UNIVERSIDADE D
COIMBRA

# Manual do Programador

## PROJETO PRÁTICO DE PRINCÍPIOS DE PROGRAMAÇÃO PROCEDIMENTAL


**REALIZADO POR :**

Diogo Rodrigues - 2022257625
Maria Ferraz - 2023217813

**FCTUC - DEI**

# Índice

# Introdução

Visa-se com este projeto desenvolver uma aplicação que auxilie um médico na gestão dos seus doentes de forma digital.

A aplicação criada permite, de forma interativa, armazenar e gerir dados clínicos, nomeadamente através das seguintes operações:

- Introduzir dados de um novo doente;
- Eliminar todos os dados de um doente, com auxílio de um mecanismo de pesquisa simples (através do seu ID ou Nome);
- Apresentar uma lista de todos armazenados, por ordem alfabética;
- Mostrar uma lista com todos os doentes que possuem valores de tensão máxima acima de um valor solicitado ao utilizador (apenas os registos são apresentados por ordem decrescente);
- Apresentar toda a informação clínica de um doente (com auxílio da pesquisa referida anteriormente);
- Acrescentar dados clínicos à ficha de um doente.

Os dados dos doentes são armazenados diretamente em ficheiros de texto, que são carregados aquando do ínicio do programa e mantidos constantemente atualizados consoante as operações do utilizador.

# Aplicação

No âmbito da gestão eficiente e coerente dos dados foram definidas as seguintes estruturas: Data; Doente; l_node_doente_t; lista_doentes_t; Registo; l_node_registo_t; lista_registos_t. Encontram-se devidamente inicializadas nos respectivos ficheiros em “/lib/”. Todas as funções que interagem diretamente com as listas (de doentes/registos) encontram-se inicializadas e definidas em lista[registos/doentes].h e lista[registos/doentes].c, respetivamente. Funções que envolvam leitura de ficheiros ou entrada stdin (ainda que afetem a lista indiretamente) encontram-se, por sua vez, inicializadas e definidas em [doentes/registos].h e [doentes/registos].h. Outras funções de caráter genérico podem ser encontradas inicializadas e definidas em misc.h e misc.c, respetivamente.

Procurou-se alcançar uma maior modularidade e uniformidade do programa ao utilizar diversas funções para a receção de determinados dados sem repetição de código.

O programa corre indefinidamente até ser devidamente terminado recorrendo à opção de saída presente no menu, a abordar mais à frente.

# Carregamento de dados

Seguindo uma abordagem cronológica no que diz respeito à execução do programa, é inicializada a lista de doentes, seguida da chamada a função carregarDoentes() à qual é fornecido o ponteiro da lista e onde são tratados os respectivos conteúdos do ficheiro “doentes.txt”, linha a linha (onde se encontram os dados pessoais dos ficheiros). Os dados lidos e tratados são guardados numa estrutura “Doente” adequada onde também é guardado um ponteiro que refere a respetiva lista de registos (devidamente inicializada), estando por isso em condições de ser inserido num novo nó da listaDoentes. De seguida, é lido e processado o ficheiro de registos (carregarRegistos()) que guarda cada registo em um nó da lista de registos de cada doente.

A lista de registos encontra-se referenciada por um ponteiro, presente em cada nó da lista de doentes. A lista de registos de cada doente encontra-se organizada segundo duas ordens possiveis: por ordem cronológica ou ordem de tensão máxima decrescente. Ambas são acessiveis a partir dos respectivos ponteiros listaRegistos->next e listaRegistos->nextTensao.

Em todas as listas (doentes/registos), a inserção de novos nós é feita de modo a que estes sejam inseridos diretamente na posição por forma a que fiquem devidamente ordenados nas respectivas listas (em todas as formas de as percorrer).

As funções responsáveis pelo carregamento de todos os dados são a carregarDoentes() e carregarRegistos().

# Introdução de dados

Para minimizar a introdução de dados incorrentos/inválidos, implementaram-se mecanismos de proteção para a introdução dos seguintes dados:

- Nome: apenas são aceites entradas que contenham caracteres [a-z] e [A-Z];
- Data de nascimento: são apenas aceites entradas com 2 separadores (“-”), que passem a verificação se a data existe (por exemplo através da verficação de um mês e dia com um ano bissexto/comum: 29/02/2023 não seria aceite porque 2023 não é bissexto; verificação da correspondência dia/mês; verificação dos caracteres introduzidos);
- Cartão de cidadão: foi implementada uma verificação da validade através algoritmo disponibilizado oficialmente, além da verificação da integridade geral dos caracteres introduzidos;
- Número de telefone: apenas são aceites número inteiros de 9 digitos cujo primeiro seja 9;
- Endereço de Email: é verificada a existência de “@” e validade do domínio (de forma simples) através do posicionamento do “.” (são recusadas entradas com “.@” ou “@.”;

- Para a introdução de novos registos clínicos, apenas são aceites datas (seguindo os mesmos critérios da data de nascimento) e números inteiros para os vários campos;

Com o objetivo de tornar as entradas de dados uniformes, foram desenvolvidas as funções receberData(), receberInteiro(), receberNome(), receberCC(), receberTel(), receberEmail() que têm em conta os aspetos de segurança referidos anteriormente e permitem uma maior modularidade para o possível desenvolvimento de novas funcionalidades.

# Menu


O menu desenvolvido apresenta as diferentes opções disponiveis e aguarda a introdução de um número de 1 a 7 pelo utilizador.

1. Insere um novo doente (solicitando ao utilizador todos os dados necessários para o efeito);
2. Eliminar um doente (permite ao utilizador pesquisar o doente alvo através do seu nome e/ou ID);
3. Listagem de todos os doentes por ordem alfabética;
4. Listagem de doentes com tensões máximas acima de um valor especificado pelo utilizador;
5. Apresentar toda a informação de um determinado doente (à semelhança da eliminação de um doente, é possível efetuar uma pesquisa do doente alvo);
6. Introdução de um novo registo (também à semelhança da eliminação de um doente, é possível efetuar uma pesquisa do doente alvo);
7. Terminar a execução do programa.

## Inserção de novos doentes

A inserção de novos doentes é efetuada após a receção/determinação de todos os dados a inserir na estrutura novoDoente (através da função receberNovoDoente(). Feito isto, é chamada a função inserirNodeDoente() que aloca memória para um novo nó e determinada a posição na lista onde inserir o novo nó recorrendo à função procurarNodeDoente(), a lista é mantida permanentemente ordenada alfabeticamente (conforme os nomes dos doentes). Encontrados os nós, atualizam-se os respetivos ponteiros de modo a que o novo nó faça parte da lista. Por fim, recorre-se à função appendFicheiroDoentes() que simplesmente acrescenta o novo doente ao final do ficheiro de doentes.

## Remoção de um doente

A remoção de um doente do sistema consiste essencialmente na eliminação do nó do respetivo doente da lista. Após o apuramento do doente a remover, segundo a entrada do utilizador, recorre-se à função removerNodeDoente() que recorrendo ao ID único do doente no sistema, procura o nó correto, altera as ligações dos nós e liberta a memória alocada para o doente removido. Por fim, reescrevem-se os ficheiros dos doentes e dos registos com todos os conteúdos (posteriores à remoção) da lista de doentes e respectivas listas de registos no ficheiro adequado.

## Listagem de doentes

A listagem de doentes por ordem alfabética é apenas a apresentação dos conteúdos da lista de doentes. Uma vez que é mantida permanentemente ordenada (através da inserção dos nodes segundo ordem alfabética), percorrer a lista de doentes e apresentar os conteúdos de cada nó irá resultar na apresentação de todos os doentes por ordem alfabética

A listagem de doentes com uma tensão acima a um valor, por sua vez, consiste na chamada da função listarDoentesTensaoAcimaLimite() à qual é fornecido o valor inteiro “limite”, devidamente solicitado ao utilizador e posteriormente tratado. Esta função irá percorrer a lista de registos de cada doente da lista de doentes (por ordem alfabética) e verificar se existem nós que verifiquem a condição desejada. Se existirem nós nestas condições, é apresentada a identificação do doente seguido dos conteúdos dos nós de registo em causa. É de notar que, de modo a que os nós dos registos sejam apresentados por ordem decrescente, a lista de registos é percorrida de forma alternativa (à ordem cronológica) através dos ponteiros auxiliares que permitem percorrer cada lista de registos pela ordem de grandeza das tensões.

## Inserção de novos registos

A inserção de novos registos requer do utilizador um doente alvo e de seguida todas as informações necessárias para preencher um novo registo. A função inserirNodeRegisto() irá encarregar-se de inserir o registo na lista (modificando as respetivas ligações dos nós - os ponteiros relativos à ordem cronológica e de decrescentes tensões). Terminada esta tarefa, acrescenta-se o novo registo ao ficheiro de registos.

## Conclusão

No desenvolvimento deste projeto procurou-se otimizar o uso de memória assim como a eficiência da execução através da minimização de ordenações de dados e operações de leitura/escrita de ficheiros.