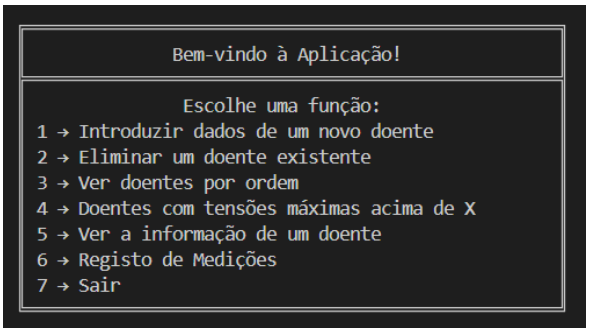
Manual do Programador - PPP

Introdução:

Foi-nos pedido a construção de uma aplicação que faça com que um utilizador consiga gerir os dados de um doente , podendo ser a informação de um doente (nome, data de nascimento, cartão de cidadão, telefone, email) ou o registo dos seus dados clínicos s (tensões arteriais – mínima e máxima –, peso e altura) , diante isso , construímos uma aplicação que se encontra dividido em 3 ficheiros : “projeto.c” , “projeto structs.h” , “projeto funcoes.c”.



Estrutura e Modo de Utilização:

O programa apresenta uma interface onde apresenta um menu com as 7 opções que o utilizador pode fazer.

A primeira opção permite um utilizador introduzir um novo doente na plataforma , onde serão pedidas , de forma segura e verificadas , as informações do doente tais como: (nome, data de nascimento, cartão de cidadão, telefone, email). Ao fim da introdução do doente irá ser retornado para o menu principal.

```
Opção:1
►          INSERIR DOENTE          ◄
Introduza os dados do novo doente:
Nome: Rodrigo
Telemóvel: 961234567
CC: 20242024
Email: rodrigo@gmail
Data de Nascimento (dd mm aaaa): 12 04 2004
```

A segunda opção permite eliminar um doente existente , mostrando ao utilizador todos os doentes existentes e os seus respectivos ID 's que irão servir de fator para a eliminação desse mesmo doente, assim o utilizador terá de escrever o ID respectivo ao doente que quer eliminar.

```
Opção:2
►          ELIMINAR DOENTE          ◄
1
Ana Maria Silva
3
Andre
2
Paulo Silva
4
Rodrigo
ID:
```

A terceira opção permite ver a lista de doentes existente mas por ordem alfabética , sendo apenas necessário escrever o número 3 e será apresentada a lista por ordem alfabética com as suas respectivas informações e registos.

```
Opção:3
►          LISTA ORDENADA          ◄
1
Ana Maria Silva
30/10/2000
939786543
anaSilva@dei.uc.pt

3
Andre
21/7/2004
967755623
ande@dada

2
Paulo Silva
3/10/2002
960786543
pSilva@dei.uc.pt
```

A quarta opção serve para o utilizador saber quais os doentes que apresentam o valor da tensão máxima acima de um certo valor , esse valor é devidamente pedido ao utilizador e de seguida será apresentado os ID 's dos doentes que seguem essa condição e a respectiva tensão máxima.

```
Opção:4
►          DOENTES ACIMA DA TENSÃO MÁXIMA          ◄
Diga qual o valor mínimo da tensão máxima pretende: 100

Id: 1
Tensão Máxima: 128
```

A quinta opção mostra ao utilizador todas as informações e todos os registos que um doente tem , para isso , é necessário o ID do doente de modo a só apresentar as informações desse mesmo , ao fim de ser inserido , é mostrada toda essa informação do doente e os seus registos clínicos.

```
Opção:5
►          INFORMAÇÃO DE UM DOENTE          ◄
Diga o ID do doente que quer ver: 2
2
Paulo Silva
3/10/2002
960786543
pSilva@dei.uc.pt
```

A sexta opção faz com que o utilizador possa inserir o registo clínico de um doente que já exista na aplicação , ao digitar o número 6 , serão pedidos de forma segura : (tensões arteriais – mínima e máxima, peso e altura) , no entanto se o ID não existir será exibido uma mensagem a dizer que ele não existe e o registo não será efetuado , caso contrário o registo é devidamente associado ao doente com o ID anteriormente colocado.

```
Opção:6
►          INSERIR REGISTO DE DOENTE          ◄
Introduza os dados do registo do doente:
ID: 1
Data de Registo (dd mm aaaa): 22 05 2024
Tensão Máxima: 140
Tensão Minima: 67
Peso: 77
Altura: 176
```

Por fim , a sétima opção permite simplesmente ao utilizador fechar o programa em segurança.

```
7 → Sair

Opção:7
A encerrar a Aplicação...
```

Ficheiros Utilizados:

Ao longo do programa vamos utilizar dois ficheiros do tipo .txt , que são eles doentes.txt e registos.txt, estes são carregados pela aplicação ao início e permitem carregar e guardar respectivos dados (doentes , registos). As funções ler Doente e ler Registos tratam de forma segura e correta da leitura dos ficheiros e respetiva colocação nas listas.

```
void lerDoente(pLista lista){
    FILE *ficheiro;
    FILE *ficheiro_temp;
    ficheiro = fopen("doentes.txt","r");
    ficheiro_temp=fopen("doentes_temp.txt","w");
    if(ficheiro==NULL || ficheiro_temp == NULL){
        printf("ficheiro não foi aberto \n");
        return;
    }
    char linha[TAM];
    int conta_linha=0;
    int adicionado=1;
    while (fgets(linha,TAM,ficheiro)!=NULL)
    {
```

```
    fclose(ficheiro);
    fclose(ficheiro_temp);
    remove("doentes.txt");
    rename("doentes_temp.txt","doentes.txt");
}
```

```
void lerRegisto(pLista lista){
    FILE *ficheiro;
    FILE *ficheiro_temp;
    ficheiro = fopen("registos.txt","r");
    ficheiro_temp=fopen("registos_temp.txt","w");
    if(ficheiro==NULL || ficheiro_temp == NULL){
        printf("ficheiro não foi aberto \n");
        return;
    }
    char linha[TAM];
    int conta_linha=0;
    int adicionado=1;
    while (fgets(linha,TAM,ficheiro)!=NULL)
    {
```

```
    }
    fclose(ficheiro);
    fclose(ficheiro_temp);
    remove("registos.txt");
    rename("registos_temp.txt","registos.txt");
}
```

Estruturas de Dados Utilizadas:

Neste trabalho foi usado uma lista ligada para os doentes , e cada nó da lista (doente) contém uma lista ligada dentro que corresponde aos registos de cada doente.
Para inserirmos doentes na lista ligada respetiva, usamos a função inserir
que permite inserir por ordem alfabética. A função garante sempre a alocação de memória para um novo nó. Se a alocação falhar, uma mensagem de erro é exibida. Caso contrário, o novo nó é iniciado com os dados do paciente e é inserido na lista.
Se a lista estiver vazia, o novo nó representa o primeiro elemento. Se a lista já tiver doentes, a função percorre a lista para encontrar a posição correta, mantendo-a ordenada alfabeticamente pelos nomes dos pacientes. Os ponteiros auxiliares (ant e atual) são usados para procurar a posição de inserção. A função compara os nomes usando strcasecmp. Já na lista ligada de cada doente(registos) , a inserção é feita de modo a que a lista esteja ordenada por ordem decrescente da tensão máxima , de modo a facilitar algumas funcionalidades do programa . Sendo assim é usada a função insere_registo que apresenta as mesmas características da função insere , diferenciando na forma em que há a comparação entre nós para a inserção , esta verifica se até onde o valor do registo inserido é maior do que os já existem e é inserido corretamente. A complementar estas funções são usadas outras como a função procura , função elimina e função vazia.

```
}data;
typedef struct registo{
    int id,tensao_max,tensao_min,peso,altura;
    data data_registo;

}registo;
typedef struct noListaregisto{
    struct registo registoLista;
    struct noListaregisto *prox;
}noListaregisto;
typedef noListaregisto *rLista;
typedef struct doente
{
    int id,telemovel;
    char nome[tam],cc[tam],email[tam];
    data data_nascimento;
}doente;
typedef struct noListadoente
{
    struct doente doenteLista;
    struct noListadoente *prox;
    rLista listaRegisto;
}noListadoente;

typedef noListadoente *pLista;
```

```
void insere(pLista lista, struct doente d)
{
    pLista no = (pLista)malloc(sizeof(noListadoente));
    if (no != NULL)
    {
        no->doenteLista = d;
        no->prox = NULL;
        if(lista->prox==NULL){
            lista->prox=no;
        }
        else{
            pLista ant=lista;
            pLista atual= lista->prox;
            while(atual!=NULL && strcasecmp(atual->doenteLista.nome,d.nome)<0){
                ant=atual;
                atual=atual->prox;
            }
            ant->prox=no;
            no->prox=atual;
        }
    }
    else{
        printf("Erro a alocar memória para o nó");
    }

}
```

```
void insere_registo(pLista lista, struct registo r){
    pLista aux = lista->prox; // Dá skip ao header
    while (aux!=NULL)
    {
        if(aux->doenteLista.id==r.id){
            noListaregisto *no = (noListaregisto*)malloc(sizeof(noListaregisto));
            if(no==NULL){
             printf("Erro a alocar memória");
            }
            no->registoLista=r;
            noListaregisto *ant=NULL;
            noListaregisto *atual = aux->listaRegisto;
            while (atual!=NULL &&atual->registoLista.tensao_max>r.tensao_max)
            {
                ant=atual;
                atual=atual->prox;
            }
            if(ant==NULL){ // Ou seja lista vazia

                no->prox=aux->listaRegisto;
                aux->listaRegisto=no;
             }
            else{
             no->prox=atual;
             ant->prox=no;
            }
            return;
        }

        aux=aux->prox;
    }
    printf("Id não existe\n");
```

Principais funções criadas

A aplicação apresenta várias funções , além de inserir os doentes e registos que foram anteriormente abordadas , sendo que algumas apresentam de certa forma um grau de importância superior que são as funções que representam o pedido do utilizador, são : imprime_alfabetica , listar_tensao_max , imprime_dados.

A função imprime_alfabetica(pLista lista) , que representa a função 3 que é ver os doentes por ordem alfabética , percorre a lista ligada dos doentes que , ao já estar ordenada por ordem alfabética devido à inserção característica , facilita e faz com que a função apresenta só uma série de prints referentes a cada atributo presente no nó lido. Assim esta função apresenta todos os Doentes e as suas informações de forma ordenada.

```c
void imprime_alfabetica(pLista lista) // imprimir todos os dados incluindo os registos de um determinado doente
{
    pLista aux = lista->prox; /* Salta o header */
    while (aux)
    {
        printf("%d\n",aux->doenteLista.id);
        printf("%s \n", aux->doenteLista.nome);
        printf("%d/%d/%d \n",aux->doenteLista.data_nascimento.dia,aux->doenteLista.data_nascimento.mes,aux->doenteLista.data_nascimento.ano);
        printf("%d \n", aux->doenteLista.telemovel);
        printf("%s \n", aux->doenteLista.email);
        printf("\n");
        aux = aux->prox;
    }
}
```

A função listar_tensao_max(pLista lista,int valor) , representa a função 4 que é ver doentes que apresentem um valor de tensão máxima superior a esse valor , recebe a lista dos doentes e um valor inteiro escrito pelo utilizador , e o que a função faz é , visto que cada lista ligada dos registos correspondente a um doente está ordenada por ordem decrescente (tensão máxima) , o primeiro valor de cada lista representa assim o maior valor de tensão máxima de cada doente. Posto isto o primeiro passo é ir buscar esse primeiro valor e comparar com o valor que foi inserido pelo utilizador , se for maior adicionamos aos registos , senão passamos para o próximo doente. Ao fim de ter os valores das tensões máximas de cada doente que são superiores ao valor inserido , o passo seguinte da função é ordenar os registos de modo decrescente para quando for feita a impressão ser apresentado primeiramente o doente que tem um valor de tensão máxima superior ao valor pedido e também superior a todos os outros valores. Posto isto a função percorre os registos e faz o print ordenado.

```c
void listar_tensao_max(pLista lista,int valor){
    struct registo registos[50];
    int j = 0;
    pLista aux = lista->prox; // Salta o header
    while(aux){
        //Vai ao primeiro elemento da lista registo (maior) e vê se é maior que o valor pedido
        noListaregisto *registo_atual = aux->listaRegisto;
        if (registo_atual && registo_atual->registoLista.tensao_max > valor) {
            registos[j] = registo_atual->registoLista;
            j++;
        }
        aux = aux->prox;
    };
    //ordenar os registos de maneira decrescente
    for (int i = 1; i < j; i++) {
        struct registo registo_atual = registos[i];
        int j = i - 1;
        while (j >= 0 && registos[j].tensao_max < registo_atual.tensao_max) {
            registos[j + 1] = registos[j];
            j = j - 1;
        }
        registos[j + 1] = registo_atual;
    }
    for (int i = 0; i < j; i++) {
        struct registo r = registos[i];
        printf("\nId: %d\n", r.id);
        printf("Tensão Máxima: %d\n", r.tensao_max);
    }
}
```

A função imprime_dados(pLista lista, int id) , equivalente ao número 5 , recebe a lista dos doentes e tal como o nome diz , ao receber um ID , irá percorrer a lista de modo a verificar se há um doente com esse respetivo ID , e caso seja sucedido , dá conta de imprimir as informações presentes na programa , quer seja as informações gerais do doente , como todos os registo que o doente já fez , mostrando tudo o que está relacionado com esse doente.

```c
void imprime_dados(pLista lista, int id){
    pLista aux = lista->prox; /* Salta o header */
    int existe=0;
    while (aux)
    {
        if(aux->doenteLista.id == id){
            printf("%d\n",aux->doenteLista.id);
            printf("%s \n", aux->doenteLista.nome);
            printf("%d/%d/%d \n",aux->doenteLista.data_nascimento.dia,aux->doenteLista.data_nascimento.mes,aux->doenteLista.data_nascimento.ano);
            printf("%d \n", aux->doenteLista.telemovel);
            printf("%s \n", aux->doenteLista.email);
            printf("\n");
            noListaregisto *registo_atual = aux->listaRegisto;
            existe=1;
            while (registo_atual)
            {
                printf("\nData de Registo: %d/%d/%d\n", registo_atual->registoLista.data_registo.dia, registo_atual->registoLista.data_registo.mes,
                printf("Tensão Máxima: %d\n", registo_atual->registoLista.tensao_max);
                printf("Tensão Mínima: %d\n", registo_atual->registoLista.tensao_min);
                printf("Peso: %d\n", registo_atual->registoLista.peso);
                printf("Altura: %d\n", registo_atual->registoLista.altura);
                registo_atual=registo_atual->prox;
            }
             break;
        }
       aux= aux->prox;
    }
    if(existe==0){
        printf("Doente com ID %d não encontrado.\n", id);
    }
}
```

Não podemos deixar de mencionar outras funções que englobam o ler e escrever doentes ou registos , quer dos ficheiros quer do input do utilizador , que mantém e garantem a funcionalidade do programa e de uma forma global são os pilares da aplicação , e também algumas funções que servem como proteção/defesa do programa quer de inputs quer de erros no ficheiro , principalmente a função verifica_data que mantém todos os doentes com uma informação o mais fidedigna possível e quer verifica_telemovel/cc/email_ficheiro que garantem a veracidade e a não repetição de dados.

André Lourenço Albuquerque nº2022231505
Rodrigo Morenito Borges nº2022244993