MANUAL DO PROGRAMADOR

Projecto Princípios de Programação Procedimental – 2014/2015

Igor Emanuel da Ponte Rodrigues - 2014212177

Joel Filipe Rogão Pires - 2014195242

Luis Miguel Rodrigues de Almeida Ramos - 2008112685

Ficheiros:

- main.h
- main.c

main.h:

Ficheiro header, posteriormente incluído no ficheiro main.c.

Contém todos os includes, defines (variáveis globais), estruturas utilizadas, assim como o protótipo de todas as funções utilizadas ao longo de main.c.

Neste programa são utilizadas 2 estruturas:

user:

Char * name -> guarda o nome de um paciente

Int id -> guarda o valor de identidade de um paciente

Patient:

user person -> Utiliza, de uma estrutura user, os valores de name e id.

time_t timestamp -> Aqui fica guardada a data, com precisão até aos segundos, de quando o paciente deu entrada no sistema.

int priority -> Um valor (entre 0 e 4) que determina qual a prioridade do paciente.

patient next -> Ponteiro para o próximo paciente (por ordem decrescente)

patient prev -> Ponteiro para o próximo paciente (por ordem crescente.)

Variáveis globais:

patient absolutefirst -> Guarda qual é o paciente que entrou mais recentemente no sistema.

patient plisth -> Guarda os pacientes que já tiveram entrada no sistema, mas que ainda não passaram a triagem.

patient redlh -> Guarda os pacientes que tiveram entrada no sistema, e lhes foi atribuída prioridade vermelha.

patient yellowlh -> Guarda os pacientes que tiveram entrada no sistema, e lhes foi atribuída prioridade amarela.

patient greenlh -> Guarda os pacientes que tiveram entrada no sistema, e lhes foi atribuída prioridade verde.

int autosave -> Variável que guarda o valor de autosave (0 para autosave off, 1 para autosave on).

int gid[] -> Array que armazena todos os valores de id.

int tid -> Variável que guarda o valor da ultima posição utilizada em gid[].

int red -> Número de pacientes de prioridade vermelha que precisam de ser atendidos.

int yellow -> Número de pacientes de prioridade amarela que precisam de ser atendidos.

main.c:

Contém todo o código referente às funções utilizadas no resto do programa.

Funções:

void init() -> Função que inicializa o programa. Cria listas, ficheiros, e inicializa o autosave a 1 (ligado).

void term() -> Finaliza o programa. Guarda toda a informação das listas no ficheiro, e posteriormente elimina-as da memória.

int mainMenu() -> Função que interage com o utilizador. Pede um input de 1 a 9, que ficará guardado na variável choice.

void redirecttofunction(int choice) -> O inteiro choice tem guardado a escolha do utilizador. A função vai, em função do argumento choice, redireccionar o utilizador para a função pretendida.

void admitPatient() -> Função onde o utilizador insere um nome e um id valido, para adicionar um novo paciente ao sistema.

int verifyid(char * id, ssize_t read) -> char * id contém um valor de id de um paciente, ssize_t read contem o tamanho de uma string. Esta função verifica se o id é valido ou não. Se for, devolve 1, se não, devolve 0.

int validateid(int id) -> int id contém um id. Função que valida o id inserido, caso já exista retorna 0, caso contrário retorna 1.

void removeid(int id) -> int id contém um id. Esta função procura o id dado como parâmetro, e remove-o do array gid[].

void screening(patient list) -> patient list contém uma lista de pacientes. Esta função vai escolher o paciente indicado pelo utilizador, e atribuir-lhe uma prioridade.

int givepriority(int print) -> int print contém um valor 0 ou 1. Se o valor de print for 1, a função apresentará certas escolhas ao utilizador. Se for 0, as escolhas apresentadas serão diferentes.

Esta função devolve o valor inserido pelo utilizador, que pode variar entre 1 e 5, (se print for 1) ou entre 1 e 3(caso print seja 0).

int verifypriority(int print, char * priority, ssize_t read) -> int print contém um valor. char * priority vai conter uma prioridade. ssize_t read vai conter um tamanho de uma string.

A função vai verificar o valor inserido. Se print for 1, vai verificar se pertence ao intervalo [1,5]. Se for 0, vai verificar se pertence a [1,3].

Retorna 1 ou 0, caso tenha sucesso ou não, respectivamente, ao validar a priority.

void deleteabscreening(int ab, patient list) -> int ab contém um valor. patient list contém uma lista de pacientes.

Remove um paciente de uma lista qualquer.

void chooseprint() -> Consoante a escolha do utilizador, imprime uma das listas.

void patientrules() -> Dita as principais regras para o atendimento de pacientes.

void determinenextpatient(int idl) -> int idl contém um valor. A função vai, segundo as regras definidas, determinar qual é o próximo paciente a ser atendido.

patient findnext(patient list) -> patient list contem uma lista. Esta função devolve o primeiro paciente que encontrar na lista list.

patient createlist() -> Função que inicializa uma lista. Começa pelo header e devolve-o.

void destroylist() -> Destrói uma lista, removendo-a da memória.

int isheader(patient list) -> patient list contém um paciente. A função verifica se list é o header ou não. Se for, devolve 1. Se não, devolve 0.

void insertandsort(char * name, int id, time_t timestamp, int priority, int attended, patient list) -> char * name é um argumento que contém um nome. int id contém o valor de id. time_t timestamp tem um valor time_t. int priotiy contém o valor de priority do paciente. int atender contém o valor de attended do paciente. patient list contém uma lista de pacientes.

A função insere e ordena em list o paciente com as propriedades name, id, timestamp, priority e attended.

void insertprev(patient new) -> patient new contém um paciente. Ordena o paciente new consoante a sua timestamp.

patient findinlist(char * name, int id, patient list) -> char * name contém um nome de um paciente. int id um id. patient list uma lista de pacientes. A função vai procurar o paciente com as características name e id na lista list. Se o encontrar, devolve-o. Se não, devolve NULL.

time_t gettimestamp(char * name, int id, patient list) -> char * name contém um nome. int id contém um id. patient list contém uma lista de pacientes.

Procura em list o paciente com as propriedades name e id, e devolve a sua timestamp ou 0, caso tenha encontrado a paciente ou nao respectivamente.

int changelist(char * name, int id, time_t timestamp, int priority, int attended, patient list, patient other) -> char * name contém um nome. int id contém o valor de id do paciente. time_t timestamp contém um valor time_t.

int prority guarda o valor da prioridade do paciente.

patient list contém uma lista de pacientes (onde o paciente se encontra). Patient other contém a lista de pacientes para onde o queremos transferir.

A função troca o paciente que pertence à lista list que tenha as propriedades name, id, timestamp, priority, attended para a lista other.

Retorna 1 ou 0 caso tenha sucesso ou não, respectivamente.

int deletefromlist(char * name, int id, patient list) -> char *name contém um nome. int id contém um id. patient list contém uma lista de pacientes.

A função que elimina o paciente com as propriedades name e id da lista list.

Retorna 1 ou 0 caso tenha sucesso ou não, respectivamente.

void printftol(patient list) -> patient list contém uma lista de pacientes. Imprime os pacientes pertencentes à lista list, por ordem decrescente de entrada no sistema.

void printltof() -> Imprime todos os pacientes, por ordem crescente de entrada no sistema.

int createfile(char * filename) -> char * filename contém o nome que queremos para o ficheiro.

Função que cria um ficheiro.

Devolve 1 se o ficheiro existir, e devolve 0 se não existir.

void deletefile(char * filename) -> char * filename contém o nome do ficheiro. Elimina o ficheiro com o nome filename.

void import(char * filename) : char * filename contém o nome do ficheiro. Esta função importa a informação existente no ficheiro para as listas do programa.

time_t convertto(char * timestring) -> char *timestring contém uma string. A função vai converter a string timestring para time_t, e devolve esse mesmo time_t.

char * convertfrom(time_t timestamp) -> time_t timestamp contém um valor time_t. A função converte time_t timestamp para uma string, e devolve a mesma.

void export(char * filename) -> char * filename contém o nome do ficheiro. Exporta para o ficheiro filename a informação das listas.

void savefiles() -> Guarda todas as listas do programa no ficheiro.

void signalhandler() -> Função que faz handling de um sinal.