**Manual do Programador**

Sistema de Gestão de Reservas para uma Estação de Serviços

**Docente: Marco Vieira**

Nuno Filipe Cardia Ferreira 2013153319

Sofia Mendes Gonçalves 2011158780

# Descrição Geral

Pretende-se com este projeto, implementar, em linguagem C, um programa que permite a gestão de utilizadores, reservas e pré-reservas de uma estação de lavagem e manutenção de automóveis. O programa corre em Linux.

O programa está organizado da seguinte forma:

**MENU PRINCIPAL**

**1 - Menu do Cliente**

**2 - Menu de Reservas**

**3 - Menu de Pré-reservas**

**0 - Sair**

A nível da primeira opção, Menu do Cliente, é possível fazer a gestão de clientes através das seguintes opções:

**MENU DE CLIENTES**

**1 - Introduzir novo cliente**

**2 - Eliminar cliente**

**3 - Imprimir lista de clientes**

**0 - Voltar ao Menu Principal**

A opção "Introduzir novo cliente" permite introduzir os dados "Nome", "Morada"," Telefone" e" E-mail" e, sendo que o programa se encontra protegido, a inserção de dígitos em "Nome" não é possível, o "Telefone" deve conter obrigatoriamente 9 algarismos e "E-mail" tem que possuir o caracter especial que o caracteriza, o "@". Estes dados, à medida que vão sendo inseridos vão sendo guardados não só numa lista específica, como também num ficheiro criado especificamente para estes.

É também possível eliminar clientes através da inserção do número do cliente que lhe foi atribuído.

Relativamente ao Menu de Reservas, as opções possíveis são as seguintes:

**MENU DE RESERVAS**

**1 - Inserir nova reserva**

**2 - Cancelar reserva**

**3 - Imprimir lista de reservas**

**4 - Ordenar reservas por data**

**5 - Imprimir reserva de um cliente específico**

**0 - Voltar ao Menu Principal**

A opção "Inserir nova reserva" permite ao utilizador introduzir os dados relativos à reserva que se pretende, tais como se é "Lavagem" ou "Manutenção", o número do cliente que pretende o serviço, e a hora e a data para quais se pretende fazer a reserva. No caso de já existir uma reserva para essa hora, isto é, se houver sobreposição de horários de serviços, aparecerá uma mensagem a avisar que a data e hora já está ocupada e a perguntar se o utilizador pretende manter os dados que inseriu, o que vai criar uma pré-reserva para essa mesma data e hora. No caso de o utilizador escolher a opção de alterar os dados da reserva, são requisitados novamente os dados da reserva.

A opção de "Cancelar Reserva" permite ao utilizador eliminar uma reserva previamente criada, requisitando a data e hora da reserva a eliminar. Nesta situação, no caso de haver uma pré-reserva para esta hora, essa pré-reserva vai passar a ser uma reserva e é cancelada.

"Imprimir lista de reservas" e "Ordenar reservas por data" seguem a mesma ordem de ideias que no caso do Menu de Clientes.

"Imprimir reserva de um cliente específico" imprime uma mensagem a requerer o numero do cliente pretendido e imprime no ecrã as reservas desse mesmo cliente.

Por fim, relativamente ao Menu de Pré-Reservas, as opções disponíveis são:

**MENU DE PRÉ-RESERVAS**

**1 - Cancelar pré-reserva**

**2 - Imprimir lista de pré-reservas**

**3 - Ordenar pré-reservas por data**

**0 - Voltar ao menu principal**

Todas as opções do menu de pré-reservas seguem a mesma ordem de ideias do menu de reservas.

# Funções

## Funções de Verificação de Inputs:

### verifica\_string(char \*s): permite verificar se o input é uma cadeia de caracteres e devolve 1 se tal acontecer.

### verifica\_numero(char \*numero): permite verificar se o input é um conjunto de algarismos e devolve 1 se tal acontecer.

### hour\_check(char \*horas): valida a hora inserida como input. Permite apenas a inserção de um horário entre as 8:00 e as 18:00, de 30 em 30 minutos.

### verifica\_data(int dia,int mes,int ano): valida a data inserida como input, tendo como base o calendário gregoriano.

### check\_email(char \*email): valida o e-mail inserido como input e obriga a que o utilizador insira o caracter especial que o caracteriza, o "@".

### clean\_stdin(void): semelhante a fflush(stdin) em windows. Devido a esta função em algumas partes do programa necessitarão de um input de qualquer carater para o mesmo continuar.

### List\_c cria\_lista(void): cria uma lista de clientes.

### procura\_cliente(List\_c lista, int chave, List\_c \*anterior, List\_c \*atual): percorre a lista de clientes previamente criada, tendo como referência o número de cliente.

### procura\_cliente\_2 (List\_r lista, int chave, List\_r \*ant, List\_r \*actual): semelhante a procura\_cliente(...), com a particularidade de devolver o valor 1 se cliente com um determinado número de cliente associado existir.

### inserir\_cliente(List\_c lista, int num\_cliente):permite introduzir na lista de clientes os dados (Nome, Morada, Telefone, E-mail) relativos a uma determinada pessoa. Evoca a função procura\_cliente() de modo a integrar a informação recolhida na lista de clientes previamente criada.

### imprime\_lista\_cliente(List\_c lista): mostra o conteúdo da lista de clientes que deve conter as informações recebidas na função inserir\_cliente().

### elimina\_cliente(List\_c lista): após a identificação do número de cliente, permite eliminar da memória todos os dados associados a esse número.

### ordenar\_nome\_cliente(List\_c lista): coloca os clientes previamente inseridos na função inserir\_cliente() por ordem alfabética.

### ordenar\_numero\_cliente(List\_c lista): coloca os clientes previamente inseridos na função inserir\_cliente() por ordem crescente em relação ao número de cliente.

### get\_int\_len (int value): função que verifica o “tamanho” de um inteiro. Faz parte da função de proteção do input da data.

### \*subtrai\_horas\_lavagens(char horas[size]): recebe a hora introduzida como input e aplica uma subtração de 30 minutos a essa mesma hora de modo a verificar se esse horário se encontra ocupado.

### \*soma\_horas\_lavagens(char horas[size]): recebe a hora introduzida como input e aplica uma soma de 30 minutos a essa mesma hora de modo a verificar se esse horário se encontra ocupado.

### List\_r criar\_lista\_reservas(void): *cria uma lista de pré-reservas*.

### List\_r criar\_lista\_pre\_reservas(void): cria uma lista de reservas.

### procura\_reserva (List\_r lista, int chave, List\_r \*ant, List\_r \*actual): *percorre a lista de reservas previamente criada, tendo como referência o número de cliente.*

### procura\_reserva\_2 (List\_r lista, int chave, List\_r \*ant, List\_r \*actual): semelhante a procura\_reserva(...), com a particularidade de devolver o valor 1 se cliente com um determinado número de cliente associado existir.

### inserir\_pre\_reserva(List\_r lista,int num, int \*data, char \*horas, char \*tipo): evoca a função procura\_reserva() e insere os dados recebidos como input na função verificar\_reserva(). Adiciona esses dados à lista de pré-reservas.

### inserir\_reserva(List\_r lista,int num, int \*data, char \*horas, char \*tipo): evoca a função procura\_reserva() e insere os dados recebidos como input na função verificar\_reserva(). Adiciona esses dados à lista de reservas.

### verifica\_reserva(List\_r lista, int \*data, char \*horas): função que verifica se a data e a hora, dados como argumentos, podem ser incluidos na lista das reservas;

### verificar\_reserva(List\_r lista,List\_r lista\_1): função que verifica se os dados da reserva, que foram introduzidos por input, podem ser introduzida na lista das reservas, caso contrário vai para a lista das pré-reservas.

### imprime\_lista\_pre\_reservas(List\_r lista): mostra o conteúdo da lista de pré-reservas que deve conter as informações recebidas na função verificar\_reserva().

### verificar\_pre\_reserva(List\_r lista, List\_r lista1): função que verifica se a pré-reserva pode ser passada para as reservas.

### imprime\_lista\_reservas(List\_r lista): mostra o conteúdo da lista de reservas que deve conter as informações recebidas na função verificar\_reserva().

### elimina\_reserva(List\_r lista): recebe como inputs a data e a hora da reserva que se pretende eliminar e elimina a reserva associada.

### elimina\_pre\_reserva(List\_r lista): recebe como inputs a data e a hora da reserva que se pretende eliminar e elimina a reserva associada.

### imprime\_reserva\_cliente(List\_r lista): evoca a função procura\_reserva\_2() de modo a poder validar a existência de um determinado número de cliente. Recebe como input o número de cliente atribuído a uma determinada pessoa e imprime a reserva associada à mesma.

### ord\_data\_reserva(List\_r lista): ordena a lista introduzida como argumento por data, da mais antiga para a mais recente.

### ord\_reservas\_numero\_cliente(List\_r lista): ordena a lista introduzida como argumento por numero de cliente, por ordem crescente.

### elimina\_tudo(List\_r lista, List\_r lista1): elimina todas as reservas e pré reservas.

### escrever\_ficheiro\_clientes(List\_c lista\_clientes): guarda a informação introduzida na lista de clientes num ficheiro com o nome "dados\_clientes.txt".

### escrever\_ficheiro\_reservas(List\_r lista\_reservas): guarda a informação introduzida na lista de reservas num ficheiro com o nome "dados\_reservas.txt".

### escrever\_ficheiro\_pre\_reservas(List\_r lista\_reservas): guarda a informação introduzida na lista de reservas num ficheiro com o nome "dados\_pre\_reservas.txt".

### main(): contém os menus "Principal", "de Clientes", "de Reservas" e "de Pré-reservas" e evoca as funções necessárias para o funcionamento do programa.