

Exercícios – Extra

Esta folha de exercícios tem como objetivo complementar as fichas dadas nas aulas práticas 1 e 2. A gestão dinâmica de vetores é o tópico principal, tanto através da utilização das funções da biblioteca *standard C*, como da biblioteca de manipulação de *strings* disponibilizada na aula prática 2. É ainda proposto um exercício para praticar técnicas de *debugging*.

1 – Cada um dos segmentos de código (A a G) em comentário no ficheiro ***para_debug.c*** constitui um pequeno programa que contém um ou mais *bugs* que vão/podem despoletar erros em *run time* (i.e. durante a execução). Crie um projeto vazio no Eclipse e inclua aquele ficheiro. Depois, “descomentando” um segmento de cada vez, compile e corra o respetivo código, detete os *bugs* e corrija-os. Poderá encontrar algumas informações sobre como usar ferramentas de *debugging* no Eclipse no tutorial disponibilizado no Moodle.

2 – Crie um registo com nome `ComplexRect` que define o tipo de dados de números complexos na representação retangular. Um dos campos é a parte real e a outra é a imaginária. Da mesma forma crie um registo com o nome `ComplexAng` que define o tipo de números complexos na representação polar. Aloque um vetor de números complexos rectangulares que deverá ser preenchido pelo utilizador. Implemente uma função que converte os números complexos para a sua representação polar, módulo e ângulo. Utilize gestão dinâmica de memória.

Exemplo

```
Quantidade de numeros complexos a converter: 3
Insira os coeficientes das partes real e imaginaria...
...do 1º numero complexo: 1 1
...do 2º numero complexo: 0 1
...do 3º numero complexo: -2 0
As respectivas representacoes polares sao:
mod=1.41 ang=45
mod=1 ang=90
mod=2 ang=180
```

3 – Pretende-se criar um programa para analisar informação sobre as capitais de vários países do mundo e respetiva população.

- Defina os campos de uma estrutura chamada `capital`, necessários para armazenar a informação relativa a uma capital – o país (1 palavra), a própria capital (1 palavra) e o número de habitantes. Considere que o nome do país e da capital têm no máximo 50 caracteres.
- Implemente a função `analizarFicheiro` que lê um determinado ficheiro e devolve um vetor de registos e o número de capitais existentes.

```
capital* analisarFicheiro(FILE *ficheiro, int *num_capitais)
```

O vetor deverá ser alocado dinamicamente e conter um registo para cada capital encontrada. O número de capitais encontradas deverá ser indicado por `num_capitais`. A ordem das capitais é irrelevante nesta fase.

- c) Implemente a função `gerarRelatorio` que escreve para um ficheiro de saída um relatório dos resultados.

```
void gerarRelatorio(char *nome_ficheiro, capital *resultados, int
num_capitais)
```

A função deve escrever no ficheiro um cabeçalho e uma listagem, por ordem decrescente de população, das capitais guardadas no vetor resultados e respetivo número de habitantes. O tamanho do vetor resultados é indicado pelo argumento `num_capitais`.

Para o ficheiro de entrada **capitais.txt**, o ficheiro de saída **relatorio.txt** deverá apresentar o seguinte resultado:

```
Capitais:
1 - Toquio - Japao - 13185502
2 - Seul - Coreia_do_Sul - 10464051
3 - Moscovo - Russia - 10126424
(...)
99 - Tunes - Tunisia - 767629
100 - Asgabate - Turcomenistao - 763537
---
total: 100 capitais
```

- d) Escreva o corpo da função `main` de forma a que o programa implemente os seguintes passos:

- Pedir e ler nome de um ficheiro de entrada e outro de saída
- Ler o ficheiro de entrada – função `analisarFicheiro`
- Escrever no ficheiro de saída – função `gerarRelatorio`

4 – 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 vetor que já não tem repetições deverá produzir um vetor 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

5 [Hacker edition | difícil] – O ficheiro “IMBD.txt” contém informação relativa a um conjunto de filmes em formato de texto. Depois de analisar a forma como a informação está armazenada nesse ficheiro, crie um programa através dos seguintes passos:

- Altere o registo (**struct**) **elemento** da biblioteca de vetores de forma a permitir armazenar o número de **votos** (int), a **classificação** (float), o **título** (string) e o **ano** (int) de um filme;
- Altere a função **vetor_insere** para permitir que sejam inseridos no vetor todos os dados de um filme;

- c) Se necessário, altere também as restantes funções. Para simplificar, elimine da nova biblioteca de vetores de filmes as funções `vetor_remove`, `vetor_elemento`, `vetor_atribui` e `vetor_baralha`;
- d) Crie uma função que leia o ficheiro **“IMBD.txt”** e devolva um **vetor** de filmes devidamente preenchidos com as informações relativas a todos os filmes constantes do ficheiro lido;
- e) Crie uma função `vetor_ordena_ano` de forma a ordenar o vetor de filmes por **ordem cronológica** crescente (i.e., utilizando o campo **“ano”**). Baseie-se na abordagem usada para a anterior função `vetor_ordena`;
- f) Crie uma função `vetor_pesquisa_ano` que devolva um novo vetor com todos os filmes daquele ano. Baseie-se na abordagem usada para a função `vetor_pesquisa`;
- g) Crie e corra a função `main` de forma a testar as funções criadas.