Aula prática 3

A aula prática 3 tem o objetivo de aprofundar a utilização da biblioteca de vetores apresentada na aula prática 2. Adicionalmente, são abordados algoritmos de pesquisa e ordenação em vetores.

1 O ficheiro "**plantas.txt**" contém informação relativa às plantas existentes no Jardim Botânico do Porto. Usando a biblioteca de funções de vetores disponibilizada na aula prática 2, crie um programa que implemente as seguintes funcionalidades:

a) Carregue a informação relativa às plantas existentes no Jardim Botânico para um vetor (uma linha no ficheiro corresponde a uma planta) e indique quantas plantas existem.

```
Foram lidas 268 plantas!
Pos 0 - Quercus rubra L.
...
Pos 267 - Laburnum anagyroides (Medic.)
```

b) Indique em que posição do seu vetor se encontra a primeira ocorrência da planta "Podocarpus macrophyllus (Thunberg)".

```
A planta Podocarpus macrophyllus (Thunberg) encontra-se na posição 8.
```

c) Ordene alfabeticamente todas as plantas que existem no Jardim Botânico.

```
Vetor ordenado:
Pos 0 - Abelia x grandiflora (Andre) Rehder
...
Pos 267 - Viburnum opulus L.
```

d) Implemente uma nova função que conte o número de ocorrências de uma dada string no seu vetor. Verifique quantas ocorrências da planta "Ginkgo biloba (L.)" existem no Jardim Botânico.

Protótipo:	int vetor_num_ocorrencias(vetor *vec, char *pesquisa);
Argumentos:	**vec** vetor com o nome das plantas
	pesquisa string que contém a frase a pesquisar
Retorno:	Número de ocorrências da string pesquisa no vetor vec

```
A planta Ginkgo biloba (L.) aparece 2 vezes no vetor.
```

e) Crie uma função que recebe como parâmetro um vetor e remove todas as ocorrências repetidas. Usando esta função, verifique quantas plantas não repetidas existem no Jardim Botânico.

Protótipo:	void vetor_remove_repetidas(vetor *vec);
Argumentos:	**vec** vetor onde se pretende remover as ocorrências repetidas

```
Após remoção das plantas repetidas o vetor tem 120 posições.
```

f) Implemente uma função que lhe permita obter as plantas de um determinado género. Para tal, deverá fornecer o género a que a planta pertence, bem como o vetor com as plantas que serão alvo desta pesquisa. Por exemplo: uma pesquisa pelo género "Acacia" num vector que já não tem repetições deverá produzir um vector com 3 resultados: "Acacia retinodes Schlecht.", "Acacia nilotica (L.)Willd. ex Delile", "Acacia verticillata (L´Her.) Willd.",

Nota: Designa-se por "pesquisa parcial" quando é realizada uma pesquisa em que a "string" a ser pesquisada não precisa de corresponder inteiramente às "strings" alvo. Por exemplo, pesquisando por "Bun" numa qualquer lista de "strings", um resultado possível é "Bugs Bunny".

Protótipo:	vetor* vetor_pesquisa_parcial(vetor *vec, char *pesquisa);
Argumentos:	**vec** apontador para o vetor com as palavras a pesquisar **pesquisa** apontador para a palavra a procurar parcialmente
Retorno:	Apontador para um novo vetor que contenha os resultados da pesquisa parcial

2 Este exercício tem como objetivo desenvolver algoritmos genéricos de ordenação e pesquisa em vetores. Os algoritmos deverão ser genéricos tanto no tipo de dados que ordenam tal como na ordem que aplicam aos dados (ex., ordem ascendente, descendente, etc..). Estes algoritmos irão, de certa forma, replicar o comportamento das funções de ordenação e pesquisa da *stdlib*.

Note que, para implementar algoritmos genéricos no tipo de dados em C, é necessário recorrer a alguns "truques". Em particular, os dados são passados às funções neste exercício com **void***, ou seja, usando um **apontador genérico**. Estes apontadores poderão ser convertidos para o tipo de dados correto fazendo um *cast*. Adicionalmente, são usadas funções de comparação específicas para os diferentes tidos de dados, e que também são passadas às funções usando um **apontador para a função**.

Para compreender melhor estes conceitos, leia atentamente o ficheiro "ex2.c" onde irá encontrar a base para este exercício.

2.1 Implemente um algoritmo de pesquisa linear. Esta função deverá retornar um apontador para o elemento no vetor caso esteja presente, ou NULL caso contrário.

2.2 Implemente um algoritmo de pesquisa binária. Esta função deverá retornar um apontador para o elemento no vetor caso esteja presente, ou NULL caso contrário.

2.3 Implemente o algoritmo de ordenação por inserção.

2.4 Implemente uma função de comparação para "char" que ordene os elementos de forma descendente e posicione letras minúsculas antes das maiúsculas (eg: "AzbB" -> "zbBA"). Assuma que só existem caracteres alfabéticos (A->Z e a->z).

Outros exercícios: folha de exercícios extra.