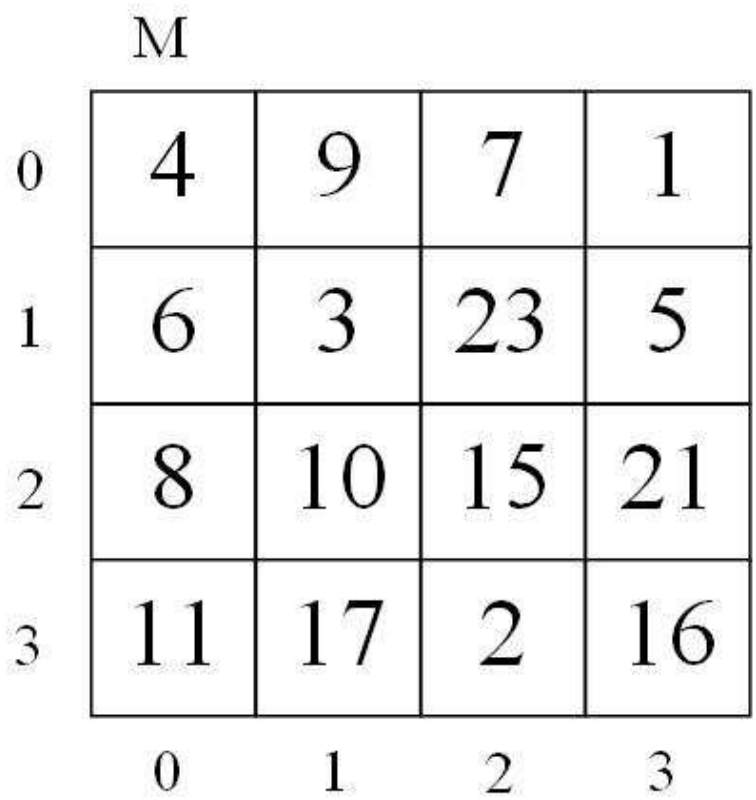
Agora temos o algoritmo completo, com todas as operações solicitadas executadas passo a passo, com a apresentação do resultado no final do algoritmo.

2.3.7 Estrutura Matriz

A estrutura de dados tipo **matriz** nada mais é do que um vetor com mais de uma dimensão, apresentando agora não somente uma linha, como no vetor, mas várias linhas e várias colunas. Podemos dizer que um vetor é unidimensional e matriz é multidimensional, por poder apresentar várias dimensões.

A figura 2.2 apresenta a estrutura do tipo matriz com 3 (três) linhas e 4(quatro) colunas.

Figura 2.2 - Matriz



A identificação da posição de cada elemento na matriz é realizada através de dois valores: o valor da posição da linha e da posição da coluna onde se encontra o elemento. Por exemplo, o valor **5** encontra-se na posição [1][2], ou seja, na linha de índice **1** e coluna de índice **2**.

Para declarar variáveis do tipo matriz, fazemos da mesma forma semelhante a declaração de variáveis do tipo vetor. Identificamos o tipo de dados que a matriz vai armazenar, inteiro, real ou caractere, definimos um nome válido para a variável e em seguida identificamos o número de linhas e o número de colunas a serem utilizadas na matriz, sempre nessa ordem (linha/coluna).

Exemplo de declaração de variáveis do tipo matriz:

inteiro: MAT[3][4];

Observe que agora, considerando que a matriz possui um índice para as linhas e um índice para as colunas, precisamos declarar duas

variáveis auxiliares, **i** e **j**, uma para controlar o índice das linhas **[i]** e outra para controlar o índice das colunas **[j]**.

inteiro: MAT[3][4],i,j;

A **entrada** de dados em variáveis do tipo matriz é realizada com o auxílio de duas estruturas **para**, uma para controlar os valores atribuídos a linha e outra para controlar os valores atribuídos a coluna, como a seguir:

```
...
inteiro: MAT[3][4],i,j;
Para i de 0 até 2 passo 1 faça
    Para j de 0 até 3 passo 1 faça
        imprimir("Digite um valor:");
        ler (MAT[i][j]);
    fim para;
fim para;
...
```

A **saída** de dados é executada de forma semelhante a entrada de dados, alterando apenas no comando a ser executado, no caso, a exibição do valor armazenado na posição identificada pelos valores de linha **[i]** e coluna **[j]**.

```
...
inteiro: MAT[3][4],i,j;
Para i de 0 até 2 passo 1 faça
    Para j de 0 até 3 passo 1 faça
        imprimir("Digite um valor:");
        ler (MAT[i][j]);
    fim para;
fim para;
Para i de 0 até 2 passo 1 faça
    Para j de 0 até 3 passo 1 faça
        imprimir(MAT[i][j]);
    fim para;
fim para;
...
```

Para completar o exemplo, vamos realizar algumas operações com a matriz. Vamos calcular a média dos valores da matriz e identificar quantos valores maiores que 10 existem na matriz.

O cálculo da **média** necessita da realização da **soma**, com a declaração de um **acumulador**, atribuindo valor inicial igual a zero e em seguida a realização da soma dentro dos laços. Será necessário também **contar** quantos elementos existem na matriz e para isso utilizaremos um **contador** para essa tarefa. Após a soma, fora do laço, o cálculo da média é realizado e apresentado o resultado.

```
...
inteiro: MAT[3][4],i,j,S,CONT;
S ← 0;
CONT ←0;
Para i de 0 até 2 passo 1 faça
    Para j de 0 até 3 passo 1 faça
        imprimir("Digite um valor:");
        ler (MAT[i][j]);
        //Cálculo da soma
        S←S+MAT[i][j];
        //Contagem dos valores lidos
        CONT←CONT+1;
    fim para;
fim para;
//Exibição dos valores da Matriz
Para i de 0 até 2 passo 1 faça
    Para j de 0 até 3 passo 1 faça
        imprimir(MAT[i][j]);
    fim para;
fim para;
//Cálculo e Exibição da Média
imprimir ("Média=",S/CONT);
...
```

A última operação solicitada é para realizar a **verificação de quantos valores maiores que 10 existem na matriz**. Para realizar essa tarefa, será necessário verificar em cada posição da matriz (todos os MAT[i][i])

se o valor ali armazenado é maior que 10. Caso seja verdadeiro, utilizamos um **contador** para realizar a contagem de cada elemento válido.

```
inicio
 inteiro: MAT[3][4],i,j,S,CONT,C10;
 S ← 0;
CONT ←0;
C10←0;
Para i de 0 até 2 passo 1 faça
   Para j de 0 até 3 passo 1 faça
      imprimir("Digite um valor:");
      ler (MAT[i][j]);
      //Cálculo da soma
      S←S+MAT[i][j];
      //Contagem dos valores lidos
      CONT←CONT+1;
      //Verificação se o valor é maior que 10
      Se (MAT[i][j] > 10 )
        então
           C10←C10+1;
      fim se;
   fim para;
fim para;
//Exibição dos valores da Matriz
Para i de 0 até 2 passo 1 faça
   Para j de 0 até 3 passo 1 faça
      imprimir(MAT[i][j]);
   fim para;
fim para;
//Cálculo e Exibição da Média
imprimir ("Média=",S/CONT);
//Exibição do resultado da verificação
imprimir ("Total valores maiores que 10=",C10);
fim.
```

Observe que em um mesmo laço, várias operações podem ser realizadas. No primeiro laço realizamos a entrada de dados, o cálculo da soma dos valores da matriz, a verificação da quantidade de elementos da matriz e a verificação de quantos elementos maiores que 10 existem na matriz. Se você desejar, pode colocar cada operação em uma estrutura de repetição separada para realizar cada umas das operações, após a entrada de dados.

2.4 EXERCÍCIOS RESOLVIDOS

a) Sendo H = 1 + 1/2 + 1/3 + 1/4 + ... + 1/N, fazer um algoritmo para gerar o número H.

```
inicio
inteiro: N;
real: H;
H ← 0;

faça
  imprimir("Digite número:");
  ler(N);
enquanto (N<=0);

enquanto (N>=1) faça
  H← H + 1/N;
  N← N – 1;
fim enquanto;

imprimir("H=",H);
fim.
```

b) Faça um algoritmo para calcular o imposto de renda de um grupo de 10 contribuintes, considerando que:

- os dados de cada contribuinte (CIC, número de dependentes e renda bruta anual) serão fornecidos pelo usuário via teclado;
- para cada contribuinte será feito um abatimento de R$ 600 por dependente;
- a renda líquida é obtida diminuindo-se o abatimento com os dependentes da renda bruta anual;
- para saber quanto o contribuinte deve pagar de imposto, utiliza-se à tabela abaixo:

RENDA LÍQUIDA	**IMPOSTO**
até R$ 1000	0

de R$ 1001 a R$ 5000 15%

acima de R$ 5000 25%

- o programa deverá imprimir, para cada contribuinte, o número do CIC e o imposto a ser pago;

- ao final, o programa deverá imprimir o total do imposto arrecadado pela Receita Federal e o número de contribuintes isentos;

```
INICIO
 Inteiro: X, CPF, ND, I;
 Real: RB, IMP, RL, SOMA;
 I← 0;
 X← 0;
 SOMA← 0;
   Enquanto (X<10) faça
        Imprimir ("Informe o CPF:");
        Ler(CPF);
        Imprimir ("Número de Dependentes:");
        Ler(ND);
        Imprimir ("Renda Bruta:");
        Ler (RB);
        RL ← RB - ( NP * 600 );
        Se ( RL <= 1000 )
                Então
                   I ← I + 1;
                   IMP← 0;
                Senão
                   Se (RL > 5000)
                         Então
                            IMP← RL * 0,25;
                         Senão
                            IMP← RL * 0,15;
                   Fim se;
        Fim se;

        SOMA ← SOMA + IMP;

        Se (IMP = 0)
          Então
             Imprimir ("Contribuinte CPF: ",CPF, "ISENTO");

          Senão
             Imprimir ("Contribuinte CPF: ", CPF, "Pagar Imposto: R$",IMP);
        Fim se;
   Fim enquanto;

   Imprimir ("Total de Imposto Recebido: R$", SOMA);
   Imprimir ("Total de Isentos:", I);

FIM.
```

c) Faça um algoritmo para ler um vetor de reais com 20 posições, e:

- Exibir o vetor lido;
- Calcular e exibir a soma dos elementos;
- Verificar e informar o maior elemento do vetor.

```
inicio
 real: V[20],S,AUX;
 inteiro: i;
 S ← 0;
 Para i de 0 até 19 passo 1 faça
    imprimir("Digite um valor:");
    ler(V[i]);
    S←S+V[i];
 fim para;
 AUX ←V[0];
 Para i de 0 até 19 passo 1 faça
    imprimir(V[i]);
    Se (V[i] > AUX)
       então
          AUX←V[i];
    fim se;
 fim para;
 imprimir("Soma=",S);
fim.
```

d) Faça um algoritmo para ler uma matriz de inteiros 3x3 e:

- Mostrar os elementos da matriz;
- Calcular e exibir a média dos elementos da diagonal principal;
- Calcular e exibir a soma da linha 0;

```
inicio
   inteiro: M[3][3],i,j,S,CDP,SL0;
   S←0;
   CDP←0;
   SL0←0;
   Para i de 0 até 2 passo 1 faça
      Para j de 0 até 2 passo 1 faça
         imprimir("Digite um valor:");
         ler(M[i][j]);
         //Cálculo da soma dos elementos da diagonal principal
         Se( i = j )
            então
               S←S+M[i][j];
               //Contagem dos elementos da diagonal principal
               CDP←CDP+1;
         fim se;
         //Cálculo da soma da linha 0
         Se ( i = 0)
            então
               SL0←SL0+M[i][j];
         fim se;
      fim para;
   fim para;
   //Exibição da Matriz
   Para i de 0 até 2 passo 1 faça
      Para j de 0 até 2 passo 1 faça
         imprimir(M[i][j]);
      fim para;
   fim para;
   imprimir("Média Elementos Diagonal Principal:",S/CDP);
   imprimir("Soma Elementos Linha 0:",SL0);
fim.
```

2.5 AVALIAÇÃO DE APRENDIZAGEM

a) Dado um conjunto de valores inteiros e positivos, fazer um algoritmo para determinar qual o menor valor do conjunto. Um valor -1(flag) indica o fim dos dados.

b) Fazer um algoritmo para achar o maior e o menor valor de uma série de 10 números.

c) Faça um algoritmo que implemente o **jogo da senha** (para 2 pessoas):

- o jogador 1 digita uma senha (valor inteiro entre 0 e 100) sem o conhecimento do jogador 2
- o jogador 2 tem 5 chances para descobrir a senha
- a cada tentativa do jogador 2, o programa deve avisar se o valor digitado é maior, menor ou igual à senha
- se o jogador 2 acertar a senha, o programa não deve pedir mais nenhuma tentativa

d) Faça um algoritmo para ler 100 números inteiros, calcular e imprimir:

- A média aritmética;
- O maior número lido;
- O menor número lido;
- Exibir todos os números lidos.

e) A série de **Fibonacci** é formada pela seqüência: 1,1, 2, 3, 5, 8, 13, 21, 34, 55, ... Escreva um algoritmo que gere a série de Fibonacci até o vigésimo termo.

f) Fazer um algoritmo para calcular e exibir o fatorial de N(N!).

g) Um determinado material radioativo perde metade de sua massa a cada 50 segundos. Dada a massa inicial, em gramas, fazer um

algoritmo que determine o tempo necessário para que essa massa se torne menor do que 0,5 gramas. Escreva a massa inicial, a massa final e o tempo calculado em horas, minutos e segundos.

h) Escreva um algoritmo que leia dois vetores de 10 posições e faça a multiplicação dos elementos de mesmo índice, colocando o resultado em um terceiro vetor. Mostre o vetor resultante.

i) Escreva um algoritmo que leia um vetor de 20 posições e mostre-o. Em seguida, troque o primeiro elemento com o último, o segundo com o penúltimo, o terceiro com o antepenúltimo, e assim sucessivamente. Mostre o novo vetor depois da troca.

j) Escrever um algoritmo que lê 2 vetores X(10) e Y(10) e os escreva. Crie, a seguir, um vetor Z que seja

- a união de X com Y;
- a soma entre X e Y;
- o produto entre X e Y;

Escreva o vetor Z a cada cálculo.

k) Faça um algoritmo que leia um código numérico inteiro e um vetor de 50 posições de números reais. Se o código for zero, termine o algoritmo. Se o código for 1, mostre o vetor na ordem direta. Se o código for 2, mostre o vetor na ordem inversa.

l) Faça um algoritmo que leia um vetor de 80 posições e encontre o menor valor. Mostre-o juntamente com seu número de ordem(índice).

m) Elaborar um algoritmo que lê uma matriz M(6,6) e um valor A e multiplicar a matriz M pelo valor A e colocar os valores da matriz multiplicada por A em um vetor V(36) e escrever no final o vetor V.

n) Escreva um algoritmo que lê uma matriz M(5,5) e calcula as somas:

- da linha 4 de M

- da coluna 2 de M
- da diagonal principal
- da diagonal secundária
- de todos os elementos da matriz M

Escrever essas somas e a matriz.

o) Escrever um algoritmo que lê uma matriz M(5,5) e cria 2 vetores SL(5) e SC(5) que contenham, respectivamente, as somas das linhas e das colunas de M. Escrever a matriz e os vetores criados.

p) Faça um algoritmo que calcule a média dos elementos da diagonal principal de uma matriz 10 X 10 de inteiros.

q) Na teoria dos sistemas, define-se como elemento **minimax** de uma matriz o menor elemento da linha onde se encontra o maior elemento da matriz. Escreva um algoritmo que leia uma matriz 10 X 10 de inteiros e encontre seu elemento **minimax**, mostrando também sua posição.

r) Faça um algoritmo que leia uma matriz 15 X 15 de reais e calcule a soma dos elementos da diagonal secundária.

REFERÊNCIAS BÁSICAS

[1] FARRER, Harry. et. al. **Programação estruturada de Computadores. Algoritmos estruturados**. 2 Ed. Rio de Janeiro: LTC Livros Técnicos e Científicos Editora Ltda, 1989.

[2] MARTINS, J. Pavão. **Introdução à programação usando Pascal**. Lisboa: Editora McGraw-Hill de Portugal Lda, 1994.

[3] ARAÚJO, Everton Coimbra de. **Algoritmos: Fundamentos e Prática**. 2 Ed. ampl. e atual. Florianópolis: VisualBooks Editora, 2005.

[4] GUIMARÃES, Ângelo de Moura. LAGES, Newton Alberto de Castilho**. Algoritmos e estruturas de dados**. Rio de Janeiro: Campus, 1985.

[5] SCHILDT, Herbert. **C Completo e total**. São Paulo: MAKRON BOOKS, 1997.

[6] LOUDON, Kyle. **Dominando algoritmos com C**. São Paulo: CIENCIA MODERNA COMPUTAÇÃO, 2000.

[7] LAUREANO, Marcos**. Programando em C para Linux, Unix e Windows**. Rio de Janeiro: Brasport Livros, 2005.

[8] MEDINA, Marco. FERTIG, Cristina. **Algoritmos e programação: teoria e prática**. São Paulo: NOVATEC INFORMATICA, 2005.

[9] FERREIRA, Aurélio Buarque de Holanda. **Novo Dicionário Aurélio Século XXI.** Rio de Janeiro: Nova Fronteira, 1999.

[10] INFORMAÇÃO. In: Wikipédia: a enciclopédia livre. Disponível em: <http://pt.wikipedia.org/wiki/Informa%C3%A7%C3%A3o>. Acesso em: 15/07/2009.

[11] PEREZ, Anderson Luiz Fernandes. **Linguagens de programação: sintaxe e semântica de linguagens de programação e conceitos de linguagens compiladas e interpretadas**. Disponível em <http://www.univasf.edu.br/~anderson.perez/ensino/intprog/> acesso em 16/07/2009.

[12] BARBOSA, Lisbete Madsen. **Ensino de algoritmos em cursos de computação.** São Paulo: EDUC, 2001.

[13] BIT. In: Wikipédia: a enciclopédia livre. Disponível em: <http://pt.wikipedia.org/wiki/Bit>. Acesso em: 15/07/2009.

[14] CÓDIGO-FONTE. In: Wikipédia: a enciclopédia livre. Disponível em: <http://pt.wikipedia.org/wiki/C%C3%B3digo-fonte>.

Acesso em: 15/07/2009.

[15] PI (Π). In: Wikipédia: a enciclopédia livre. Disponível em: <http://pt.wikipedia.org/wiki/Pi>. Acesso em: 17/07/2009.

[16] BUGS. In: Wikipédia: a enciclopédia livre. Disponível em: <http://pt.wikipedia.org/wiki/Bug> . Acesso em: 15/08/2009.

[17] FREEDMAN, Alan. **Dicionário de Informática**. São Paulo: Makron Books, 1995.

[18] RAIZ QUADRADA. Wikipédia a Enciclopédia Livre. Disponível em <http://pt.wikipedia.org/wiki/Raiz_quadrada> acesso em 30/07/2009.

REFERÊNCIAS COMPLEMENTARES

SCHILDT, Herbert. **C Completo e total**. São Paulo: MAKRON BOOKS, 1997.

DAMAS, Luís. **Linguagem C**. 10. ed. Rio de Janeiro: LTC, 2007.

LOUDON, Kyle. **Dominando algoritmos com C**. São Paulo: CIENCIA MODERNA COMPUTAÇÃO, 2000.

JAMSA, Kris. **Programando em C/C++: a bíblia**. São Paulo: Makron Books, 2000.

LOPES, Anita. **Introdução a programação: 500 algoritmos resolvidos**. Rio de Janeiro: Campus, 2002.

LAUREANO, Marcos. **Programando em C para Linux, Unix e Windows.** Rio de Janeiro: BRASPORT LIVROS, 2005.

DEITEL, Paul J.; DEITEL, Harvey M. **C++ como programar**. Porto Alegre: Bookman, 2001.

PINTO, Wilson Silva. **Introdução ao desenvolvimento de algoritmos e estrutura de dados**. São Paulo: Érica, 1990. KERNIGHAN, Brian W.; RITCHIE, Dennis M. **C: a linguagem de programação**. Rio de Janeiro: Campus, 2000. (005.133 K39c).

MANZANO, José Augusto N. G.; OLIVEIRA, Jayr Figueiredo. **Estudo dirigido de algoritmos**. São Paulo: Érica, 1997. SALVETTI, Dirceu Douglas; BARBOSA, Lisbete Madsen. **Algoritmos**. São Paulo: Makron Books, 1998. (005.1. S183A).

MIZRAHI, Victorine Viviane. **Treinamento em linguagem C: curso completo**. Módulo I. São Paulo: McGraw-Hill, 1990.