Ficha de Trabalho n.º 1 (Revisão)

Objectivos: Revisão sobre ponteiros, memória dinâmica; utilização de ficheiros binários, métodos de ordenação e pesquisa.

Observação: O projecto deve ser implementado de um modo estruturado, com funções específicas para cada requisito e ficheiros separados para o programa principal e *headers* julgados convenientes.

Problema: Considere a estrutura de dados **REGISTO_UTILIZADORES** e um ficheiro binário com a designação "Utilizadores.dat" que contêm registos de utilizadores de um Site. Elabore um programa organizado em funções que disponibilize um conjunto de funcionalidades que terão como dados de trabalho um ficheiro de registos de utilizadores com muitos registos do tipo REGISTO_UTILIZADORES, com a informação apresentada na tabela abaixo.

Obs. A informação dos utilizadores deve depois ser registada em memória num vector/array, alocando somente o espaço necessário!

```
typedef struct
{
        int dia;
        int mes;
        int ano;
}DATA;
```

```
typedef struct
{
        char nome[50];
        char utilizador[20];
        char password[20];
        float joia;
        DATA data_registo;
        char email[30];
        char pagina_web_pessoal[50];
        int telemovel;
        int numero_acessos;
        DATA data_ultimo_acesso;
}REGISTO_UTILIZADORES;
```

Parte 1:

Implemente as seguintes funcionalidades, criando as funções com o nome indicado:

1. LerFicheiro; // Deve ler os dados do ficheiro;
2. Duplicar os dados da estrutura de dados original (com os dados lidos do ficheiro);
3. OrdenarDados; // Deve implementar um método de ordenação
4. LibertarMemoria; // Deve libertar toda a memória alocada
5. ListarUtilizadores (lidos do ficheiro); // Deve Listar todos os utilizadores do vector original (obtido no exercício 1)
6. ListarUtilizadores (depois de ordenados); // Deve Listar os utilizadores depois de ordenados
7. Criar um programa principal, de modo a poder chamar todas as funções desenvolvidas nas questões 1, 3, 5 e 6.

Parte 2:

8. Contar o número de pessoas que fizeram o último acesso ao site num dado ano (função: ContarPessoasAcessosAno);
9. Pesquisar pelo código (função: PesquisarCod);
10. Pesquisar pelo nome (função: PesquisarNome);
11. Determinar a pessoa que fez mais acessos (função: PessoaMais_Acessos);
12. Determinar a soma de todas as "joias" (função: TotalJoias);
13. Determinar qual o mês em que houve mais registos (função MesMaisRegistos);
14. Alterar o programa principal de modo a poder invocar as novas funcionalidades desenvolvidas.

Extras:

a) Como perceberam, o fich. *)"Utilizadores.dat" não foi fornecido. Mas, pretendem resolver o problema e poderem executar as opções da aplicação….
   Sugestão: Implementar uma função para gerar aleatoriamente muitos dados REGISTO_UTILIZADORES e gravar no ficheiro "Utilizadores.dat". Por exemplo, uma função com o seguinte protótipo:
   void GerarFicheiro_So_Para_Testes(char *nficheiro);

b) Elabore novas funções que permitam efectuar a ordenação dos dados utilizando outros algoritmos (ex. Selection Sort, Insertion Sort, Shell Sort), havendo, em qualquer das funções, um parâmetro que permite especificar qual o tipo de ordenação pretendida: (a)scendente/(d)escendente. Efectue também uma avaliação comparativa do desempenho de cada um dos algoritmos implementados;

c) Implemente funcionalidades para pesquisar por código ou nome, usando alguns dos algoritmos que conheça.
d) Crie uma funcionalidade que permita a importação de dados de utilizadores, utilizando um ficheiro texto de nome "utiliz.txt", com os dados correspondentes ao tipo REGISTO_UTILIZADOR;
e) Alterar o programa principal de modo a poder invocar as novas funcionalidades desenvolvidas.

Imagine que pretende eliminar/acrescentar algum REGISTO ….
Bom!, parece que está na altura de pensar uma nova estrutura de dados para guardar toda esta informação!!!