FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Licenciatura em Engenharia Informática
Ano letivo 2023/2024

MANUAL DO PROGRAMADOR

no âmbito da cadeira de
Princípios de Programação Procedimental

Trabalho realizado por
Henrique Diz (2023111111)
Tomás Gonçalves (2019232712)

ÍNDICE

INTRODUÇÃO

O presente Manual do Programador pretende clarificar a implementação dos componentes no software "S_MED".

Este projeto teve como objetivo desenvolver um sistema de gestão de Pacientes de um Hospital, que permita realizar operações como:

(i) Adicionar um Paciente
(ii) Remover um Paciente
(iii) Consultar os Pacientes inseridos na Database
(iv) Consultar Pacientes por Tensão Máxima
(v) Adicionar um novo Registo de um Paciente
(vi) Consultar a Informação total de um Paciente
(vii) Consultar o nº total de Pacientes dos HUC.

O programa adotou uma abordagem modular para assegurar um funcionamento eficiente. Está organizado em diversos ficheiros, cada um responsável por partes específicas do sistema, facilitando assim a gestão e compreensão das suas funcionalidades.

A gestão de pacientes, implementada no arquivo "*pacientes.c"*, permite a adição de novos pacientes ao sistema, procura de pacientes pelo nome, listagem de todos os pacientes e a sua remoção. Para cada paciente, é armazenado: nome, data de nascimento, cartão de cidadão, e-mail, telefone, ID e todos os registos associados a este.

A gestão dos registos de cada paciente, por sua vez, é implementada em "*registos.c"*. Este módulo permite a adição de registos médicos específicos a cada paciente, procurar registos por data e a listagem dos registos de acordo com critérios como tensão arterial acima de um determinado valor. Cada registo inclui informações detalhadas como data de registo, tensões arteriais, peso e altura do paciente.

O sistema também inclui funções auxiliares, implementadas em "*funcoes_auxiliares.c"*, responsáveis pela verificação dos dados introduzidos pelos usuários. Estas funções garantem que os dados são introduzidos de forma correta consoante o seu tipo, como verificação de emails, números de cartão de cidadão e formatos de datas, aumentando a robustez e a segurança do sistema.

As funcionalidades da aplicação, como adicionar pacientes, procurar pacientes, listar pacientes e remover pacientes e registos são então centralizadas no arquivo "*funcionalidades.c"*, onde está presente a lógica principal do programa.

Para o desenvolvimento deste projeto, os IDE escolhidos foram o *VSCode* e o CLion.
Para armazenar e controlar as várias versões da aplicação, foi criado um repositório no *GitHub* que pode ser acedido a partir do seguinte link.

Ficheiro "funcionalidades.c"

O ficheiro "*funcionalidades.c"* é o ficheiro principal na gestão de pacientes e registos médicos.

Aqui definimos as principais funções que vamos utilizar, com uma interface organizada para as funcionalidades do sistema.

Uma das principais funções implementadas permite ao user inserir novos pacientes no sistema. Esta função solicita dados como nome, data de nascimento, número do cartão de cidadão, e-mail, entre outros, e utiliza verificações de validade para garantir que esses dados estejam em formatos corretos antes de adicionar o paciente à base de dados.

Outra função importante é a de listar pacientes, que percorre a base de dados e imprime as informações relevantes de cada paciente.

Além disso, com a função de remover pacientes, possibilitamos a remoção de um paciente da base de dados após localizá-lo e, se encontrado, ajustando a estrutura de dados conforme necessário para manter a integridade da base de dados.

Para gerir os registos médicos, implementámos uma função que adiciona novos registos de saúde para pacientes existentes. Cada registo inclui informações como data, tensões arteriais máxima e mínima, peso e altura.

A função verifica se o paciente ao qual o registo será associado existe, e que os dados do registo são válidos. Similar à lista de pacientes, há uma função para listar registos médicos, que percorre os registos médicos e os mostra organizados.

ESTRUTURAS

DATA

- Representada pela estrutura *data*, que contém três inteiros (*dia, mes e ano*).

REGISTO

- A estrutura *registo* contém uma data (*data_registo*), dois inteiros representando a tensão arterial (*tensao_min* e *tensao_max*), o peso e a altura do paciente.

BLOCO REGISTOS

- A estrutura *bloco_registo* é um nó que armazena um registo de paciente e um ponteiro para o próximo nó (*prox)*. É então utilizada para formar uma lista ligada de registos (*registos*)

INFO

- A estrutura *info* guarda as informações pessoais dos pacientes, como data de nascimento (*data_nascimento*), número do cartão de cidadão, email, nome, telefone, id e a lista de registos (*pessoa_registo*).

BLOCO PACIENTES

- A estrutura *bloco* é um nó que armazena as informações de um paciente (*info*) e um ponteiro para o próximo nó (*prox*). Utilizada para formar uma lista ligada de pacientes (*PACIENTES*).

FICHEIROS E FUNÇÕES PRINCIPAIS

Ficheiro “pacientes.c”

1. **cria_pacientes**: Inicializa a lista de pacientes com um nó de cabeçalho que contém um paciente (Header) com informações padrão e id = 0 (usamos este id para armazenar o número total de pacientes).
2. **procura_paciente**: Procura um paciente na lista por nome e retorna o nó anterior e o nó atual da posição encontrada.
3. **insere_pacientes**: Insere um novo paciente na lista de forma ordenada alfabeticamente pelo nome.
4. **destroi_pacientes**: Liberta a memória ocupada pela lista de pacientes e pelos registos.
5. **load_pacientes**: Carrega os dados dos pacientes a partir de um ficheiro de texto ("doentes.txt") e insere na lista.
6. **save_pacientes**: Guarda os dados dos pacientes da lista no ficheiro de texto ("doentes.txt").

Ficheiro “registos.c”

1. **cria_registos**: Inicializa a lista de registos de um paciente com um nó de cabeçalho que contém um registo padrão.
2. **procura_registo**: Procura um registo na lista por data e retorna o nó anterior e o nó atual da posição encontrada.
3. **insere_registo**: Insere um novo registo na lista de registos de um paciente de forma ordenada cronologicamente.
4. **destroi_registo**: Liberta a memória ocupada pela lista de registos de um paciente.
5. **load_registos**: Carrega os registos dos pacientes a partir do ficheiro de texto ("registos.txt") e insere nas listas de registos dos pacientes.
6. **save_registos**: Guarda os registos dos pacientes da lista no ficheiro de texto ("registos.txt").

Função “running”

A função *running* recebe apenas um ponteiro para a lista de pacientes (*PACIENTES*).

Começa com um loop *while* que é executado indefinidamente, permitindo que o sistema continue em execução até que o user escolha a opção para sair. O menu principal é exibido nesta função, mostrando as opções disponíveis para o user, como adicionar paciente, eliminar pacientes, listar pacientes, etc…

Ao exibir o menu, usamos um inteiro “*choice”* para obter a escolha do user. Em seguida, um bloco *switch* é utilizado para determinar qual ação deve ser tomada com base na opção selecionada.
Cada caso no *switch* corresponde a uma opção do menu e chama a função correspondente para executar a ação desejada.

Dentro de cada caso do *switch*, as funções apropriadas são chamadas para manipular os dados de acordo com a escolha do usuário.

Por exemplo, se o usuário escolher a opção para adicionar um novo paciente, a função “*add_patient()”* será chamada para solicitar e adicionar os detalhes do paciente à lista.

As funções para manipular os pacientes (como adicionar, buscar, listar, etc.) são implementadas em outros ficheiros do código e são chamadas conforme necessário.

Após a execução de algumas operações que alteram a lista de pacientes (por exemplo, adicionar ou excluir pacientes), o número total de pacientes e as informações nos ficheiros são atualizadas. Isso garante que outras partes do sistema tenham acesso ao número atualizado de pacientes.

Quando o user escolhe sair do sistema, o loop *while* é interrompido e encerramos o programa através do *return.*

FUNÇÕES AUXILIARES

Ao longo do desenvolvimento do projeto, foi tida em atenção a necessidade de proteger o programa, tanto de eventuais erros devolvidos por funções utilizadas, como de *inputs* do utilizador fora do que seria esperado.

Para minimizar então estes problemas por parte de utilizador, estão implementadas alguma funções como:

i) **verifica_numeros**

Esta função tem como objetivo verificar se uma string contém apenas dígitos numéricos. Percorre cada char da string, e se encontrar algum char que não seja dígito (usando a função *isdigit*), a função retorna 0, indicando que a string não é totalmente numérica. Caso todos os chars sejam digitos, a função retorna 1.

ii) **verifica_string**

Esta função verifica se uma string contém apenas letras e espaços. Semelhante à função anterior, percorre cada caractere da string. Se encontrar algum caractere que não seja uma letra (usando a função "*isalpha" da biblioteca "ctype.h"*) ou um espaço, a função retorna 0. Caso contrário, retorna 1, indicando que a string contém apenas letras e espaços.

iii) **verifica_email**

Esta função verifica se a string fornecida é um email válido. A verificação é baseada na presença de um único caractere '@' seguido por um ponto '.' em algum lugar após o '@', que são requisitos básicos para um email válido. Retorna 1 (verdadeiro) se ambos forem encontrados, indicando que o email é válido. Caso contrário, retorna 0 (falso).

iv) **verifica_cartao_cidadao**

Esta função verifica se a string fornecida representa um número de cartão de cidadão válido. O número deve ter exatamente 9 dígitos, sempre.

v) **limpar_buffer**

Esta função é utilizada para limpar o buffer de entrada do teclado. Após a leitura de uma entrada do user, especialmente quando se usa a função *scanf*, pode haver caracteres indesejados no buffer (como o char de nova linha, por exemplo). Esta função utiliza um laço para consumir e descartar esses caracteres até encontrar o fim do buffer ou um caractere de nova linha.

vi) **find_id**

Esta função procura um ID específico dentro do array das estruturas de pacientes e retorna o ponteiro para o nó correspondente, caso o ID seja encontrado. Caso contrário, retorna -1.

vii) **input_numeros**

Esta função obtém o input do user, garantindo que a entrada seja composta apenas por números com um número máximo constituído por 10 dígitos. Solicita ao user valores até que uma entrada válida seja recebida.

viii) **input_data**

Esta função obtém a entrada do user para uma data, garantindo que a data esteja no formato correto "DD/MM/AAAA" e dentro dos parâmetros válidos para cada um dos formatos. Valida a formatação e a validade dos valores de dia, mês e ano.

ix) **input_strings**

Esta função obtém a entrada do user, garantindo que a entrada seja composta apenas por letras e espaços. Solicita ao user até que uma entrada válida seja recebida.

CONSIDERAÇÕES FINAIS

Este manual do programador apresenta um guia detalhado para o desenvolvimento de um sistema de gestão de pacientes. Desde estruturas de dados até implementações de funções específicas para validação e manipulação de inputs, o manual cobre todos os aspectos críticos necessários à compreensão deste projeto.

Ao seguir os exemplos fornecidos, os programadores podem assegurar que os inputs dos users sejam rigorosamente validados, e que o sistema funcione de maneira eficaz e segura.

A implementação de funções auxiliares, como verificação de email, validação de número de cartão de cidadão e verificação de datas, exemplificam boas práticas de programação e facilitam a manutenção e expansão do sistema no futuro.

Além disso, a robustez dos dados garante a integridade das informações dos pacientes, minimizando erros e inconsistências. Este cuidado é essencial em qualquer sistema que lida com dados sensíveis e pessoais, como os de saúde.