Departamento de Matemática da Universidade de Coimbra		
2003/2004	**Métodos de Programação II**	**4**

3 Tipo de Dados Compostos, Conjuntos ("Sets")

Exercício 3.1 Considere as seguintes declarações:

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

onde `inf` e `sup` são constantes declaradas do modo mais conveniente. Escreva sub-programas 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 3.2 Escreva sub-programas para:

1. Determinar quantos algarismos distintos contém um inteiro positivo dado.
2. Construir o conjunto dos algarismos distintos de um inteiro positivo.
3. Construir o conjunto dos algarismos pares que são comuns a dois inteiros.

Exercício 3.3 Considerando `inf` $= 2$ crie e escreva o conjunto de números 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 múltiplos de 2, 3, 5, ... até a lista só conter números primos.)

Exercício 3.4 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 pintado: cores;
                    vizinhanca: vizinhos;
                    independencia: data
             end;
    mapa = array [pais] of paises;
var europa: mapa;
```

Pretende-se:

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

Exercício 3.5 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 sub-programa 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 convalescença.

Exercício 3.6 Considere as seguintes declarações:

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

Elabore um sub-programa que dado um vector `registo_de_amigos` 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 3.7 Sendo dadas as declarações:

```
const Imax = 100;
      Cmax = 10000;
type
    Dimensao = 1..Imax;
    Contas   = 1..Cmax;
    Cliente  = record NumeroConta : Contas;
                      Saldo       : Real;
               end;
    ConjuntoVigaristas = set of Contas;
    Bancos   = array [Dimensao] of Cliente;
var Banco      : Bancos;
    ListaNegra : ConjuntoVigaristas;
```

elabore sub-programas 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 estão na lista negra podem levantar até 50% do seu saldo;
   - os clientes que não estão 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.