Aula prática 3

A aula prática 3 tem o objetivo de introduzir uma estrutura linear alternativa aos vetores: as listas ligadas. Os exercícios são baseados numa biblioteca de funções que implementa uma lista duplamente ligada. Adicionalmente, a lista ligada é comparada com o vetor em algumas operações típicas.

1 – Analise a biblioteca de funções de listas ligadas de *strings* em C disponibilizada em "lista.zip". Tendo por base esta biblioteca de funções crie um programa de teste que efetue as seguintes funcionalidades:

a) O ficheiro "arquivo.txt" contém uma listagem de jogos antigos e a respectiva consola. Usando as funções disponibilizadas na biblioteca de funções leia o ficheiro e preencha uma lista ligada, onde cada elemento contém uma string com a informação contida em cada linha do ficheiro.

```
Foram carregados 43 jogos.
Pos 0 -> Grand Theft Auto: Liberty City Stories (PS2)
...
Pos 42 -> Shaun White Skateboarding (Wii)
```

b) Implemente uma função que pesquise uma *substring* na lista ligada, isto é, se entre as *strings* guardadas na lista ligada, alguma contém a *substring* que se procura. Por exemplo, a *string* "Minecraft" contém a *substring* "craft" mas não contém a *substring* "Mines". A pesquisa deverá ter como resultado uma nova lista contendo todas as *strings* da lista ligada original que contêm a *substring*.

Protótipo:	lista *lista_pesquisa_substring(lista * lst, char *substring);
Argumentos:	**lst** apontador para a lista **substring** string que se pretende pesquisar
Retorno:	apontador para uma nova lista com todas as *strings* que contêm a *substring*; se não existirem, retorna uma lista vazia

c) Pesquise e remova todos os jogos da "PS2". De seguida crie um novo ficheiro "jogos2.txt" com a nova lista.

```
Apos remocao de jogos PS2 a lista tem 33 jogos.
```

d) O ficheiro "oldies.txt" contém uma listagem de jogos mais antigos. Uma vez que pretende adicionar esta lista à sua lista inicial, crie uma função que lhe permite concatenar duas listas.

Protótipo:	lista* lista_concatena(lista *lst1, lista *lst2);
Argumentos:	**lst1** apontador para a primeira lista **lst2** apontador para a segunda lista
Retorno:	apontador para a lst1 com o resultado da concatenação das duas listas
Obs.:	a lista com o resultado da concatenação deverá ser a lst1 e não uma lista nova

```
Foram carregados 11 jogos antigos:
Pos 0 -> Stop the Express (ZX_SPC)
...
Pos 10 -> Roller Coaster (ZX_SPC)

Apos concatenacao, a lista final contem 44 jogos:
Pos 0 -> FIFA 12 (PS3)
...
Pos 43 -> Roller Coaster (ZX_SPC)
```

e) Ordene a lista de jogos.

```
Lista ordenada:
Pos 0 -> Dream Trigger 3D (3DS)
...
Pos 43 -> Stop the Express (ZX_SPC)
```

f) Indique em que posição da sua lista se encontra o jogo “Duke Nukem (PS3)”.

```
O jogo Duke Nukem (PS3) encontra-se na posicao 15.
```

g) Implemente uma função que permita inserir uma nova *string*, mantendo a ordem alfabética dos da sua lista.

Protótipo:	int lista_insere_ordenada(lista *lst, char *inserir);
Argumentos:	**lst** apontador para a lista **inserir** string a inserir na lista
Retorno:	-1 se inserir já fizer parte da lista ou a posição onde foi inserida se não fizer

```
O jogo Minesweeper (PC) foi inserido na posicao 12.
```

h) Antes de terminar o programa apague todas as listas que criou.

2 – Compare o desempenho das listas ligadas com os vetores, com base nas bibliotecas disponibilizadas nesta aula e na aula prática 2. Utilize o ficheiro “nomes.txt” como dados de entrada para inserir nas estruturas de dados. Crie um programa que teste as seguintes tarefas em ambas as estruturas de dados:

- Inserir todos os nomes do ficheiro no início;
- Aceder a 10000 elementos em posições aleatórias;
- Remover todos os elementos (um de cada vez) a partir do início;
- Inserir todos os nomes do ficheiro no fim.

Compare e interprete os resultados obtidos. Para medir o tempo, considere a função clock() da biblioteca time.h. Em baixo encontra um exemplo de utilização desta função.

```
clock_t start_t, end_t;
start_t = clock();

/* ... tarefa a testar ... */

end_t = clock();
printf("tempo: %.3f\n", (double)(end_t - start_t) / CLOCKS_PER_SEC);
```