Universidade de Coimbra
Faculdade de Ciências e Tecnologias
Departamento de Engenharia Informática

Gestão de Espaços de uma Instituição

Manual do Programador

Emanuel dos Santos Matos – 2011149813
Rui Miguel Freitas Rocha – 2010135453

Coimbra, 7 de Junho de 2013

Índice

Introdução

Com este projecto pretende-se desenvolver uma aplicação que ajuda na gestão de espaços de uma instituição. Esta aplicação deve armazenar na memória informações básicas sobre os dados dos clientes e dados das salas.

Para alem disso, deve ainda permitir ao utilizador uma forma interactiva de fazer reservas, pré-reservas, cancelar reservas e pré-reservas, listagens por utilizador ordenadas por data e listar por salas ordena por datas no sentido crescente.

Esta aplicação pode revelar-se bastante útil, pois permite ao utilizador gerir os seus espaços de uma forma fácil e eficiente.

Para além disso, esta aplicação, permite ao utilizador salvar as alterações feitas durante a execução, dados do cliente, salas, reservas e pré-reservas em ficheiros de texto de modo a que as mesmas estejam disponíveis quando a aplicação voltar a ser iniciada.

Para a realização desta aplicação necessitamos das seguintes competências:

- Escrita de programas em C. Domínio do ambiente de desenvolvimento
- Escrita de código correctamente formatado e identado
- Acesso a ficheiros
- Definição de novos tipos de dados
- Domínio de estruturas de dados dinâmicas
- Utilização de filas de espera
- Utilização de algoritmos de ordenamento

Este manual tem por objectivo facilitar a compreensão do funcionamento da aplicação, explicando ao leitor não só o raciocínio sobre o qual esta aplicação foi idealizada, como também o funcionamento e a utilização de algumas funções, algoritmos, variáveis, etc, usadas no desenvolvimento da mesma.

No final da leitura deste manual esperamos que o leitor se sinta elucidado sobre todo o funcionamento e raciocínio associados a esta aplicação.

Estruturas (estruturas_dados.h)

No ficheiro estruturas_dados.h encontram-se as estruturas definidas estruturas de dados utilizadas nesta aplicação para armazenar em memorias os dados referente as salas, utilizadores e reservas, como se pode observar de seguida:

```
typedef struct Reserva *Lreservas;
typedef struct Pessoa *Lutilizadores;
```

Dois ponteiros, um aponta para a lista de reservas (*Lreservas) e outro aponta para a lista de utilizadores (*Lutilizadores).

```
typedef struct Sala{
   Lreservas res;
   Lreservas pre_res;
}Sala;
```

Estrutura Sala. Esta estrutura vai armazenar os dados referentes as reservas e pré-reservas de uma sala.

```
typedef struct Data{
   int dia;
   int mes;
   int ano;
   int hora;
   int minutos;
}Data;

typedef struct Pessoa{
   int NIF;
   int n_telemovel;
   char nome[tamanho];
   Lutilizadores next;
}Pessoa;

typedef struct Reserva{
   int NIF;
   int sala;
   Data data;
   Lreservas next;
}Reserva;
```

Referente a estrutura Pessoa recebe os dados pessoais do utilizador, NIF, telemóvel e nome do cliente e um ponteiro que aponta para o elemento seguinte da lista ligada.

No que diz respeito a estrutura Reserva, esta armazena o NIF do cliente, o numero da sala, uma variável tipo para armazenar a data e a hora em que se pretende fazer a reserva e contem um ponteiro para o próximo elemento da lista.

```
FILE *fp;
```

Utilizamos uma variável global do tipo file para facilitar o acesso aos ficheiros de armazenamento de dados.

```
#define MAXsalas 30
#define tamanho 50
#define freservas "reservas.txt"
#define fpre_reservas "pre-reservas.txt"
#define futil "clientes.txt"
```

Definimos 3 constantes para facilitar o acesso a ficheiros e a utilização dos mesmos. Também definimos 2 constantes para delimitar o tamanho máximo de salas e do tamanho do nome do utilizador.

Ficheiro Principal do Projecto (main.c)

Ao iniciamos a aplicação main criamos de imediato duas estruturas de dados, uma para salas e outra para utilizadores. De seguida é feito o acesso aos ficheiros com a função carregar_dados que recebe um vector de salas e a lista de utilizadores para carregar dos ficheiros de texto clientes, reservas e pré-reservas.

Após o acesso aos ficheiros o ecrã é limpo mostrando de seguida o menu principal no qual o utilizador escolhe as varias opções tais como fazer reservas, cancelar, listagens, guardar e sair.

É permitida ainda uma consulta pelo NIF do utilizador ou pelo número das salas mostrando as reservadas e pré-reservadas já feitas ordenadas por data.

Sempre que o utilizador escolhe uma opção o ecrã e limpo e mostra novamente ao utilizador o menu para que o utilizador possa escolher a próxima tarefa a realizar, tornado desta forma mais fácil a orientação no menu. Sempre que uma opção do menu não é valida o ecrã é limpo mostrando novamente o menu.

Ao encerar o programa destrói todas as estruturas criadas ao iniciar, libertando assim a memória alocada para a sua execução.

Etapas de execução: Criação e destruição de listas

- Lreservas cria_lista_res(void);
- Lutilizadores cria_lista_util(void);

A primeira função cria uma lista do tipo Lreservas em que é iniciada a lista ligada para as reservas e para as pré-reservas, uma vez que a estrutura de ambas é a mesma.

No que diz respeito a segunda função, cria uma lista de utilizadores para armazenar os dados pessoais do cliente.

- Lreservas destroi_lista_reservas(Lreservas lista);
- Lutilizadores destroi_lista_util(Lutilizadores lista);
- void destruir_tudo(Sala *salas, Lutilizadores util);

Na primeira função vamos eliminar toda a lista ligada de reservas e também de pré reservas sendo desta forma libertada toda a memoria alocada anteriormente.

Na segunda função é destruída a lista de utilizadores e a memoria alocada é libertada.

Quando a função destrui_tudo, elimina todos os dados alocados no final do programa para não ficar a ocupar memoria evitadamente.

Etapas de execução: Acesso a ficheiros

Nesta aplicação podemos separa o acesso a ficheiros em duas situações distintas, carregar dados para memória e guardar dados alterados no disco.

Estas funções encontram-se em ficheiros.c.

As principais funções utilizadas para carregar os dados para memória são:

- void carregar_dados(Sala *salas, Lutilizadores util);
- void ler_nomes(Lutilizadores util);
- void ler_reservas(Sala *salas, char *ficheiro);

A função carregar_dados carrega os dados para as listas das reservas, das pre-reservas e utilizadores com a ajuda das funções ler_reservas e ler_nomes.

- void guardar_dados(Sala *salas, Lutilizadores util);
- int ficheiro_vazio(char *fich);

A função guardar_dados vai colocar todos os dados existentes nas listas ligadas para três ficheiros de texto anteriormente referidos. Sempre que chamamos esta função, o conteúdo dos ficheiros é apagado sendo depois novamente inserido com as alterações realizadas pelo utilizador.

A segunda função é uma função auxiliar para verificar se o ficheiro esta vazio ou não.

Etapas de execução: Manipulação de dados

Ao longo da execução temos a necessidade de criar novas estruturas como novas reservas, pré-reservas e utilizadores, mas também temos a necessidade de remover reservas e pré-reservas, ordenar e actualizar as reservas por data.

Para adicionar uma reserva a lista de reservas utilizamos a função:

- void fazer_reserva(Sala *salas, Lutilizadores util);
- int pedir_dados(Sala *salas, Lutilizadores util, Lreservas *aux);
- int ver_disponibilidade(Sala *salas, int sala, Data data);
- int verifica_reserva_data(Lreservas pre_reservas, Data data);
- int tem_pre_reservas(Lreservas pre_reservas, int nif, Data data);

Esta função (pedir_dados) recebe um vector de salas, uma lista de utilizadores e a lista de reservas e com o auxílio da função ver_disponibilidade verifica se já existe uma reserva para a data escolhida pelo utilizador, caso isto seja verdade, o utilizador escolhe se deseja fazer uma pré-reserva ou se pretende fazer uma reserva noutra sala.

Relativamente a função verifica_reserva_data vai verificar se pode pré-reservar na data escolhida pelo utilizador ou se a mesma já se encontra ocupada e a função tem_pre_reserva vai verificar se existem pré-reservas para a sala escolhida com ajuda da função anteriormente descrita e com a identificação do cliente, nif.

- int compara_data(Data data1, Data data2);
- int verifica_data(Data *dat);
- int bissexto(int ano);

Ao efectuar uma reserva utilizamos a função verifica_data para sabermos se a data inserida pelo utilizador é valida ou não e também utilizamos a função bissexto para controlarmos se o ano é bissexto ou não. Quanto a função compara_data, esta é utilizada para comparar as datas de modo a sabermos qual é a data mais recente e a mais antiga.

- void preencher(Sala *salas, Lutilizadores cliente, Lreservas res, int nif, int flag);
- void insere_no(Lreservas lista, Lreservas no);

.

As funções insere_no é uma função auxiliar da preencher que serve para fazer a reserva ou pré-reserva na sala escolhida pelo utilizador recebendo uma flag se for zero fazemos uma reserva e caso seja um fazemos uma pré-reserva.

- void insere_nome(Lutilizadores lista, Lutilizadores no);

Esta função serve para inserir os dados do utilizador tais como o nif, telemóvel e nome, na lista de utilizadores.

- void cancelar_reserva(Sala *salas, Lutilizadores util);
- void cancelar_pre_reserva(Sala *salas, Lutilizadores util);
- Lreservas elimina_no(Lreservas pre_reservas, Data data);
- Lreservas procura_no(Lreservas res, Data data);

As funções cancelar_reserva e cancelar_pre_reserva são duas funções para o utilizador cancelar a reserva ou a pré-reserva anteriormente realizada utilizando a função elimina_no e também a procura_no para encontrar o registo pretendido e retira-lo da lista de reservas ou pré-reservas consoante o que o utilizador pretende realizar.

- void imprime_sala(Sala *salas, Lutilizadores cliente);
- void imprime_utilizador(Sala *salas, Lutilizadores cliente);
- void imprime_salas(Sala *salas,int f);
- void imprime_informacao(Lreservas reservas, Lutilizadores util);
- void imprime_no(Lreservas lista_res, Lutilizadores lista_util);
- Lutilizadores procura_utilizador(Lutilizadores lista,int nif);

Estas funções servem todas para imprimir e mostrar ao utilizador o que é pretendido, a função procura_utilizador tal como o nome indica vai procurar o utilizador a lista pelo seu número de identificação.

- int controla_digitos(int num);
- int verifica_nif(int nif, Lutilizadores lista, Lutilizadores util);
- void ordena(Lreservas reserva, int ord);
- int conta(Lreservas lista_res);

A função controla_digitos serve para sabermos se o utilizador inseriu o número correcto de números servindo para controlarmos os erros.

A função ordena serve para ordenarmos as listas de reserva e pré-reserva pela data e pela flag, se for 1 ordena por ordem crescente mas se for -1 ordena por ordem decrescente.

Quanto a função verifica_nif, esta vai procuro o utilizar pelo nif pretendido.

Relativamente a função conta, esta vai contar o número de nos na lista.

- void tirar_(char *s);
- void por_(char *s);

Estas duas funções servem para retirar um espaço em branco e para por um espaço em branco, respectivamente, e são utilizadas a quando da leitura e escrita nos ficheiros.