## Capítulos 11 + 12: Ordenar e Pesquisa + Entrada e Saída de Ficheiros

### Exercício 11.01 (Revisão)

Considere o sequinte `array` ordenado de inteiros:

| **index:** | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **value:** | 10 | 15 | 25 | 30 | 33 | 34 | 46 | 55 | 78 | 84 | 96 | 99 |

Aplicando o algoritmo de **pesquisa binária**, procure por `23`. Escreva o sequência dos elementos que são comparados e, para cada comparação, indique os valores de `low` e de `high`.

### Exercício 11.02 (Revisão)

Assumimos que todos os elementos do `array` são distintos; isto é, que não existem valores duplicados no `array`. Qual seria o efeito dum `array` com valores duplicados no algoritmo de **pesquisa linear**?

### Exercício 11.07 (Programação Nível 1)

Modifique o algoritmo ***bubble sort*** para ordenar dados `String`. Escreva um programa de teste para verificar se a sua implementação está correta.

### Exercício 11.08 (Programação Nível 1)

Implemente um programa que pede `N` palavras ao utilizador (`String`s) e escreve-as por ordem alfabética. Use o algoritmo de ordenação do exercício `11.06` ou o do exercício `11.07`.

### Exercício 11.10 (Programação Nível 2)

Outro algoritmo simples de ordenação é o ***insertion sort***.

Suponha que temos uma lista ordenada, `x`, com `N` elementos e que precisamos de inserir um novo elemento `X` na lista, de forma a ficarmos com uma lista ordenada com `N + 1` elementos.

Podemos inserir `X` na posição correta da lista comparando-o sucessivamente com os elementos `x[N - 1]`, `x[N - 2]`, `x[N - 3]` e por aí adiante. Cada vez que comparamos `X` com `x[i]` desviamos ("*shift*") `x[i]` para `x[i + 1]` se `X < x[i]`. Assim que encontrarmos um `x[i] < X` guardamos `x` na posição `i + 1` e paramos.

Podemos aplicar este processo para ordenar uma lista arbitrária de `N` elementos: Começamos com uma lista (ordenada) com um elemento e inserimos o segundo elemento para obtermos uma lista ordenada com dois elementos. Depois inserimos um terceiro elemento e obtemos uma lista ordenada com três elementos.

Escreva um programa que implemente este algoritmo. Pode simplificar assumindo que vai apenas ordenar listas de inteiros.

### Exercício 12.03 (Revisão)

Usando um bloco `try-catch`, escreva um fragmento de código que abre um certo ficheiro `default.dat` sempre que uma tentativa para abrir um ficheiro indicado pelo utilizador levantar uma exceção.

### Exercício 12.06 (Programação Nível 1)

Implemente um programa que pede palavras ao utilizador e guarda-as, uma por linha, num ficheiro de texto. O programa termina quando o utilizador escrever a palavra STOP (sem distinguir maiúsculas de minúsculas).

### Exercício 12.07 (Programação Nível 1)

Implemente um programa que lê as palavras do ficheiro de texto criado pelo exercício `12.06`. Depois de lido o conteúdo, escreve a palavra mais curta, a mais comprida e o comprimento médio das palavras.

### Exercício 12.10 (Programação Nível 1)

Implemente um programa que gera aleatoriamente N inteiros e que os guarda num ficheiro binário `integers.dat`. O valor de N é indicado pelo utilizador ou passado como argumento do programa.

Abra o ficheiro num editor de texto e observe o aspeto de um ficheiro binário.