XEROCAR

Folha de Exercícios nº 1

MC100
Primeiro Semestre de 1993

Exercício 1 Considere as seguintes declaração de variáveis:

```
var
  A,B,C: integer;
  D,E: real;
  F,G,H: boolean;
  I: char;
```

Suponha, também, que essas variáveis tenham os seguintes valores em um dado instante:

Variável	Valor
A	7
B	−6
C	49
D	13.3
E	1.0E3
F	True
G	False
H	True
I	'A'

Qual será o valor final das seguintes expressões, *se válidas* (caso contrário, explicitar que tipo de erro, se de *sintaze* ou de *execução*, ocorrerá e qual a causa do erro)?

1. A−C mod Trunc(D)+4 (*Resp: 1*)
2. Odd(D) (*Resp: Erro de sintaze - tipo de parâmetro ilegal*)
3. Sqrt(Chr(Abs(B−A)+Trunc(E)))
4. not ((A>=B) and (D>=E) or F)
5. F and not G or not H
6. A=B and D<E
7. Chr(Pred(Ord('C')+1))
8. A div C*MaxInt+Round(D)
9. (B−MaxInt) div Trunc(E)
10. (C div A>Abs(B)) and F

Exercício 2 Execute o seguinte programa em PASCAL, isto é simule o que computador faria ao executá-lo. Mantenha tabelas com os valores de todas as variáveis utilizadas durante a execução.

```
program P1;
 var
   a,b,c,d,i,aux: integer;
begin
 Read(a,b,c,d);
 for i := 1 to 3 do
  begin
   if a>b then
    begin
     aux := a;
     a := b;
     b := aux
    end;
   if b>c then
    begin
     aux := b;
     b := c;
     c := aux
    end;
   if c>d then
    begin
     aux := c;
     c := d;
     d := aux
    end
  end;
 Write(a,b,c,d);
end. { P1 }
```

Exercício 3 O que escreve o programa a seguir?

```
program SolveIt;
 var vert: integer;
{ E agora, algumas declarações de procedimentos: }
 procedure Flat;
  var length: integer;
 begin
  for length := 1 to 10 do
     Write('*');
     Writeln;
 end { Flat };
 procedure Side;
  var space: integer;
```

```
    begin
      Write('*');
      for space := 1 to 8 do Write('␣');
      Writeln('*');
    end { Side };
  { O evento principal: }
  begin
    Flat;
    for vert := 1 to 6 do Side;
    Flat;
  end { SolveIt }.
```

Exercício 4 Escreva um procedimento que escreve uma tabela com os 15 primeiros inteiros positivos, seus quadrados e cubos. Cada linha deve se referir a um inteiro.

Exercício 5 Escreva um programa que gere uma figura similar à seguinte, mas expandida a 10 linhas e 38 colunas.

```
...*.....*...
..***....***..
.*****..*****.
**************
```

Exercício 6 Escreva dois programas, *TorreNoTabuleiro* e *RainhaNoTabuleiro*, baseado no programa *BispoNoTabuleiro* a seguir, que dá as posições possíveis para a Rainha e Torre, respectivamente, num tabuleiro de xadrez.

```
program BispoNo Tabuleiro;
  { Escreve um tabuleiro de xadrez. A posição do bispo é lida do teclado
    e seus movimentos possíveis são mostrados no tabuleiro. }
  var
    NaLinha, NaColuna: integer;                    { Posição do Bispo. }
    Linha, Coluna: integer;          { Posição do tabuleiro a ser analisada. }
begin
  Write('Posição␣do␣Bispo:␣');
  Readln(NaLinha, NaColuna);
  for Linha := 1 to 8 do
    begin
      for Coluna := 1 to 8 do
        if (Linha=NaLinha) and (Coluna=NaColuna)
          then Write('+')
          else
            if (Linha−Coluna = NaLinha−NaColuna)
               or (Linha+Coluna = NaLinha+NaColuna)
              then Write('*')
              else
                if (Linha+Coluna) mod 2=0
                  then Write('B')
                  else Write('P');
      Writeln
    end
end { BispoNoTabuleiro }.
```


Exercício 7 Escreva um procedimento que leia n ($n > 0$) pares de números, x_i, p_i, e calcule sua média ponderada:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i p_i}{\sum_{i=1}^{n} p_i}$$

Exercício 8 Escreva um procedimento que calcule a aproximação para a integral:

$$\int_0^x e^{-u^2}\,du = x - \frac{x^3}{3*1!} + \frac{x^5}{5*2!} - \frac{x^7}{7*3!} + \cdots$$

Sugestão: Interrompa o cálculo quando o k-ésimo termo (k qualquer) ficar menor (em módulo) que uma certa constante ϵ.

Exercício 9 O método de Newton para encontrar uma aproximação da raiz quadrada de um número a é descrito pelas seguintes equações:

$$\begin{aligned} x_0 &\leftarrow \frac{a}{2} \\ x_{i+1} &\leftarrow \frac{1}{2}(x_i + \frac{a}{x_i}), \qquad \text{para } i = 0, 1, 2, \ldots \end{aligned}$$

Assim, quanto mais alto o valor de i, melhor a aproximação de $\sqrt{a}$ dada por x_{i+1}.

Escreva um programa que leia o número a, calcule e escreva x_{10}.

Exercício 10 Modifique o programa do exercício 9 para encontrar x_{i+1} tal que $x_{i+1} - x_i \leq 0.01$.

Exercício 11 Suponha que voce invista seu dinheiro a juros fixos de $r\%$ ao mês. Após n meses, o seu investimento crescerá segundo a seguinte fórmula:

Número de Meses	Investimento Acumulado
1	$a + (r \times a) = a(1 + r)$
2	$a(1 + r) \times (1 + r) = a(1 + r)^2$
3	$a(1 + r)^2 \times (1 + r) = a(1 + r)^3$
$\vdots$	$\vdots$
n	$a(1 + r)^{n-1} \times (1 + r) = a(1 + r)^n$

Escreva um programa para calcular e escrever a tabela acima, dado um investimento inicial a, um número n de meses e juros de $r\%$.

Exercício 12 Escreva um procedimento que receba como parâmetro um número n e escreva a seguinte tabela de números (*Triângulo de Pascal*):

0	$\binom{0}{0}$					
1	$\binom{1}{0}$	$\binom{1}{1}$				
2	$\binom{2}{0}$	$\binom{2}{1}$	$\binom{2}{2}$			
3	$\binom{3}{0}$	$\binom{3}{1}$	$\binom{3}{2}$	$\binom{3}{3}$		
4	$\binom{4}{0}$	$\binom{4}{1}$	$\binom{4}{2}$	$\binom{4}{3}$	$\binom{4}{4}$	
⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	$\binom{n}{0}$	$\binom{n}{1}$	$\binom{n}{2}$	$\binom{n}{3}$	...	$\binom{n}{n}$
	0	1	2	3	...	n

Não é necessário escrever as linhas que separam os índices do conteúdo da tabela. O número $\binom{p}{q}$ é conhecido como *coeficiente binomial* e dá o número de combinações de p elementos q a q. É calculado da seguinte maneira:

$$\binom{p}{q} = \frac{p!}{q!(p-q)!}$$

Exercício 13 Dado um número n, seja $inv(n)$ o número que se obtém invertendo-se a ordem dos dígitos de n. Por exemplo, $inv(332) = 233$. Um número é *palíndromo* se $inv(n) = n$. Por exemplo, 34543, 1, 99. Escreva um procedimento que receba como parâmetro um número n e verifique se n é *palíndromo*, escrevendo a resposta adequada.

Exercício 14 Este programa tem várias partes:

1. Escreva um trecho de programa que teste se um dado número inteiro positivo m é *primo*, isto é, se m é maior ou igual a 2 e divisível apenas por 1 e m.
2. Usando o trecho de programa em (1), escreva um programa que, dado um número r, encontre o próximo número primo maior que r.
3. Usando (2), escreva um programa completo, com todas as declarações, que leia um número n e escreva todos os seus fatores primos e a multiplicidade com que cada um divide n. Por exemplo, para $n = 420$, a saída do seu programa deve ser algo como:

$n = 420$
Fatoração de n:

primo	multiplicidade
2	2
3	1
5	1
7	1

Exercício 15 Execute os programas a seguir e determine os valores escritos pelos comandos *Write*:

(a)

```
program P1;
 var a,b,c: integer;
 procedure P (x,y: integer; var z: integer);
 begin
   z := x+y+z; Write(x,y,z)
 end; { P }
begin
 a := 5;  b := 8;  c := 3;
 P(a,b,c); P(7,a+b+c,a); P(a*b,a div b,c);
end. { P1 }
```

(b)

```
program P2;
 var i,j,k: integer;
 procedure P (var i: integer);
 begin
   i := i+1; Write(i,j,k)
 end; { P }
 procedure Q (h: integer; var j: integer);
  var i: integer;
  procedure R;
  begin
    i := i+1
  end; { R }
 begin
  i := j;
  if h=0
    then P(j)
    else if h=1
           then P(i)
           else R;
  Write (i, j, k)
 end; { Q }
begin
 i := 0; j := 1; k := 2;
 Q(0,k); Q(1,i); Q(2,j)
end. { P2 }
```

(c)

```
program P3;
 var x,y,z: integer;
 procedure Ordena (var p,q,r: integer);
  procedure Troca (var s,t: integer);
   var aux: integer;
```

```
    begin
      aux := s; s := t; t := aux
    end; { Troca }
  begin { Ordena }
    if p>q then Troca(p,q);
    if q>r then
      begin
        Troca(q,r);
        if p>q then Troca(p,q)
      end
  end; { Ordena }
begin { P3 }
  x := 90; y := 25; z := 50;
  Ordena(x,y,z);
  Writeln(x,y,z);
  x := 6; y := 2; z := 1;
  Ordena(x,y,z);
  Writeln(x,y,z)
end. { P3 }
```

(d) Qual seria o resultado do programa anterior se *Ordena* tivesse sido declarado da seguinte maneira:

procedure *Ordena* (**var** *p,q: integer; r: integer);*

Exercício 16 O que escreve o programa a seguir, supondo que devam ser lidos os valores 2 e 3? Indique o escopo de cada variável do programa e dos procedimentos.

```
program TesteEscopo;
  var a,b,x,y: integer;
  procedure Mistura (var q,a: integer; b,r: integer);
    var x,z: integer;
  begin
    Readln(x);
    z := -x; x := x+1; q := 2*q+a; b := 1+z-2*x;
    Writeln(q,a,b,r,x,z)
  end; { Mistura }
begin
  a := 1; b := 2; x := 7; y := 11;
  Writeln(a,b,x,y);
  Mistura(a,b,x,y);
  Writeln(a,b,x,y);
  Mistura(b,a,x-y,y);
  Writeln(a,b,x,y)
end. { TesteEscopo }
```


Exercício 17 Considere o programa a seguir:

```
program P1;
  var x,y,z: integer;
  procedure p (          );
    var t: integer;
  begin
    t := i; i:= j; j := t
  end; { p }
begin
  x := 1; y := 2; z := 3;
  p(x,y); p(y,z); p(z,x);
  Writeln(x,y,z);
end. { P1 }
```

Indique os resultados que seriam impressos se o conteúdo do retângulo fosse:

1. (var i,j: integer);
2. (i: integer; var j: integer);
3. (i,j: integer)

Exercício 18 Considere o programa a seguir:

```
program A;
  var k,z: integer;
  procedure B (var y: integer; w: integer);
    var z: integer;                          { linha 4 }
  begin
    z := w*y; y := w+z div y
  end; { B }
begin
  z := 5;
  k := 3;
  B(z,k); Writeln(z,k);
  B(k,z); Writeln(z,k);
end. { A }
```

1. O que será impresso?
2. O que será impresso se for eliminada a *linha 4* do programa?

Exercício 19 Define-se o zero de uma função real $f(x)$ como um valor x_0 tal que

$$(f(x_0 - \epsilon) < 0) = (f(x_0 + \epsilon) > 0)$$

sendo ϵ escolhido arbitrariamente pequeno. Escreva uma função (ou procedimento) que encontre um zero de $f(x)$ no intervalo $a \leq x \leq b$ se a relação

$$(f(a) < 0) = (f(b) > 0)$$

for verdadeira. Suponha que $f(x)$ seja uma função previamente declarada. **Sugestão:** Use o *método da bissecção*, que consiste em dividir ao meio, repetidamente, o intervalo contendo o zero.

Exercício 20 Para calcular a integral

$$S = \int_a^b f(x)dx$$

pode-se usar a aproximação de uma soma finita de "valores amostrados":

$$S_k = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \ldots + 4f_{n-3} + 2f_{n-2} + 4f_{n-1} + f_n)$$

onde $f_i = f(a + ih), h = (b - a)/n$ e $n = 2^k$. O número de pontos de amostragem é $(n + 1)$ e h é a distância entre dois pontos de amostragem adjacentes. O valor S da integral é aproximado pela seqüência $S_1, S_2, S_3, \ldots$, que converge se a função é suficientemente bem comportada (suave). O método acima é chamado de *método de Simpson.*

1. Escreva uma função que calcule a integral de uma função pelo método de Simpson.

2. Escreva um programa que faça uso da função desenvolvida em (1) para calcular:

(a)

$$\int_0^2 x^2\,dx$$

(b)

$$\int_0^{\Pi/2} \text{sen}x\,dx$$

(c)

$$\int_0^2 e^{-u^2}\,du$$

(d)

$$\int_0^{\Pi/2} \frac{dx}{(a^2\cos^2 x + b^2\text{sen}^2 x)^{1/2}}$$

Sugestão: Defina um dos parâmetros da função de Simpson, como sendo o *código* de uma função. Por exemplo: código 1 corresponde a x^2, código 2 corresponde a sen$x$ e assim por diante. Depois, escreva uma função que, dado o código acima e o valor de x, retorna o valor da função correspondente ao código.

Exercício 21 Segundo Gauss, a integral

$$I = \frac{2}{\Pi}\int_0^{\Pi/2} \frac{dx}{(a^2\cos^2 x + b^2\text{sen}^2 x)^{1/2}}$$

pode ser calculada usando-se duas seqüências convergentes $s_0, s_1, s_2, \ldots$ e $t_0, t_1, t_2, \ldots$, cujos termos são definidos pela relação de recorrência

$$\begin{aligned} s_i &= \frac{s_{i-1} + t_{i-1}}{2} \\ t_i &= \sqrt{s_{i-1}\, t_{i-1}} \end{aligned}$$

para $i > 0$, com $s_0 = a, t_0 = b$ e $\lim s_i = 1/I$. Escreva uma função correspondente a esse cálculo.

Exercício 22 Dada a matriz

$$A = \begin{pmatrix} 2 & 1 & 3 \\ 3 & 3 & 1 \\ 1 & 2 & 1 \end{pmatrix}$$

(a) Execute o comando

for i := 1 **to** 3 **do**
for j := 1 **to** 3 **do** $C[i,j] := A[A[i,j],A[j,i]]$

Qual o valor da variável C resultante? Suponha que todas as variáveis tenham sido declaradas corretamente.

(b) Substitua no comando a variável C por A. Qual o valor de A resultante?

Exercício 23 Uma matriz complexa Z é representada por um par (X, Y) de matrizes reais, de modo que $Z = X + iY$. Faça um procedimento que calcule o produto de duas matrizes complexas (A, B) e (C, D), isto é

$$X + iY = (A + iB)(C + iD).$$

Veja que

$$(A + iB)(C + iD) = (AC - BD) + i(AD + BC).$$

Exercício 24 Um polinômio

$$p_n(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1} + a_n$$

pode ser representado por um vetor de coeficientes. Escreva um procedimento para calcular $p_n(x)$ para um dado valor de x.

Exercício 25 Faça um procedimento que determina tanto o máximo como o mínimo de um conjunto de n números representados pelo vetor A:

var *A: array [1..n] of integer;*

Exercício 26 Escreva um programa que leia um vetor A não ordenado de n inteiros e escreva o vetor na mesma seqüência, ignorando valores duplicados encontrados no vetor lido.

Exercício 27 Escreva procedimentos que, dada uma matriz A de m linhas e n colunas:

(a) Calcule a transposta de A.

(b) Verifique se A é simétrica.

Exercício 28 Dado um elemento $A[i,j]$ de uma matriz A, dizemos que os elementos *adjacentes* a $A[i,j]$ são $A[i-1,j-1]$, $A[i-1,j]$, $A[i-1,j+1]$, $A[i,j-1]$, $A[i,j+1]$, $A[i+1,j-1]$, $A[i+1,j]$ e $A[i+1,j+1]$. Note que $A[1,1]$ tem somente três elementos adjacentes, o mesmo acontecendo com $A[1,n]$, $A[n,1]$ e $A[n,n]$. Analogamente, os outros elementos da primeira e última linha e coluna de A tem somente 5 elementos adjacentes. Todos os outros elementos de A tem 8 elementos adjacentes. Escreva um programa que leia uma matriz A de números inteiros, de m linhas e n colunas e e produza uma matriz B, também de m linhas e n colunas, tal que $B[i,j]$ contenha a média dos elementos adjacentes a $A[i,j]$.

Exercício 29 Dado um vetor v de n números inteiros (índices: $l..u$), escreva um procedimento que os rearranje de maneira que o primeiro ($v[\,l\,]$) seja o máximo da seqüência. Suponha que $1 \leq l$ e $u \leq Max$.

Exercício 30 Dado um vetor v de n números inteiros (índices: $1..Max$), escreva um procedimento que coloque seus elementos em ordem decrescente, utilizando o procedimento do exercício 29.

Exercício 31 Escreva um procedimento que calcula e escreve uma matriz $A_{m\times n}$ cujos elementos são:

$$a_{ij} = i \times j$$

Essa matriz faz parte de uma tabuada de multiplicação. É possível fazer esse exercício usando vetor de apenas uma dimensão? E sem usar vetores?

Exercício 32 Escreva um procedimento

procedure *Multiplica* (**var** *A: matriz, NLinA, NColA: integer;*
var *B: matriz; NLinB, NColB: integer;*
var *C: matriz;* **var** *NLinC, NColC: integer*);

que multiplica duas matrizes, de maneira que:

$$C_{n\times p} = A_{n\times m} \times B_{m\times p}$$

Observação: Os elementos de C são dados por:

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$

Exercício 33 A matriz abaixo representa o Triângulo de Pascal de ordem 6:

```
1
1  1
1  2   1
1  3   3   1
1  4   6   4  1
1  5  10  10  5  1
```

Os elementos extremos de cada linha são iguais a 1. Os outros elementos são obtidos somando-se os dois elementos que aparecem imediatamente acima e à esquerda na linha anterior. Assim, $10 = 4 + 6$.

Escreva três versões de um procedimento que, dado n, gera e escreve o Triângulo de Pascal de ordem n, utilizando:

1. Uma matriz
2. Dois vetores
3. Apenas um vetor

Exercício 34 Uma matriz de permutações é uma matriz quadrada cujos elementos são zeros ou uns tal que em cada linha e em cada coluna exista um e apenas um elemento igual a 1. A matriz abaixo representa uma matriz de permutações de ordem 3:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Escreva um procedimento

procedure *MatrPerm* (**var** *m: matriz; n: integer; i, j: integer*);

que gera uma matriz de permutações de ordem n, supondo que $m_{ij} = 1$.

Exercício 35 Dada uma coleção de n inteiros, escreva um procedimento que determina quantas seqüências isoladas de $1, 2, \ldots, n$ números iguais existem. Exemplo: para $n = 13$,

3 4 4 1 2 5 5 5 2 2 6 2 2

há 4, 3 e 1 grupos de 1, 2 e 3 elementos, respectivamente.

Exercício 36 Seja A um vetor de N elementos

A: **array** [1..N] **of** integer;

Um *segmento quase-ascendente* de tamanho $q - p$ é um segmento de A tal que

A[p] ... A[q], $1 \leq p < q \leq$ N

em que existe no máximo um valor i, $p < i < q$, satisfazendo A[$i-1$] < A[i]. Por exemplo, x, y, z abaixo são segmentos quase-ascendentes:

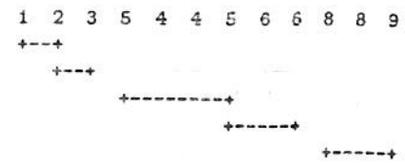
$$\overbrace{4\,\underbrace{5\,8}_{y}\,7\,11}^{x}\,\underbrace{6\,3\,4\,5}_{z}$$

Escreva um procedimento

procedure *MazQuaseAsc* (**var** *vetor: VetorTipo;* **var** *inicio, tamanho: integer*);

que devolve em *início* o início do maior segmento quase-ascendente e em *tamanho* o comprimento do segmento de um *vetor* dado.

Exercício 37 Uma variante do exercício 36: pode-se definir uma *máxima seqüência suave*, onde A[i] − A[j] $\leq 1, \forall i, j$. Exemplo:



Exercício 38 Suponha que deva ser lida uma seqüência de n inteiros positivos, $n \geq 1$, n passado como parâmetro. Escreva um procedimento que determina o tamanho da *maior seqüência ordenada*, isto é, qual a maior subseqüencia $x_i, \ldots, x_j$ onde se verifica $x_i \leq x_{i+1} \leq \ldots \leq x_j$. O procedimento deverá escrever o tamanho dessa seqüência e em que posição começa. Exemplo:


```
Resposta: Tamanho=4; Inicio=5
```

Caso exista mais de uma subseqüência, deve ser dada a posição da primeira delas.

Exercício 39 Suponha que um vetor p de caracteres (índices: 1..*Maxp*) tenha sido preenchido com uma frase lida do teclado. Entre as palavras que compõem a frase pode haver um ou mais brancos ou os sinais de pontuação ',' e '.'. Escreva um procedimento que, dado o vetor p como parâmetro e o seu tamanho corrente (última posição preenchida), escreva as palavras da frase em *ordem alfabética crescente*, uma por linha. O vetor passado como parâmetro não deve ser alterado.

Exercício 40 Suponha que tenham sido feitas as seguintes declarações:

const *MaxVetor* = ... ;
type *Vetor* = **array** [*1..MaxVetor*] **of** *integer;*
procedure *Inverte* (**var** *v: Vetor; n: integer);*

Escreva o corpo do procedimento *Inverte* que inverte um vetor v, isto é, troca o valor $v[\ 1\ ]$ com $v[\ n\ ]$, $v[\ 2\ ]$ com $v[\ n-1\ ]$, etc.

Exercício 41 Suponha que tenham sido feitas as seguintes declarações:

const
MaxLinhaMatriz = ... ;
MaxColunaMatriz = ... ;
type *Matriz* =
array [*1..MaxLinhaMatriz, 1..MaxColunaMatriz*] **of** *real;*

Escreva um procedimento *MaxMinMedMatriz* que recebe como parâmetro uma matriz e seus limites correntes e devolve os valores máximo e mínimo e o que mais se aproxima do seu valor médio.

Exercício 42 Suponha que dois vetores, v_1 e v_2, de inteiros, estejam em ordem *crescente* de seus valores. Escreva um procedimento *Intercala*, com os parâmetros adequados, que devolve um terceiro vetor, ordenado crescentemente, formado a partir dos elementos de v_1 e v_2. Escreva também as declarações de tipos convenientes.

Exercício 43 Escreva uma função *recursiva*

function *Palindroma (var v: VetChar; l,u: integer): boolean*

que devolve *True* se o vetor de caracteres representado por v (índices mínimo e máximo: l e u, respectivamente) armazena uma palíndroma. Diz-se que uma seqüência é *palíndroma* se esta, olhada da esquerda para a direita, for igual à seqüência obtida olhando-se da direita para a esquerda.

Exercício 44 Escreva um procedimento

procedure *Permutacao (v: VetInt; l,n: integer)*

que escreve todas as $(n-l+1)!$ permutações de $n-l+1$ elementos $v[\ l\ ]$, $v[\ l+1\ ]$,..., $v[\ n\ ]$ (l é a posição inicial do vetor de inteiros).

Exercício 45 Suponha que tenham sido feitas as seguintes declarações:

const *MaxVetor = 20;*
type *Vetor* = **array** [*1..MaxVetor*] **of** *char;*
function *Anagrama* (**var** *v1,v2: Vetor; n: integer): boolean;*

Escreva o corpo da função *Anagrama* que devolve *True* se o vetor *v2* representa um anagrama do vetor *v1*. Um vetor é um anagrama de outro se todas as letras de um ocorrem no outro, *em mesmo número, independente da posição.* Ambos os vetores têm o mesmo tamanho, $n \geq 0$. Note que dois vetores de tamanho zero são anagramas entre si. Exemplos: ROMA, MORA, ORAM, AMOR, RAMO são anagramas entre si.

Exercício 46 Uma matriz esparsa é um vetor bidimensional cuja maioria dos elementos é zero. É muito caro armazenar tal matriz, pois poucos elementos possuem informações significativas. Um modo mais eficiente de representar uma matriz esparsa $m \times n$ é usar outra, de tamanho $k \times 3$, na qual se armazenam os índices da linha e coluna e o valor somente dos elementos não nulos. Esta é a chamada *representação reduzida* da matriz esparsa.

Por exemplo, a matriz

$$\begin{pmatrix} 0 & 0 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & -8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

pode ser representada como:

$$\begin{pmatrix} 1 & 3 & 7 \\ 2 & 4 & -8 \\ 4 & 1 & 2 \end{pmatrix}$$

Escreva um procedimento
procedure *RepresentacaoReduzida* (*m*,*n*: *integer*; **var** *mr*: *MatrEspReduzida*);

que lê uma matriz esparsa $m \times n$, *em ordem de linhas*, *produza* e *escreva* sua representação reduzida, devolvendo o resultado no parâmetro *mr*. Escreva a declaração de tipo adequado para *MatrEspReduzida*. Como não será devolvido o tamanho da matriz *mr* gerada, preencha sua última linha com zeros.

Exercício 47 Considere a seguinte declaração de tipo:
type *VetorInteiros* = **array** [1..*MAXVETORINTEIROS*] **of** *integer*;

Escreva uma função *iterativa*
function *NumeroOcorrencias* (**var** *v*: *VetorInteiros*; *l*,*u*: *integer*; *x*: *integer*): *integer*;

e outra *recursiva*
function *NumeroOcorrencias* (**var** *v*: *VetorInteiros*; *l*,*u*: *integer*; *x*: *integer*): *integer*;

que devolvem o número de ocorrências do valor *x* no vetor *v*[*l*..*u*].

Exercício 48 Suponha que números decimais de muitos dígitos, que não poderiam ser armazenados em um tipo inteiro ou real, por exemplo, sejam implementados por registros, da seguinte maneira:

```
type
  NumDec =
      record
        Tam: integer;
        Digs: array [0..MAXNUMDEC] of integer
      end
```

A posição 0 do vetor *Digs* armazena o último dígito (isto é, o menos significativo) do número; a posição 1 o penúltimo, etc, e a posição indicada pelo campo *Tam* armazena o primeiro dígito (isto é, o mais significativo).

Exemplo: se *p* é do tipo *NumDec*, o número inteiro decimal 35690025018 será armazenado da seginte maneira:

0	1	2	3	4	5	6	7	8	9	10	···	MAXNUMDEC
8	1	0	5	2	0	0	9	6	5	3	···	

(acima está representado o conteúdo de *p.Digs*. Neste caso, *p.Tam* = *10*, isto é, *p.Tam* é a última posição preenchida do vetor *p.Digs*). Zeros não-significativos não são armazenados.

Escreva um procedimento
procedure *SomaNumDec* (**var** *p*: *NumDec*; *q*: *NumDec*);

que soma dois inteiros decimais representados dessa maneira, **devolvendo o resultado no primeiro parâmetro**, isto é, o primeiro operando da soma é alterado durante a operação e deverá representar, ao final da mesma, o resultado da operação soma. Note que os números podem ter tamanhos distintos.

Observação: Todo número tem, *no mínimo*, um dígito (ou seja, *p.Tam* = *0*, no mínimo), mesmo que represente o número 0 (zero).

Exercício 49 Usando as mesmas declarações de tipos do Exercício 47, escreva um procedimento
procedure *Intercala* (**var** *p*,*q*: *VetorInteiros*; *tp*,*tq*: *integer*;
var *r*: *VetorInteiros*; **var** *tr*: *integer*);

que *intercala* os valores dos vetores *p* e *q*, de tamanhos *tp* e *tq*, respectivamente, para criar um novo, *r*, de tamanho resultante *tr*. Os dois vetores de entrada estão em ordem crescente de seus valores e o vetor resultante também deve manter essa ordem.

Observação: O número mínimo de elementos de cada vetor é 0 (*zero*).

Impresso em 27 de março de 1995

Segunda Lista de Exercícios

MC100
Primeiro Semestre de 199 5

Esta lista contém exercícios sobre algoritmos recursivos. Em todos eles você deverá escrever um programa ou trecho de programa em Pascal. Faça isso de maneira ordenada, explicitando a base da recursão e o caso geral. Nem sempre isso será óbvio a partir do enunciado. A implementação será tão mais clara quanto for essa descrição inicial. Em alguns casos a diferença entre essa descrição inicial e o programa em Pascal é mínima.

Os exercícios 1 a 5 são exemplos de recursão linear ou unidirecional, isto é, na definição da função ou procedimento há apenas uma chamada recursiva que resolve um subproblema menor que o inicial. Os exercícios 6 a 11 são exemplos de recursão em árvore, em que dois ou mais subproblemas devem ser resolvidos recursivamente.

1. O máximo divisor comum de dois numeros a e b tem a seguinte propriedade: $\mathrm{mdc}(a,b) = \mathrm{mdc}(a-b,b)$. Use essa propriedade para escrever uma versão recursiva da função que calcula o m.d.c. de dois números inteiros.

2. Observe que

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n}{k}\frac{(n-1)!}{(k-1)!(n-k)!} = \frac{n}{k}\binom{n-1}{k-1}.$$

   Escreva uma função recursiva *coef_binomial(n,k)* que calcula $\binom{n}{k}$ usando a propriedade acima.

3. Dados o número real x e um erro eps, a seguinte recorrência descreve uma função aexp(x) que aproxima e^x a um valor y tal que $|y - e^x| \leq eps$.

$$\mathrm{aexp}(x) = \begin{cases} 1/\mathrm{aexp}(-x), & x < 0 \\ (\mathrm{aexp}(x/2))^2, & x > eps \\ 1 + x, & 0 < x \leq eps \end{cases}$$

   Escreva a função aexp(x); faça alguns experimentos usando diferentes valores de eps e x e compare os diferentes valores de $aexp(x) - e^x$.

4. Dado um polígono de $n+2$ lados, a função $C(n)$ abaixo calcula o número de maneiras diferentes de se triangularizar o interior do polígono usando diagonais que não se intersectam:

$$C(n) = \frac{4n-2}{n+1}C(n-1),$$

   onde supomos que $C(0) = 1$. Escreva a função $C(n)$ que calcula o que se deseja. Sabe-se também que $C(n) = \frac{1}{n+1}\binom{2n}{n}$. Qual dessas duas definições é melhor no que toca o número de multiplicações executadas?

5. Dado um vetor com n elementos escreva um procedimento recursivo que imprime as $n!$ permutações desses elementos.

6. Seja v um vetor com n elementos ordenáveis. Sejam v_1 e v_2 a primeira e segunda metades de v, respectivamente. Sejam t_1, d_1 o máximo e o mínimo de v_1 e t_2, d_2 o máximo e o mínimo de v_2. Como você usaria t_1, d_1, t_2 e d_2 para calcular o máximo e o mínimo de v? Deduza de suas conclusões um algoritmo recursivo para calcular simultâneamente o máximo e o mínimo dos elementos de v. Implemente esse algoritmo em Pascal.

7. Escreva em Pascal a função $A(x,y)$ (de Ackerman) descrita a seguir, onde $x, y \geq 0$. Tente executar essa função para alguns valores dos seus parâmetros. Até que valores dos parâmetros você obteve uma resposta sem ocasionar overflow ? Que funções de x são iguais a $A(x,y)$ quando $y = 1, 2$ e 3 ?

$$A(x,y) = \begin{cases} 1, & x = 0, y \geq 0 \\ 2, & x = 1, y = 0 \\ x + 2, & x \geq 2, y = 0 \\ A(A(x-1),y),y-1), & x, y > 1. \end{cases}$$

vide verso

8. Uma outra recorrência que calcula o número de triangulações por diagonais de um polígono é a seguinte:

$$C(n) = \sum_{k=1}^{n} C(k-1)C(n-k),$$

onde $C(0) = 1$ e $C(1) = 1$. Escreva essa versão da função $C(n)$. Qual a desvantagem mais óbvia de se usar essa definição ao invés da outra definição recursiva dada acima?

9. Seja v um vetor de n elementos. O algoritmo de ordenação chamado de *Quicksort* é o seguinte:

   (a) Reorganize v de tal maneira que $v[1]$ ocupe uma nova posição r, com os elementos $v[1], \ldots, v[r-1]$ todos menores ou iguais a $v[1]$ e os elementos $v[r+1], \ldots, v[n]$ todos maiores que $v[1]$.

   (b) Aplique Quicksort recursivamente nas porções $v[1], \ldots, v[r-1]$ e $v[r+1], \ldots, v[n]$. O vetor resultante estará ordenado.

   Escreva e teste o procedimento Quicksort.

10. Seja T uma lista de n caixas de tamanhos diferentes $t_1, t_2, \ldots, t_n$ *em ordem crescente*. Dado um container de tamanho k quer-se decidir se existe um subconjunto de T cuja soma dos tamanhos seja igual a k. Seja $P(n,k)$ a função que é verdadeira se existe tal subconjunto e falsa caso contrário. Então podemos dizer que

$$P(n,k) = P(n-1,k) \text{ ou } P(n-1,k-t_n)$$

Explique porque essa afirmação é correta. Deduza dessa afirmação um algoritmo para calcular $P(n,k)$ e escreva a função correspondente em Pascal.

11. Digite e execute o seguinte procedimento recursivo em Pascal:

```
procedure s(x,y,r : integer);
begin
 if r > 0 then
  begin
   s(x-r,y+r,r div 2);
   s(x+r,y+r,r div 2);
   s(x-r,y-r,r div 2);
   s(x+r,y-r,r div 2);
   quadrado(x,y,r)
  end
end;
```

O procedimento *quadrado(x,y,r)* desenha um quadrado de lado $2r$ centrado no ponto (x,y). Voce deve implementar esse procedimento usando as rotinas gráficas do Turbo Pascal.