Exercício 1.17 Usando o sub-programa 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 1.18 Igual ao exercício 1.12, mas aproveitando o facto de o vector estar ordenado.
Sugestão: 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 1.19 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 1.

Exercício 1.20 Faça um sub-programa **recursivo** para calcular o determinante de uma matriz quadrada de ordem **n**.

Exercício 1.21 Escreva um sub-programa **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.

2 Tipo de Dados Compostos, Registos ("Records")

Exercício 2.1

1. Escreva sub-programas 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 acima. 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 2.2

1. Escreva sub-programas 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 sub-programas para:

   — escrever uma fracção;

   — somar duas fracções;

   — simplificar uma fracção.

   SUGESTÃO: Considere o tipo:

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