Aula prática 5

A aula prática 5 tem o objetivo de rever a utilização e manipulação de vetores dinâmicos e listas ligadas, a matéria principal que será abordada na parte prática do Miniteste 1.

---

1 Pretende-se desenvolver um programa para gerar estatísticas mensais de temperatura a partir de um ficheiro com registos diários. O programa deve ser robusto relativamente a situações inesperadas, tais como erros na abertura de ficheiros.

1.1 Especifique uma estrutura ***leitura*** com os campos necessários para guardar um registo de temperatura, contendo mês, dia e temperatura.

1.2 Crie uma função `carregarHistorico` que carregue um histórico de temperaturas a partir de um ficheiro para um vector alocado dinamicamente.

```
leitura* carregarHistorico(const char *nome_ficheiro, int *nregistos);
```

O ficheiro a abrir é especificado como argumento e o número de registos lidos deve ser guardado numa variável passada por referência (***nregistos***). Cada linha do ficheiro tem o seguinte formato:

```
<temperatura> <dia> <mês>
```

1.3 Crie uma função `gerarEstatisticas` que produza estatísticas acerca do vector de leituras, guardando estas estatísticas num ficheiro de saída (nome passado como argumento). O protótipo da função a criar é o seguinte:

```
void gerarEstatisticas(const char *nome_ficheiro, leitura *leituras, int
                                                              nregistos);
```

As estatísticas a criar são, para cada mês, a temperatura máxima e média (por esta ordem). Exemplo (parcial) do relatório usando o ficheiro de entrada ***tempo2010.txt***:

```
Janeiro 14 10.064516
Fevereiro 14 10.535714
Marco 17 11.838710
Abril 21 15.966667
…
```

1.4 Complete o código do programa fornecido de forma pedir ao utilizador o nome de um ficheiro de entrada e um nome de ficheiro de saída, produzindo em seguida as estatísticas pedidas em 1.3 com base na leitura do ficheiro de entrada.

```
Ficheiro entrada? tempo2010.txt
Ficheiro saida? estatisticas.txt
Estatisticas geradas com sucesso.
```

2 Um sistema de gestão de dados sobre cinema está em desenvolvimento e é pedido nesta fase que se implementem algumas funções. Este sistema utiliza as bibliotecas de vetores e listas, que se encontram disponíveis nos ficheiros `vetor.h`, `vetor.c`, `lista.h` e `lista.c`. Sempre que achar conveniente utilize as funções já disponíveis nas bibliotecas.

2.1 Implemente uma função que devolva o número de ocorrências de um elemento:

```
int vetor_conta_ocorrencia(vetor* v, const char* valor)
```

A função deve devolver o número de ocorrências, ou -1 em caso de erro. Se o elemento não existir, a função deve retornar 0.

Depois de implementada a função, o programa deverá apresentar:

```
conta ocorrencia: 6 (esperado: 6)
conta ocorrencia: 0 (esperado: 0)
```

2.2 Implemente uma função que permita filtrar uma lista de acordo com a primeira letra de cada elemento:

```
vetor* lista_vetor_filtra(lista *lst, char l);
```

O resultado deverá ser um vetor contendo apenas as *strings* cuja primeira letra seja igual à letra definida no argumento da função. Garanta que o algoritmo implementado tem complexidade linear. Depois de implementada a função, o programa deverá apresentar:

```
Donnie Darko
District 9
...
Donnie Darko
```

2.3 Implemente uma função que, dadas duas listas, retorne uma nova lista com os elementos intercalados:

```
lista* lista_intercala(lista *l1, lista *l2)
```

Por exemplo, dadas as listas {"prog1", "prog2"} e {"prog3", "prog4", "prog5"}, pretende-se que o resultado seja {"prog1", "prog3", "prog2", "prog4", "prog5"}.
A função deve devolver a lista com os elementos intercalados.

Caso uma das listas seja NULL, deve devolver a outra. Se ocorrer um erro, deve devolver NULL.

Depois de implementada a função, o programa deverá apresentar:

```
270 (esperado: 270)
150 (esperado: 150)
150 (esperado: 150)
(null) (esperado: null)
285 (esperado: 285)
```

Outros exercícios: miniteste exemplo e exercícios 3 e 4 da ficha de exercícios extra.