Ficha de Trabalho

Objectivos: Estruturas dinâmicas: **Listas Ligadas Ordenadas**

Os exercícios propostos nesta ficha visam criar um programa que facilite a gestão de uma lista de palavras ordenadas (como num dicionário).

Considerações iniciais:
- A **língua por omissão** para uma lista de palavras é "**PT**";
- O **nome do ficheiro por omissão** é "**PT.txt**";

1. Defina uma estrutura **Palavra**, associando-lhe o tipo de dados ***ptPALAVRA**, com um campo texto (50 caracteres).

2. Defina uma estrutura **Elemento**, associando-lhe o tipo de dados ***ptELEMENTO**, adequada para a construção de uma lista de elementos simplesmente ligada com um campo informação do tipo **ptPALAVRA**.

3. Defina uma estrutura **Lista**, associando-lhe o tipo de dados ***ptLISTA**, com os dados:

a) Língua (2 caracteres);
b) Número de palavras (inteiro);
c) Elemento inicial da lista (**ptELEMENTO**).

4. Escreva uma função **ptLISTA criar_lista()** que crie (aloque e inicialize) uma lista de elementos.

5. Escreva uma função **ptELEMENTO criar_elemento()** que crie (aloque e inicialize) um novo elemento.

6. Escreva uma função **int validar_elemento(ptELEMENTO E)** que valide os dados de um dado elemento (todos os caracteres têm de ser letras não acentuadas).

7. Escreva uma função **void ler_elemento(ptELEMENTO E)** que registe os dados de um elemento e os valide.

8. Escreva uma função **void inserir_elemento_inicio(ptLISTA L, ptELEMENTO E)** que introduza um novo elemento no início da lista. Considere que o elemento já foi criado e que a respectiva informação já foi lida e validada usando as funções das 3 questões anteriores.

9. Escreva uma função **void inserir_elemento_fim(ptLISTA L, ptELEMENTO E)** que introduza um novo elemento no fim da lista.

10. Escreva uma função **int comparar_elementos(ptELEMENTO A, ptELEMENTO B)** que compare dois elementos.

11. Escreva uma função **int elementos_iguais(ptELEMENTO A, ptELEMENTO B)** que verifique a igualdade entre dois elementos.

12. Escreva uma função **void inserir_elemento_ordenado(ptLISTA L, ptELEMENTO ele_novo)** que introduza um elemento na lista, garantido a sua ordenação.

13. Escreva uma função **ptELEMENTO pesquisar_iterativo(ptLISTA L, ptELEMENTO ele_pesquisa)** que verifique se um determinado elemento pertence a uma lista utilizando uma abordagem iterativa.

14. Escreva uma função **ptELEMENTO pesquisar_recursivo(ptLISTA L, ptELEMENTO ele_pesquisa)** que verifique se um determinado elemento pertence a uma lista utilizando uma abordagem recursiva.

15. Escreva uma função **int elementos_repetidos(ptLISTA L)** que verifique se existem elementos repetidos numa lista.

16. Escreva uma função **void libertar_elemento(ptELEMENTO ele_libertar)** que liberte o espaço alocado para um dado elemento e respectivas informações associadas.

17. Escreva uma função **ptELEMENTO remover_elemento(ptLISTA L, ptELEMENTO ele_remover)** que remova um determinado elemento de uma lista.

18. Escreva uma função **ptELEMENTO remover_primeiro(ptLISTA L)** que remova o primeiro elemento de uma lista.

19. Escreva uma função **ptELEMENTO remover_ultimo(ptLISTA L)** que remova o último elemento de uma lista.

20. Escreva uma função **void mostrar_elemento(ptELEMENTO ele_mostrar**) que mostre os dados de um dado elemento.

21. Escreva uma função **void mostrar_ordenado(ptLISTA L)** que mostre os dados de todos os elementos de uma lista pela ordem actual.

22. Escreva uma função **void mostrar_inversa(ptLISTA L)** que mostre os dados de todos os elementos de uma lista pela ordem inversa.

23. Escreva uma função **void destruir(ptLISTA L)** que destrua (remova e liberte) todos os elementos de uma lista.

24. Escreva uma função **void exportar_ficheiro(ptLISTA L, char *nome_ficheiro)** que escreva todos os elementos de uma lista num dado ficheiro de texto.

25. Escreva uma função **void importar_ficheiro(ptLISTA L, char *nome_ficheiro)** que leia todos os elementos num dado ficheiro de texto e os coloque numa lista.

26. Escreva uma função **char menu_principal()** que permita ao utilizador escolher entre as diversas funcionalidades do programa.

```
(1) Inserir um novo elemento na lista
(2) Retirar elementos da lista ( ir para o sub menu)
(3) Mostrar os elementos pela ordem actual
(4) Mostrar os elementos pela ordem inversa
(5) Mostrar o numero de elementos da lista
(6) Pesquisar por um elemento da lista
(7) Verificar se existem elementos repetidos na lista
(8) Exportar elementos para um ficheiro de texto
(9) Importar elementos de um ficheiro de texto
(0) SAIR
```

27. Escreva uma função **char menu_remover()** que permita ao utilizador escolher as diversas funcionalidades específicas de remoção de elementos de uma lista.

```
(10) Retirar o primeiro elemento
(11) Retirar um elemento especificado pelo utilizador
(12) Retirar o último elemento
(13) Destruir a lista de elementos
( 0) VOLTAR ao Menu Principal
```

28. Escreva a função principal **void main()** de modo a integrar todas as funções elaboradas anteriormente.