| Departamento de Matemática da Universidade de Coimbra | | |
|---|---|---|
| 95/96 | Métodos de Programação | 0 |

# 1 O editor de texto VI - comandos mais frequentes

1. Modos
   - Modo comando: Estado de aceitação de comandos (acções) sobre o texto.
   - Modo texto: Estado de aceitação de escrita de texto.
2. Mudança de modo
   - Comandos de inserção: passagem do modo comando ao modo texto.
   - $<$ ESC $>$ : passagem do modo texto ao modo comando.
3. Comandos
   (a) Comandos de inserção:

   **a** - Junta texto a partir da posição à direita do cursor.
   **A** - Junta texto a partir do fim da linha onde está o cursor.
   **i** - Insere texto a partir da posição à esquerda do cursor.
   **I** - Insere texto antes do primeiro caracter não branco na linha do cursor.
   **o** - Abre uma linha para introdução de texto abaixo da linha do cursor.
   **O** - Abre uma linha para introdução de texto acima da linha do cursor.

   (b) Comandos de posicionamento do cursor sobre o texto:

   **j** - Linha seguinte, mesma coluna.
   **k** - Linha anterior, mesma coluna.
   **l** - uma posição para a direita.
   **h** - uma posição para a esquerda.
   (Nota: os comandos anteriores podem ser efectuados usando as teclas com as setas correspondentes)
   **0 (zero)** - início de linha.
   **$** - fim de linha.
   **G** - fim de texto.
   **%** - parêntese correspondente.

   (c) Comandos de apagamento de texto:

   **x** - Apaga o caracter sob o cursor.
   **dw** - Apaga a palavra à direita do cursor.
   **dd** - Apaga a linha onde está o cursor.
   **D** - Apaga a porção da linha à direita do cursor.

   (d) Comandos de substituição de texto:

   **cw** - Substitui palavra até <ESC>.
   **cc** - Substitui linha até <ESC>.
   **C** - Substitui o resto da linha até <ESC>.
   **r** - Substitui caracter.
   **R** - Substitui caracter até <ESC>.

   (e) Comandos para pesquisa de expressões:

   **/modelo** - Posiciona o cursor no início de modelo se modelo existir.
   **n** - Posiciona o cursor no início da ocorrncia seguinte de modelo.

   (f) Repetição de comandos:

   **.** - Repete a última operação.

(g) Anulação de comandos:

**u** - Anula a última operação.

(h) Junção de linhas:

**J** - Coloca a linha seguinte no fim da linha onde está o cursor.

(i) Replicação de texto:

$< n >$**Y** - Copia para a memória auxiliar as $< n >$ linhas seguintes.

**p** - Copia o conteúdo da memória auxiliar para o editor. Esta memória guarda também a última coisa apagada. Para além da memória auxiliar vulgar existem ainda 26 outras identificadas pelas letras do alfabeto. Para utilizar estas memórias prefixa-se os comandos por "$< letra >$. Por exemplo:

"$< letra >< n >$**Y e** "$< letra >$**p**

**:r** $< nome >$ - obter no editor uma cópia dum ficheiro já existente.

(j) Abandono do editor:

**ZZ** - faz salvaguarda do texto e abandona o editor.

**:q!** - abandona o editor (sem salvaguardar o texto!)

**:w** - faz salvaguarda do texto e permanece no editor.

**:e** <**nome**> - muda de ficheiro.

4. Em modo de texto:

<**DEL**> - Apaga o último caracter escrito (não é imediatamente visivel).

<**CTRL**> **D** - No princípio da linha, move o cursor um TAB para a esquerda.

# 2 Funções Standard

Nesta tabela são apresentadas as funções standard da linguagem PASCAL, de acordo com o tipo dos argumentos e dos resultados. Vejamos como esta tabela deve ser consultada, usando um exemplo:

A função **trunc** encontra-se na linha do tipo **integer** e na coluna do tipo **real**. Isso quer dizer que aceita argumentos do tipo **real** e dá resultados do tipo **integer**.

| | **integer** | **real** | **boolean** | **char** |
|---|---|---|---|---|
| **integer** | abs<br>succ<br>pred<br>sqr | trunc<br>round | ord | ord |
| **real** | sin<br>cos<br>arctan<br>ln<br>sqrt<br>exp | sin<br>cos<br>arctan<br>ln<br>sqrt<br>exp<br>sqr<br>abs | | |
| **boolean** | odd | | pred<br>succ | |
| **char** | chr | | | pred<br>succ |

## 3 Instruções de Leitura e Escrita

**Exercício 3.1** Edite, compile e execute os seguintes programas:

1. ```
   program ola(output);
   begin
           writeln('ola');
   end.
   ```

2. ```
   program dois_mais_dois(output);
   begin
           writeln(2 + 2);
   end.
   ```

3. ```
   program dois_mais_dois_v2(output);
   begin
           write('2 + 2 = ');
           writeln(2 + 2);
           writeln('2 + 2 = ', 2 + 2);
   end.
   ```

4. ```
   program dois_mais_dois_v3(output, f);
   var f : text;
   begin
           rewrite(f, 'resultados');
           writeln('2 + 2 = ', 2 + 2);
           writeln(f, '2 + 2 = ', 2 + 2);
           close(f);
   end.
   ```

5. ```
   program calculo(output, f);
   var f : text;
   begin
           rewrite(f, 'resultados');
           writeln(' 2 + 3 * 4 = ', 2 + 3 * 4);
           writeln('(2 + 3)* 4 = ', (2 + 3) * 4);
           writeln(f, ' 2 + 3 * 4 = ', 2 + 3 * 4);
           writeln(f, '(2 + 3)* 4 = ', (2 + 3) * 4);
           close(f);
   end.
   ```

6. ```
   program calculo(output);
   begin
           writeln(' 2 + 3 / 4 = ', 2 + 3 / 4);
           writeln('(2 + 3)/ 4 = ', (2 + 3) / 4);
   end.
   ```

7. ```
   program soma(input, output);
   var x, y: integer;
   begin
           readln(x, y);
           writeln(x + y);
   end.
   ```
   ```

8. ```
   program soma_v2(input, output);
   var x, y: integer;
   begin
           writeln('De 2 valores inteiros');
           readln(x, y);
           writeln(x, ' + ', y, ' = ', x + y);
   end.
   ```
9. ```
   program soma_v3(f, g);
   var x, y: integer;
       f, g: text;
   begin
           reset(f, 'dados');
           rewrite(g, 'resultados');
           readln(f, x, y);
           writeln(g, x + y);
           close(f);
           close(g);
   end.
   ```

# 4 Instruções de Atribuição

**Exercício 4.1** Edite, compile e execute os seguintes programas:

1. ```
   program soma_v3(input, output);
   var x, y, z: integer;
   begin
           writeln('De 2 valores inteiros');
           readln(x, y);
           z := x + y;
           writeln(x, ' + ', y, ' = ', z);
   end.
   ```
2. ```
   program area (input, output);
   var lado, area : integer;
   begin
           writeln('Este programa calcula a area de um quadrado.');
           write('Escreva a medida do lado : ');
           readln(lado);
           area := lado * lado;
           writeln('A area e'' ', area);
   end.
   ```
3. Altere o programa anterior de modo a calcular a área de :
   - um rectângulo;
   - um triângulo.
4. ```
   program pitagoras (input, output);
   var cateto1, cateto2, hipotenusa : real;
   begin
           writeln('Este programa calcula a medida da hipotenusa');
           writeln('de um triangulo rectangulo pelo teorema de Pitagoras');
           write('Escreva as medidas dos catetos: ');
           readln(cateto1, cateto2);
           hipotenusa := sqrt( sqr(cateto1) + sqr(cateto2));
           writeln('A hipotenusa tem ', hipotenusa, ' de medida.');
   end.
   ```
5. Altere o programa anterior de modo a :
   - evitar o uso da variável hipotenusa;
   - escrever o comprimento da hipotenusa num campo de 7 caracteres;
   - escrever o comprimento da hipotenusa em notação normal com três casas decimais.
   ```

## 5 Tipos de Dados Simples, Inteiros ("Integer") e Reais ("Real")

**Exercício 5.1** Diga o tipo e calcule o valor de cada uma das expressões seguintes:

1. 2 + 3 ∗ 2 + 3 ∗ 4
2. sqr(3)
3. sqr(2.0)
4. trunc(-99.9)
5. round(99.9)
6. -round(99.9)
7. 10 div 3
8. 10 / 3
9. 128 div 3 mod 5
10. 128 mod 5 div 3
11. 193 mod 19 div 3 ∗ 127
12. (8 div 5) / (8 mod 5)

Escreva programas que lhe permitam verificar a correcção das suas respostas.

**Exercício 5.2** Tendo em conta a tabela:

| variável | tipo | valor |
|---|---|---|
| a | real | 5.7 |
| b | real | 8.2 |
| c | integer | 7 |
| d | integer | 4 |

determine o tipo e o valor de cada uma das expressões:

1. sqr(a+b) / (c+d)
2. c ∗ (c mod d)
3. (c mod d) / 2
4. trunc(a-b)
5. trunc(a-b) div c
6. exp(2 ∗ ln(c))
7. (c div d) / (-a)
8. trunc(abs(cos(x))) mod 1
9. 2 ∗ (b - a)

Escreva programas em que lhe seja possível verificar a correcção das suas respostas.

**Exercício 5.3** Considere a seguinte expressão:

$$16.4 - 51.0 \,/\, 4.8 * 0.6 + 74.92$$

Coloque o número mínimo de parentesis na expressão de modo que as operaçoes sejam efectuadas pela ordem indicada em cada uma das alíneas:

1. $- + * /$
2. $* + / -$
3. $/ - + *$

**Exercício 5.4** Proceda de modo análogo ao do problema anterior relativamente à expressão

$$22 + 16 * 100 \text{ mod } 12 - 9 \text{ div } 4$$

e de modo que as operaçoes sejam efectuadas pelas seguintes ordens:

1. + mod ∗ div -
2. div - mod ∗ +
3. - mod div ∗ +

**Exercício 5.5** Supondo que as seguintes declarações tenham sido feitas, diga se cada uma das atribuições é ou não válida, e nesse caso, proponha uma modificação necessária para que a referida instrução se torne válida :

```
var m, n : integer;
    a, b : real;
```

1. m := trunc(b);
2. m := trunc(b) + a;
3. p := m + n;
4. m := n mod a;
5. b := 2.99 ∗ 10;
6. b := $2.99 * 10^9$;
7. n := a - trunc(a);
8. ma_elec := 0.910956E-27;
9. 2ab := 2 ∗ a ∗ b;
10. a := (.54E+2-1);
11. a := m / n;
12. b := 4 + 2a;

Escreva programas em que lhe seja possível verificar a correcção das suas respostas.

**Exercício 5.6** Escreva uma instrução de atribuição para cada uma das seguintes acções:

1. O contador I é incrementado de uma unidade;
2. M é uma cópia do valor de N;
3. Q é o valor da divisão inteira dos inteiros I e J;
4. X é o valor da divisão real dos inteiros I e J;
5. I é o valor arredondado do real positivo X;
6. I é o maior inteiro inferior ou igual a X;
7. M é o maior inteiro contido na média dos reais A e B;
8. A variável t20 toma o valor da tangente de 20 graus;
9. Dado n, inteiro não nulo, a variável inteira SINAL toma o valor 1 se n for positivo e -1 se n for negativo;
10. U toma o valor do algarismo das unidades do real X;
11. R toma o valor de $\sqrt{x}$ se x for não negativo, ou de $\sqrt{-x}$, caso contrário;
12. Z toma o valor de módulo de Y elevado a X;
13. ALFA é o ângulo (em graus) cuja tangente é x;
14. $y \leftarrow y + 4x + 3x^2 + 2x^3 + x^4$ (procure minimizar o numero de operações);
15. $y \leftarrow \frac{\sin(ax)}{2a\cos^2(ax)}$
16. A área de um polígono regular de $n$ lados de comprimento $b$ é dada por:

$$area \leftarrow \frac{1}{4} n b^2 \mathrm{cotg} \frac{\pi}{n}$$

17. $raiz5de3 \leftarrow \sqrt[5]{x^3}$
18. $z \leftarrow \log_6(3x^2+6)$
19. $y \leftarrow \frac{\tan(2x)x^2}{2\sqrt{|x+5|}}$
20. $fraccao \leftarrow a + \frac{d}{e+\frac{d}{e+\frac{f}{g}}}$
21. $x \leftarrow \frac{(\sqrt{a^2+b^2}-a)^4}{\sqrt[4]{a^2+b^2}}$

Escreva programas que lhe permitam verificar a correcção das suas respostas.

# 6 Tipos de Dados Simples, Caracter ("Char") e Lógicos ("Boolean")

**Exercício 6.1** Diga o tipo e calcule o valor de cada uma das expressões seguintes, supondo:

```
var k : integer;
    p, q, r, s : boolean;
    x : real;
```

1. (pred(4)-succ(2)) mod 4 + sqrt(4) = 8
2. not(p or q) = not p and not q
3. not(p and q) = not(not p or not q)
4. (p and (q and not q)) or not (r or (s or not s))
5. not((3 mod 2 = 1) and (7 < 5))
6. (round(-65.3) < trunc(-65.3)) and p
7. (round(sin(x))=0) or (abs(round(sin(x)))=1)
8. odd(k) or odd(k+1)
9. (exp(x) > ln(x)) or s
10. succ('E')
11. ord('D')-ord('A')
12. ord('8')-ord('0')
13. chr(ord('A')+2)

Escreva programas que lhe permitam verificar a correcção das suas respostas.

| Departamento de Matemática da Universidade de Coimbra | | |
|---|---|---|
| 95/96 | Métodos de Programação | 3 |

**Exercício 6.2** Supondo que as seguintes declarações tenham sido feitas, diga se cada uma das instruções é ou não válida e, neste caso, proponha uma modificação necessária para que a referida instrução se torne válida:

```
const espaco = ' ';
var m, n : integer;
    a, b : real;
    p, q : boolean;
    c1, c2 : char;
```

1. c1 := espaco;
2. c1 := 'a';
3. c1 := c2;
4. c1 := 'c2';
5. 'c' := c2;
6. 'c2' := chr('a');
7. m := m-ord('0');
8. p := (m + n) > 0;
9. 'a' := chr(ord('b')+1);
10. b := ord(c1)+ord(c2);
11. m := succ(c1);
12. n := ord(c1)+succ(5);
13. p := q and (ord(c1) <> 'a');
14. p := q and (ord('a') >= 40);

Escreva programas que lhe permitam verificar a correcção das suas respostas.

**Exercício 6.3** Escreva uma instrução de atribuição PASCAL para cada uma das seguintes acções:

1. A variável lógica L é verdadeira se e só se L1 e L2 forem ambas falsas;
2. A variável lógica MAIOR é verdadeira se e só se X é maior que Y e é falsa no caso contrário;
3. A variável lógica L é verdadeira se e só se L1 é verdadeira mas não L2;
4. LOGIC é verdadeira se e só se M for o dobro de N;
5. BOOL é verdadeira se e só se os inteiros K e M forem iguais, em valor absoluto;
6. A variável CONSOANTE é verdadeira se e só se a variável caracter LETRA for uma consoante minúscula;
7. A variável XOR é verdadeira se e só se apenas uma das variáveis B1 ou B2 for verdadeira;
8. A variável lógica PAR é verdadeira se e só se o inteiro N é par;
9. A variável lógica BISSEXTO é verdadeira se e só se a variável inteira ANO for divisível por 4 mas não por 100 ou então for divisível por 400;
10. A variável lógica MULT é verdadeira se e só se I for múltiplo de J (ambos inteiros);
11. A variável lógica VOG é verdadeira se e só se o caracter C for uma vogal;
12. COMPLEX é verdadeiro se e só se a equação $ax^2 + bx + c = 0$ tem raízes complexas;
13. Sendo N uma variável do tipo caracter, o booleano DIGITO é verdadeiro se e só se N representar um algarismo decimal;
14. A variável inteira PAR vale 1 se n for par e 2 se n for ímpar;
15. A variável inteira ALTERNADA toma o valor de $(-1)^n$ com n inteiro;
16. Sendo c um caracter que representa um dígito, atribuir a N o valor numérico desse dígito (N pertencente a $\{0, \ldots, 9\}$);

Escreva programas que lhe permitam verificar a correcção das suas respostas.

**Exercício 6.4** Elabore os seguintes programas (separando as várias partes em secções de texto distintas separadas por curtos comentários, e.g.
{leitura} ...{cálculo} ...{escrita} ...):

1. calcule o lucro obtido na venda de um produto: leia o nome do produto, o número de unidades vendidas, o preço de custo (preco_custo), o preço de venda (preco_venda) (ambos os preços são unitários) e o IVA a pagar (em percentagem - inteiro), cacule o lucro total apurado e escreva o nome do produto e o respectivo lucro na forma ......$ .. (O IVA que o comerciante tem de pagar é calculado sobre a diferença entre o preço de custo e o de venda.)
2. calcule a área de um polígono de $n$ lados de comprimento $b$, sabendo que $A = \frac{1}{4}nb^2\text{cotg}(\frac{\pi}{n})$ .
3. calcule o valor das coordenadas polares $RO$ e $teta$ correspondentes às coordenadas cartesianas $x$ e $y$ sabendo que $RO = \sqrt{x^2 + y^2}$ e $teta = \text{arctg}(\frac{y}{x})$
4. um terreno com 80m por 100m custou 1000 contos. Calcule e escreva o preço por metro quadrado de terreno.

# 7 A instrução condicional "if _ then _ else _"

**Exercício 7.1** Preveja a saída do seguinte programa e introduza-o no computador para verificar se a saída é igual à que previu:

```
program experiencias (input, output);
var     letra: char;
        n : integer;

begin
        write('Escreva uma letra: ');
        readln(letra);
        writeln('O caracter ASCII seguinte ‘a letra "', letra, '" e'' "',
                succ(letra), '" e o anterior e'' "', pred(letra),'"');
        write('Escreva um inteiro no intervalo [32,126]: ');
        readln(n);
        if (n >= 32) and (n <= 126) then begin
                writeln('As funcoes "chr" e "ord" sao inversas uma da outra.');
                writeln('Exemplo: ');
                writeln('chr(ord(', letra, '))= ',chr(ord(letra)) );
                writeln('ord(chr(', n:2, '))= ',ord(chr(n)):2 );
        end else writeln('O valor dado nao pertence ao intervalo indicado.');
end.
```

**Exercício 7.2** Escreva um programa que:

- leia dois números reais
- escreva o maior

**Exercício 7.3** Escreva um programa que:

- leia um número inteiro
- determine se o número é par
- escreva o número indicando se é par ou ímpar

**Exercício 7.4** Escreva um programa que:

- leia um valor real, $x$
- calcule e escreva o valor da função:

$$f(x) = \begin{cases} \frac{1}{x}e^x & \text{se } x > 0 \\ e^{|x|} & \text{se } x \leq 0 \end{cases}$$

**Exercício 7.5** Escreva um programa que, dado $x$ real, calcule $f(x)$ definida por:

$$f(x) = \begin{cases} e - e^{\cos x} & \text{se } x \in [0, 2\pi[ \\ \log\cos x & \text{se } x \in [-2\pi, -\frac{3}{2}\pi[\ \bigcup\ ] - \frac{\pi}{2}, 0[ \end{cases}$$

**Exercício 7.6** Escreva um programa que, dado $x$ real, calcule $f(x)$ definida por:

$$f(x) = \begin{cases} x(\cosh x)^{(2/x)} & \text{se } x > 0 \\ 0 & \text{se } x = 0 \\ x^2 \log(\frac{1-x}{x})^2 & \text{se } x < 0 \end{cases}$$

Nota: $\cosh x = \frac{e^x + e^{-x}}{2}$

**Exercício 7.7** Elabore um programa que calcule as raízes da equação quadrática de coeficientes inteiros:

$$ax^2 + bx + c = 0$$

**Exercício 7.8** Considere a equação da velocidade no instante $t$ ($t \geq 0$):

$$v(t) = \frac{\log |2 - 2t + t^2| - e}{\cos^2[3(t-1)] + e^{-t+1}}$$

Elabore um programa que indique se, num instante $t0$ dado, o objecto se desloca sobre a respectiva trajectória no sentido positivo ou negativo, ou se muda de sentido.

Nota: quando $v > 0$, o objecto desloca-se no sentido positivo; quando $v < 0$, o objecto desloca-se no sentido negativo; quando $v = 0$, o objecto muda de sentido.

**Exercício 7.9** Considere a seguinte definição de uma função real de duas variáveis reais:

$$f(x,y) = \begin{cases} e^{\sin x} + e^{\cos y} & \text{se } x \geq 0 \text{ e } y \geq 0 \\ e^{-\sin x} - e^{-\cos y} & \text{se } x < 0 \text{ e } y \geq 0 \\ \log_e |\sin x + \cos y| & \text{se } x < 0 \text{ e } y < 0 \\ \log_e |\cos x - \sin y| & \text{se } x \geq 0 \text{ e } y < 0 \end{cases}$$

Elabore um programa que devolva o valor da função para valores $x$ e $y$, expressos em *graus*, fornecidos pelo utilizador.

**Exercício 7.10** Pretende-se um programa que converta entre quilómetros por hora, milhas por hora e nós. O programa deve perguntar ao utilizador pelo valor e pelas unidades. Por exemplo:

```
Qual o valor: 13.7894
Quais as unidades (k = kilometros por hora, m = milhas por hora, n = nos): m
```

Ao que o programa deve responder com o resultado:

```
kilometros por hora = 22.19189
nos = 11.98266
```

Recorde-se que: 1 milha = 1.609344 Km e 1 nó = 1.852000 Km por hora.

**Exercício 7.11** Pretende-se um programa que faça conversões entre graus e radianos. O programa deve perguntar ao utilizador qual o valor e as unidades do ângulo. Por exemplo:

```
Qual o valor do angulo: 37.894
Quais as unidades (g = graus, r = radianos): g
```

Ao que o programa deve responder com o resultado:

```
Radianos = 0.6613751
```

Se o utilizador der um caracter não válido quando o programa pedir as unidades, deverá ser escrita uma mensagem de erro. Por exemplo:

```
Qual o valor do angulo: 37.894
Quais as unidades (g = graus, r = radianos): w
ERRO: entrada nao valida para unidades: w
```

Recorde-se que: $\pi$ radianos = 180 graus. $\text{tg}(\pi/4) = 1$.

# 8 Instruções de repetição - os ciclos "while _ do _", "repeat _ until", "for _ do _"

**Exercício 8.1** Leia o seguinte programa e preveja a sua saída:

```
program potencias (output);
var i, p : integer;
begin
        i := 1; p := 2; {condicoes iniciais}
        while p < 1000 do begin
                writeln(i:6, p:10);
                i := i + 1;
                p := p * 2;
        end;
end.
```

**Exercício 8.2** O algoritmo de Euclides permite calcular o maior divisor comum de dois números inteiros positivos, baseando-se na seguinte propriedade:

$$\mathrm{mdc}(a, b) = \begin{cases} a & \text{se } b = 0 \\ \mathrm{mdc}(b, a \bmod b) & \text{se } b \neq 0 \end{cases}$$

Elabore um programa seguindo os seguintes passos:

- peça ao utilizador dois números inteiros positivos e leia esses inteiros ($a$ e $b$)
- enquanto $b \neq 0$
  - atribuir a *resto* o valor de $a$ mod $b$
  - faça $a$ tomar o valor de $b$
  - faça $b$ tomar o valor de *resto*
- escreva os valores dados e o respectivo máximo divisor comum.

**Exercício 8.3** Elabore um programa que leia um número inteiro e verifique se é uma capicua, através dos seguintes passos:

- Peça ao utilizador que forneça um número inteiro
- Leia esse número inteiro, $n$
- Atribua a *ncópia* o valor de $n$
- Inicialize a variável *inverso* a zero
- Enquanto *ncópia* diferente de zero
  - atribua a *dígito* o valor do algarismo das unidades de *ncópia*
  - atribua a *inverso* o seu valor anterior multiplicado por 10 mais o valor de *dígito*
  - atribua a *ncópia* o valor da sua divisão inteira por 10
- Se o número dado for igual a *inverso*
  então escreve "é capicua"
  senão escreve "não é capicua"

**Exercício 8.4** Elabore programas capazes de receber do teclado uma sucessão de números inteiros ($x_i, i \in \mathbb{N}$). O último elemento da sucessão não é utilizado nos cálculos, servindo para indicar o fim da sucessão (elementos deste tipo denominam-se "sentinelas"). Para cada alínea estabeleça o valor da sentinela mais adequado e escreva o respectivo programa.

1. Contar o número de elementos de uma sucessão de termos positivos;
2. Calcular o produto dos elementos de uma sucessão;
3. Determinar o maior e o menor de entre os elementos de uma sucessão de termos ímpares;
4. Determinar o maior elemento e o número de vezes que ocorre numa sucessão de termos negativos;
5. Calcular a média dos elementos positivos e a média dos elementos negativos numa sucessão de termos pares.

**Exercício 8.5** Determine a média $\bar{x}$ dos $i$ elementos da sucessão utilizando a seguinte fórmula:

$$\bar{x}_i = \begin{cases} 0 & \text{se } i = 0 \\ \bar{x}_{i-1} + \frac{x_i - \bar{x}_{i-1}}{i} & \text{se } i \geq 1 \end{cases}$$

**Exercício 8.6** Elabore um programa para calcular o valor da função cos, por desenvolvimento em série, desprezando termos de valor absoluto inferior a $10^{-8}$, sabendo que:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} \ldots$$

**Exercício 8.7** O valor de $\pi$ pode ser calculado através do chamado produto de Wallis:

$$\frac{\pi}{2} = \frac{2}{1} \times \frac{2}{3} \times \frac{4}{3} \times \frac{4}{5} \times \frac{6}{5} \times \frac{6}{7} \times \frac{8}{7} \times \ldots$$

Elabore um programa para o efeito, que irá construindo e multiplicando cada um dos factores, até que a diferença entre dois valores consecutivos seja, em módulo, inferior a $10^{-4}$.

**Exercício 8.8**

1. Dados dois números inteiros positivos escreva um programa que verifique se os dois números são ou não amigos, isto é, se cada um deles é igual à soma dos divisores próprios do outro.
2. Usando a alínea anterior faça um programa que escreva todos os pares de números amigos existentes entre 1 e 1000. Tente optimizar este programa.

# 9 Funções e procedimentos

**Exercício 9.1** Analise os seguintes programas e preveja a sua saída:

1. 
```
program define_funcao(output);
        function dobro (x :integer) :integer;
        begin
                dobro := 2 * x;
        end;
begin
        writeln('O dobro de ', 3:1, ' e'' ', dobro(3):1);
end.
```

2. 
```
program uso_de_procedimento (output);
var x, y : real;

        procedure trocar (var a, b : real);
        var aux : real;
        begin
                aux := a;
                a := b;
                b := aux;
        end;

begin
        x := 2.5; y := 0.5;
        writeln('x = ', x:4:1, ' y = ', y:4:1);
        trocar (x, y);
        writeln('x = ', x:4:1, ' y = ', y:4:1);
end.
```

3. Escreva um procedimento que tenha o mesmo efeito de trocar, mas sem usar a variável auxiliar (só com as operações aritméticas elementares).

4. 
```
program vejamos (output);
var x, b : integer;

        procedure P (var a : integer; b, c : integer);
        begin (* P *)
                b := b * c;
                a := b;
                writeln('P: a = ', a:1, ' b = ', b:1, ' c = ', c:1);
        end;

begin (* vejamos *)
        x := -3; b := 3;
        P (x, x, b);
        writeln('x = ', x:1, ' b = ', b:1);
        P (b, b, x);
        writeln('x = ', x:1, ' b = ', b:1);
        P (b, x, x);
        writeln('x = ', x:1, ' b = ', b:1);
        P (b, x, b);
        writeln('x = ', x:1, ' b = ', b:1);
end.
```

5. 
```
program calcula_potencia (input, output);
var     x, y : integer;
        z : real;

        procedure ler_dados (var a, b :integer);
        begin
                write('Escreva a base : '); readln(a);
                write('Escreva o expoente : '); readln(b);
        end;

        function pot (b, e : integer) : real;
        begin
                pot := exp(e * ln(b));
        end;

        procedure escreve_resultado (b, e : integer; r : real);
        begin
                writeln(b:1, ' ^ ', e:1, ' = ', r:4:2);
        end;

begin (* calcula_produto *)
        ler_dados (x, y);
        z := pot(x, y);
        escreve_resultado (x, y, z);
end.
```

**Exercício 9.2** Implemente as seguintes funções:

1. $$\begin{array}{rcl} \log : \mathbb{R} \times A & \longrightarrow & \mathbb{R} \\ (x,a) & \longmapsto & \log_a(x) = \frac{\ln(x)}{\ln(a)} \end{array}$$
   com $A = \{x \in \mathbb{R} : x > 0 \text{e } x \neq 1\}$

2. $$\begin{array}{rcl} \text{senh} : \mathbb{R} & \longrightarrow & \mathbb{R} \\ x & \longmapsto & \text{senh}(x) = \frac{e^x - e^{-x}}{2} \end{array}$$

3. $$\begin{array}{rcl} \text{f} : \mathbb{R} & \longrightarrow & \mathbb{R} \\ x & \longmapsto & \begin{cases} \sqrt{x} & \text{se } x \geq 0 \\ \sqrt{-x} & \text{se } x < 0 \end{cases} \end{array}$$

4. $$\begin{array}{rcl} \text{argsech} : ]0,1] & \longrightarrow & \mathbb{R} \\ x & \longmapsto & \text{argsech}(x) = \log(\frac{1+\sqrt{1-x^2}}{x}) \end{array}$$

| Departamento de Matemática da Universidade de Coimbra | | |
|---|---|---|
| 95/96 | Métodos de Programação | 6 |

## 10 Recursão

**Exercício 10.1** Elabore um programa que calcule os $k$ primeiros termos da série harmónica:

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\ldots+\frac{1}{n}+\ldots$$

**Exercício 10.2** Escreva um subprograma que calcule o factorial de um número. Usando esse subprograma escreva um programa que calcule o número de combinações de $m$, $n$ a $n$ sabendo que:

$$C_n^m=\frac{m!}{n!(m-n)!}$$

**Exercício 10.3** Escreva um programa que calcule o m.m.c de dois números inteiros positivos, baseando-se na seguinte propriedade:

$$\mathrm{mmc}(a,b)=\frac{a\times b}{\mathrm{mdc}(a,b)} \quad \text{e sabendo que:} \quad \mathrm{mdc}(a,b)=\begin{cases} a & \text{se } b=0 \\ \mathrm{mdc}(b, a \bmod b) & \text{se } b\neq 0\end{cases}$$

**Exercício 10.4** O sub-factorial ($!n$) de um número inteiro não negativo é definido por:

$$!n=\begin{cases} 1 & \text{se} \quad n=0 \\ !(n-1).n+(-1)^n & \text{se} \quad n>0\end{cases}$$

1. Escreva uma função recursiva para calcular $!n$.
2. Escreva a correspondente versão iterativa.

**Exercício 10.5** Implemente uma função recursiva que permita calcular a potência:

$$x^n=\begin{cases} 1 & \text{se} \quad n=0 \\ x^{n-1}*x & \text{se} \quad n>0 \\ x^{n+1}/x & \text{se} \quad n<0\end{cases}$$

para $n$ inteiro e $x$ real.

**Exercício 10.6** Seja Comissões$(m,n)$ o número de diferentes comissões de $n$ pessoas que se podem formar por eleição entre $m$ pessoas. Por exemplo, Comissões$(4,3)=4$, porque dadas quatro pessoas, **A, B, C** e **D**, há quatro possíveis comissões de três membros,**ABC, ABD, ACD** e **BCD**. É sabido que:

- Comissões$(m,n)=1$ quando $m=n$;
- Comissões$(m,n)=m$ quando $n=1$;
- Comissões$(m,n)=$Comissões$(m-1,n-1)+$ Comissões$(m-1,n)$ quando $m>n>1$.

Escreva uma função recursiva em Pascal que calcule Comissões$(m,n)$ para $m\geq n\geq 1$.

**Exercício 10.7** Implemente um procedimento recursivo que permita escrever, por ordem inversa, os dígitos de um número inteiro.

**Exercício 10.8** Para uso dos alunos do Ensino Primário, pretende-se imprimir uma tabuada de multiplicar. Por exemplo, para $n=5$, tal tabuada terá o seguinte formato:

```
1
2  4
3  6   9
4  8  12  16
5 10  15  20  25
```

Considerando

```
type valor = 1..9;
```

elabore um procedimento recursivo para imprimir uma tabuada de tamanho $n$, de cabeçalho:

```
procedure imprimir tabuada (n: valor);
```

**Exercício 10.9** Considere a seguinte função, definida em $\mathbb{N}$:

$$P(n) = \prod_{i=1}^{n}(2i - 1)$$

1. Elabore uma função recursiva para o seu cálculo.

2. Construa uma versão iterativa da mesma função.

# 11 Tipo de Dados Compostos, Tabelas ("Array")

**Exercício 11.1** Preveja a saída dos seguintes programas e introduza-os no computador para verificar se a saída é igual à que previu:

1. 
```
program UsoDeVectores (input, output);
type    nao_negativo = 0..maxint;
var     numero : nao_negativo ;

    procedure ler_numero(var num : nao_negativo);
    var lido : integer;
    begin
        write('Indique um numero: ');
        readln(lido);
        while lido < 0 do begin
            writeln('O numero nao pode ser negativo.');
            write('Indique um numero: ');
            readln(lido);
        end;
        num := lido;
    end;

    function capicua(num : nao_negativo) : boolean;
    type    TipoIndice = 1..10; (* Numeros ate' 10 algarismos *)
    var     indice, inicial, final : TipoIndice;
            a : array[TipoIndice] of 0..9;
    begin
        indice := 1;
        while num > 0 do begin
                a[indice] := num mod 10;
                num := num div 10;
                indice := indice + 1;
        end;
        inicial := 1;
        final := indice - 1;
        while (a[inicial] = a[final]) and (inicial < final) do begin
                inicial := inicial + 1;
                final := final - 1;
        end;
        capicua := inicial >= final;
    end;

begin
        ler_numero(numero);
        if capicua(numero)
        then writeln(numero:1, ' e'' capicua')
        else writeln(numero:1, ' nao e'' capicua')
end.
```

2.
```
program UsoDeMatrizes (input, output);
const   max = 10;
type    indice = 1..max;
        matriz = array [indice, indice] of real;

var     n : indice;
        a : matriz;

    procedure ler_tamanho(var n : indice);
    var lido : integer;
    begin
        repeat
            write('Indique o tamanho da matriz (entre 1 e ', max:1, '): ');
            readln(lido);
        until (lido >= 1) and (lido <=  max);
        n := lido;
    end;

    procedure ler_matriz(n : indice; var a : matriz);
    var i, j : indice;
    begin
        writeln('Escreva os elementos da matriz por linhas: ');
        for i := 1 to n do begin
                for j := 1 to n do
                    read(a[i,j]);
                readln;
        end;
    end;
    procedure escrever_matriz(n : indice; a : matriz);
    var i, j : indice;
    begin
        for i := 1 to n do begin
                for j := 1 to n do
                    write(a[j,i]:8:3);
                writeln;
        end;
    end;
begin
        ler_tamanho(n);
        ler_matriz(n, a);
        writeln;
        writeln('Diga que matriz e'' a que obtem seguidamente:');
        escrever_matriz(n, a);
end.
```

3.
```
program MatrizesESubprogramas (input, output);
type    indice = 1..100;
        vector = array[indice] of integer;
var     n: indice;
        i: integer;
        v: vector;
        mp, mi : real;
```

```
    procedure medias (v : vector; n : indice; var mpares, mimpares : real);
    var contpar, contimpar, somapar, somaimpar, i :integer;
    begin
        contpar := 0; contimpar := 0;
        somapar := 0; somaimpar := 0;
        for i := 1 to n do
            if odd(v[i])
            then begin
                contimpar := contimpar + 1;
                somaimpar := somaimpar + v[i];
            end else begin
                contpar := contpar + 1;
                somapar := somapar + v[i];
            end;
        if contimpar <> 0 then mimpares := somaimpar / contimpar
                          else mimpares := 0;
        if contpar <> 0 then mpares := somapar / contpar
                        else mpares := 0;
    end;
begin
    write('numero de componentes do vector : ');
    readln(n);
    write('componentes : ');
    for i := 1 to n do read(v[i]);
    readln;
    medias(v, n, mp, mi);
    writeln('A media dos pares e'' ', mp, ' e dos impares e'' ', mi);
end.
```

*Nota:* Todos os programas seguintes devem ser convenientemente modularizados através de subprogramas.

**Exercício 11.2** Escreva um programa que, dados um inteiro positivo $k$ e um vector de $n$ inteiros positivos, calcule a média dos múltiplos de $k$ e a média dos submúltiplos de $k$, existentes no vector.

**Exercício 11.3** São dados os $n$ elementos inteiros de um vector $x$ ($n$ constante e igual a 100) e ainda um valor inteiro $k$. Escreva um programa para imprimir todos os pares de números $x_i, x_j$, tais que $x_i + x_j = k$.

**Exercício 11.4** Escreva um programa para:

1. ler o inteiro $n$;
2. ler os $n \times n$ elementos inteiros de uma matriz $A$;
3. contar quantos elementos nulos existem acima da diagonal principal
4. se o número de elementos nulos for par, escrever a matriz, caso contrário escrever a sua transposta.

**Exercício 11.5** Faça subprogramas para:

1. Somar duas matrizes $A(n, m)$ e $B(n, m)$.
2. Multiplicar duas matrizes $A(n, m)$ e $B(m, k)$.
3. Dada uma matriz $A(n, m)$ de elementos reais, determinar a linha cuja soma dos seus elementos é máxima.
4. Calcular o determinante de uma matriz triangular $A(n, n)$ de elementos reais.

**Exercício 11.6** Escreva um programa que, dada uma matriz $A$ de $m \times n$ elementos reais, determine quantos são superiores ao valor da média de todos os elementos da matriz.

**Exercício 11.7** Um treinador de atletismo treina 5 atletas e faz 12 sessões de treino por semana. Em cada sessão, cada atleta percorre uma distância que é cronometrada. Os valores dos tempos, em segundos, são registados sob a forma de uma matriz $T(5 \times 12)$, onde cada linha diz respeito a um atleta e cada coluna a cada sessão de treino. Supondo já feita a leitura da matriz, escreva uma secção de programa para:

1. calcular e escrever a média dos tempos realizados em cada sessão de treinos;
2. determinar e escrever o melhor tempo realizado por cada um dos atletas nas 12 sessões.

**Exercício 11.8** Escreva um programa que, dada uma matriz quadrada de ordem $n$, $(0 < n \leq 100)$ de elementos inteiros, e dados dois inteiros $k$ e $l$, devolva a matriz após troca entre si das colunas $k$ e $l$.

**Exercício 11.9** Numa matriz $M(35 \times 12)$ de elementos inteiros, encontram-se registadas as notas dos 35 alunos de cada uma das 12 turmas de Métodos de Programação. Sabendo que todos os elementos de $M$ são valores entre 0 e 20, escreva um programa que:

1. determine e escreva o número de alunos aprovados (nota $\geq 10$);
2. determine e escreva a melhor nota em cada uma das turmas;
3. identifique a turma com maior número de alunos aprovados.

**Exercício 11.10**
1. Dada uma matriz de inteiros, de tipo $m \times n$ $(m, n \leq 100)$, elabore um subprograma para fazer uma permutação para a esquerda das colunas da matriz;
2. Escreva o correspondente programa de chamada.

**Exercício 11.11** Considere o seguinte procedimento:

```
procedure trocar (i, j: integer);
var aux: real;
begin
    if j>i then begin
        aux:=x[i];
        x[i]:=x[j];
        x[j]:=aux;
        trocar(i+1,j-1)
    end
end;
```

1. Sendo **x** um vector de **n** elementos reais, diga o que sucede ao vector **x** se o programa principal contiver a instrução:

   ```
   trocar(1,n);
   ```

2. Elabore um procedimento não recursivo para o mesmo efeito.

**Exercício 11.12** Escreva, usando um algoritmo recursivo, um subprograma que, dados um elemento **k** e um vector de **n** $(0 < \mathbf{n} \leq 100)$ elementos inteiros, determine o número máximo de ocorrências de **k** no vector.

**Exercício 11.13** Dado um vector de $n$ inteiros e um inteiro $k$ fazer um subprograma que procure $k$ nas $n$ primeiras posições do vector. O resultado deste subprograma deve ser a posição onde o elemento está, ou uma indicação de falha. (No caso de haver repetições o subprograma pode fornecer como resultado qualquer uma das posições).

No pior caso quantos elementos do vector é que este subprograma tem de examinar ?

**Exercício 11.14** Faça um subprograma que encontre o maior dos $n$ primeiros elementos de um vector de inteiros.

**Exercício 11.15** Utilizando o subprograma anterior ordene os primeiros $n$ elementos do vector usando o seguinte algoritmo:

1. encontre o maior dos $n$ elementos;
2. coloque esse elemento na sua posição final trocando-o com o enésimo elemento;
3. neste momento para ordenar todo o vector basta ordenar as $n-1$ primeiras posições do vector.

Note-se que a recursão implícita na descrição anterior pode ser facilmente eliminada.
(Este algoritmo de ordenação chama-se **selecção linear**.)

**Exercício 11.16** Outro algoritmo de ordenação é a adaptação de um usado pelos jogadores de cartas.

Consideramos o vector como a concatenação de duas sequências: uma ordenada, e a segunda não ordenada. No início a primeira sequência contém apenas um elemento.

O primeiro elemento da sequência não ordenada é inserido na posição correcta da sequência ordenada (movendo os elementos maiores uma posiçao para a esquerda).

Esta operação aumentou a sequencia ordenada de um elemento (e retira esse elemento à não ordenada). Repetimos até todos os elementos estarem na sequência ordenada.

Note-se que a inserção pode ser simplificada usando uma sentinela.

(Este algoritmo de ordenação chama-se **inserção linear**.)

**Exercício 11.17** Dado um vector de $n$ inteiros e um inteiro $x$ existente no vector pretende-se particionar esse vector em duas partes: os elementos inferiores a $x$ e os elementos superiores a $x$.

Uma maneira de fazer isto é:

1. Começando do lado esquerdo do vector procurar um elemento que seja maior ou igual a $x$.
2. Começando do lado direito do vector procurar um elemento que seja menor ou igual a $x$.
3. Trocar estes dois elementos.
4. Continuar até as duas procuras se cruzarem

**Exercício 11.18** Usando o subprograma anterior ordene um vector:

1. Particione o vector, usando como separador o seu elemento médio;
2. Se a partição esquerda tiver mais de um elemento ordene-a;
3. Se a partição direita tiver mais de um elemento ordene-a;

(Este algoritmo de ordenação chama-se **quicksort**.)

**Exercício 11.19** Igual ao exercício 11.13, mas aproveitando o facto de o vector estar ordenado.

Sugestao: considere o elemento na posição média do vector ($m = a[n\ div\ 2]$).

- Se $k = m$, está encontrado.
- Se $k < m$, só pode estar na primeira metade do vector.
- Se $k > m$, só pode estar na segunda metade do vector.

**Exercício 11.20** Dado um vector de 10 componentes reais $\{x_i\}_{i=1,\ldots,10}$ pretende-se normalizá-lo, isto é, dividir cada componente pela norma do vector, em que:

$$\text{norma} = \sqrt{\sum_{i=1}^{10} x_i^2}$$

1. Construa a função:

   ```
   function norma (x: vector): real;
   ```

2. Um programador menos atento, escreveu a seguinte secção de programa principal:

   ```
       ⋮
   for i:=1 to 10 do
      x[i]:=x[i]/norma(x);
       ⋮
   ```

   Esta forma, além de ineficiente, conduz também a um resultado errado. Justifique esta afirmação.

3. Elabore um programa completo, para ler o vector e escrever o resultado já normalizado, utilizando a função definida em (a).

**Exercício 11.21** Faça um subprograma **recursivo** para calcular o determinante de uma matriz quadrada de ordem **n**.

**Exercício 11.22** Escreva um subprograma **recursivo** que calcule a média ponderada de uma sequência de $n$ números, $x_1, x_2, \ldots, x_n$, através da seguinte fórmula:

$$xmed = f_1x_1 + f_2x_2 + \ldots + f_nx_n$$

em que $f_1, f_2, \ldots, f_n$ são os factores de ponderação. Suponha que os valores reais dos elementos $f_i,\ x_i,\ i = 1, \ldots, n$ estão armazenados em dois vectores $f$ e $x$, respectivamente.

## 12 Tipo de Dados Compostos, Registos ("Records")

**Exercício 12.1**

1. Escreva subprogramas para operar com complexos, declarando *complexo* como uma estrutura do tipo `record`;
2. Como se sabe uma matriz quadrada de elementos complexos diz-se hermítica se for igual à sua associada, isto é se:

$$A = A^*$$

em que $A^*$ é a matriz transposta da matriz conjugada de $A$.

(a) Considere a seguinte definição:

"Dois números reais dizem-se iguais se a distância entre ambos for inferior a $10^{-30}$"

Elabore uma função para verificar a igualdade entre dois números reais de cabeçalho:

```
function iguais (x,y: real): boolean;
```

(b) Defina o tipo complexo utilizando uma estrutura de `record`;

(c) Considerando as declarações:

```
type indice = 1..20;
     matriz = array[indice,indice] of complexo;
```

elabore uma função para verificar se uma dada matriz é ou não hermítica, utilizando o conceito de igualdade entre reais definido em 1. A função terá como cabeçalho:

```
function hermitica (A: matriz; n: indice): boolean;
```

Por razões de eficiência o algoritmo deverá parar logo que encontre um par de elementos correspondentes que não sejam conjugados.

**Exercício 12.2**

1. Escreva subprogramas para operar com fracções (ler, escrever, simplificar, somar, multiplicar, dividir, subtrair, calcular potências de fracções), declarando as fracções como estruturas do tipo `record` cujos campos são do tipo `integer`;
2. Elabore um programa para escrever os primeiros $n$ termos de uma sucessão associada à *série harmónica*:

$$H = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

sob a forma de fracção. Por exemplo, para $n = 4$, a saída do programa deverá ser:

```
1
3/2
11/6
25/12
```

Deverá estruturar devidamente o seu programa, incluindo subprogramas para:

— escrever uma fracção;

— somar duas fracções;

— simplificar uma fracção.

SUGESTÃO: Considere o tipo:

```
type fraccao = record
                 num, den: integer
               end;
```

**Exercício 12.3** Considere as seguintes declarações:

```
type julgamento = record
                       reu: integer; (* numero de identificacao *)
                       condenado: boolean
                  end;
     listajulg = array[1..200] of julgamento;
var listareus: listajulg;
```

Construa um subprograma para obter a lista (`reuscond`) dos réus que formam condenados, por ordem crescente dos números de identificação.

**Sugestão:**

1. a partir da lista com todos os réus (`listareus`) forme a lista dos réus condenados (`reuscond`),
2. ordene a lista.

**Exercício 12.4** Suponha as seguintes declarações:

```
type data = record
              dia: 1..31;
              mes: 1..12;
              ano: 1900..2100
            end;
     identidade = record nome: packed array [1..35] of char;
                         data_nas: data;
                         sexo: 1..2; (* 1=feminino, 2=masculino *)
                         estado_civil: 1..4 (* 1=solt, 2= cas, 3=viu, 4=div *)
                  end;

var candidato1, candidato2: identidade;
    dia_casamento: data;
```

1. Faça um subprograma para saber se um candidato tem 16 anos no dia do previsto casamento;
2. Faça um programa para saber se `candidato1` e `candidato2` podem casar, isto é, se são de sexos opostos, se têm ambos mais de 16 anos no dia do previsto casamento e se não são casados.

**Exercício 12.5** Sendo dados os seguintes tipos:

```
type
    Codigo = 0..1000;
    Peca = record
                Nome            : packed array [1..20] of char;
                Disponivel      : Boolean;
                PrecoUnitario   : Real;
           end;
    Armazem = array [Codigo] of Peca;
```

que descrevem uma armazém de peças. Escreva um subprograma de facturação que, recebendo os códigos e quantidades das peças encomendadas escreva, em papel, uma factura cujas *linhas de detalhe*, organizadas por coluna, contêm:

1. Para cada peça *encomendada e disponível*, o código, nome, preço unitário, quantidade encomendada e preço total da quantidade encomendada;
2. Para cada peça *encomendada e não disponível*, o código, nome, preço unitário, quantidade encomendada e a mensagem "não disponível" na coluna correspondente ao preço total;

e cuja *linha de total*, posicionada após todas as *linhas de detalhe*, contêm, na coluna correspondente ao preço total, a soma dos preços totais das quantidades encomendadas de todas as peças disponíveis. Declare os tipos e variáveis de que necessitar para elaborar o subprograma pedido. Indique como invocaria o dito subprograma.

**Exercício 12.6** São dados os seguintes tipos:

```
type
    Planeta = record
                Nome            : packed array [1..7] of char;
                Visivel         : Boolean;
                RaioOrbital     : Real;
              end;
    Sistema = array [1..9] of Planeta;
```

que descrevem os Sistema Solar. Escreva um subprograma que escreva o nome e o raio orbital dos planetas visíveis, da Terra, a olho nu.

## 13 Tipo de Dados Compostos, Conjuntos ("Sets")

**Exercício 13.1** Considere as seguintes declarações:

```
type
    elemento = inf..sup;
    conjunto = set of elemento;
```

onde `inf` e `sup` são constantes declaradas do modo mais conveniente. Escreva subprogramas para:

1. Ler um conjunto;
2. Escrever um conjunto;
3. Calcular o cardinal de um conjunto;
4. Calcular o conjunto intersecção de dois conjuntos;
5. Calcular o conjunto complementar de um conjunto.

**Exercício 13.2** Considerando `inf` = 2 crie e escreva o conjunto de numeros primos menores ou iguais a `sup` usando o método do Crivo de Eratóstenes.

(O Crivo de Eratóstenes consiste em, começando com uma lista completa dos inteiros, eliminar sucessivamente os multiplos de 2, 3, 5, ... até a lista só conter números primos.)

**Exercício 13.3** Considere as seguintes declarações:

```
const capacidade = 100;
type
    indice = 1..capacidade;
    tipo_de_doente = (em_observacao, recem_operado, convalescente);
    cama = record
              utilizada: boolean;
              doente: tipo_de_doente
           end;
    alta = set of indice;
    hospital = array[indice] of cama;
```

Elabore um subprograma que dado um vector do tipo `hospital` construa um conjunto (do tipo `alta`) de todas as camas que estão a ser utilizadas e que correspondem a doentes em convalescênça.

**Exercício 13.4** Considere as seguintes declarações:

```
const npess = 100;
type  pessoas = 1..npess;
      conjunto = set of pessoas;
      cidades = (Lisboa, Porto, Coimbra);
      registo_de_amigos = record
                            amizade: conjunto;
                            local_nas: cidades;
                            data_nas: record
                                        ano, mes, dia: integer
                                      end
                          end;
var a: array[1..npess] of registo_de_amigos;
```

Elabore um subprograma que dado o vector `a` e as pessoas `i` e `j` ($i, j \in \{1, \ldots, 100\}$), determine quais os amigos comuns a ambos que nasceram em Coimbra em 1967.

**Exercício 13.5** Considere as seguintes declarações:

```
type pais = 1..30;
     data = record dia, mes, ano: integer
            end;
     vizinhos = set of pais;
     cores = (amarelo, azul, vermelho, verde);
     paises = record num_pais: pais;
                     pintados: cores;
                     vizinhanca: vizinhos;
                     independencia: data
              end;
     mapa = array[pais] of paises;
 var europa: mapa;
```

Pretende-se:

1. Um subprograma que determine o número de paises que se tornaram independentes no século XX;
2. Considerando o mapa "já pintado", um subprograma que verifique se está "bem pintado", isto é, se não existem paises vizinhos pintados da mesma cor.

**Exercício 13.6** Sendo dadas as declarações:

```
type
  Dimensao = 1..1000;
  Cliente  = record
               NumeroConta : Integer;
               Saldo       : Real;
             end;
  ConjuntoVigaristas = set of Integer;
var
  Banco      : array[Dimensao] of Cliente;
  ListaNegra : ConjuntoVigaristas;
```

elabore subprogramas para:

1. Ler o vector `Banco`;
2. Escrever o conjunto `ListaNegra`;
3. Dado o `NumeroConta` determinar a posição (índice) do `Cliente` no `Banco`;
4. Dado o `NumeroConta` do `Cliente` e a importância que este pretende levantar actualizar o `Banco` e a `ListaNegra` de acordo com as seguintes regras:
   - Os clientes que estao na lista negra podem levantar até 50% do seu saldo;
   - Os clientes que não estao na lista negra podem levantar até 100% do seu saldo;
   - Os clientes que tentarem levantar mais de 100% do seu saldo vão para a lista negra e têm o levantamento recusado.

## 14 Tipo de Dados Compostos, Ponteiros ("Pointers")

**Exercício 14.1** Considere a seguinte declaração

```
Type seta = ^pessoa;
     tiponome = packed array [1..30] of char;
     pessoa = record
                nome : tiponome
                prox : seta;
              end;
```

1. Diga o que representa.
2. Leia os seguintes suprogramas:

```
procedure um(ponta :seta);
var     p :seta;
begin
        p := ponta;
        while p <> nil do begin
                writeln(p^.nome);
                p := p^.prox;
        end;
end;
function dois(ponta :seta):integer;
var     p :seta;
        c : integer;
begin
        p := ponta;
        c := 0;
        while p <> nil do begin
                c := c + 1;
                p := p^.prox;
        end;
        dois := c;
end;
procedure tres(n :tiponome; var ponta :seta);
var     ant, este :seta;
begin
        ant := nil;
        este := ponta;
        while este^.nome <> n do begin
                ant := este;
                este := este^.prox;
        end;
        if ant = nil
        then ponta := este^.prox
        else ant^.prox := este^.prox;
end;
```

Preveja o efeito de cada um deles, supondo que ponta é uma estrutura, já construída, do tipo declarado.

**Exercício 14.2** Considerando os mesmos tipos do exercício anterior:
Para a lista, `esta_lista`, já construída. e considerando o ponteiro `p` para um dos elementos da lista:

1. Escreva um subprograma para retirar da lista, o elemento seguinte ao apontado por `p`.
2. Escreva um subprograma para retirar da lista, o elemento apontado por `p`.

**Exercício 14.3** Considere a seguinte declaração:

```
Type seta = ^elemento;
     tiponome = packed array [1..10] of char;
     elemento = record
                        nome :tiponome;
                        seguinte :seta;
                end;
var  lista, p, q, novo :seta;
```

Dada a lista ligada, `lista`, já construída em memória, escreva subprogramas para:

1. Inserir o registo apontado por `novo`, entre o registo apontado por `p` e o seguinte.
2. Inserir o registo apontado por `novo`, entre o registo apontado por `p` e o anterior a este.

**Exercício 14.4** Um polinomio em $x$ de coeficientes reais pode ser representado por uma sequência de pares ordenados (coeficientes, grau) de preferência ordenados por ordem decrescente das potências de $x$.

Por exemplo, o polinómio

$$x^8 + 5x^6 - 7x^5 + 6x + 1$$

pode ser representado por

$$(1, 8)(5, 6)(-7, 5)(6, 1)(1, 0)$$

1. Utilizando ponteiros defina um tipo que representa a sequência anterior como uma lista ligada.
2. Escreva um subprograma `lerpolinomio`, que leia uma sequência de pares de numeros (coeficientes, grau), dado por ordem decrescente das potências de $x$.
   Por exemplo o polinómio anterior seria dado por:

   $$1\ 8\ \ 5\ 6\ \ -7\ 5\ \ 6\ 1\ \ 1\ 0$$

   O subprograma deverá construir a lista ligada representativa do polinómio.
3. Escreva uma função `soma` para somar dois polinómios e uma função produto para multiplicar um número real por um polinómio.
4. Elabore um programa de chamada correspondente.

**Exercício 14.5** Sendo dados os seguintes tipos:

```
Type A = ^B;
     B = record
             i :integer;
             s :A;
         end;
```

Preveja o resultado da execução do procedimento `Beta`:

```
Procedure Beta (var x : A);
var     a, b, c : A;
begin {Beta}
        a := nil;
        b := x;
        while not (b = nil) do begin
                c := b^.s;
                b^s := a;
                a := b;
                b := c;
        end;
        x := a;
end; {Beta}
```