Projeto Final P1

Eduardo Miguel Moreira Junqueira nº30241Sérgio Miguel Gonçalves nº30243Andrei Tall

11 de Junho de 2023

Conteúdo

1 Introdução

2 Enunciado

Objetivo: Este trabalho prático visa criar uma aplicação que permita fazer a gestão de um centro de mecânica rápida. Descrição: A aplicação deve permitir gerir funcionários, clientes e serviços do centro de mecânica rápida. Neste centro de mecânica são realizados diversos serviços: Mudanças de óleo, mudanças de pneus, alinhamento da direção, check-up ao veículo, etc. O centro de mecânica rápida tem espaço para 15 mecânicos. Sobre cada mecânico a aplicação deve armazenar: Nome, NIF, Morada, Contacto telefónico, Identificador na aplicação (sequencial) e data de nascimento. A aplicação deve permitir:

- Gerir tipos de serviços (criar, editar, remover e listar) e respetivos preços.
- Gerir mecânicos (criar, editar, remover).
- Listar mecânicos por ordem alfabética do nome.
- Listar mecânicos, ordenados por idade ascendente.
- Colocar um mecânico indisponível.
- Gerar um relatório para um ficheiro de texto com a informação de todos os mecânicos

O centro de mecânica rápida guarda as informações dos clientes que realizam serviços. Sempre que um cliente queira agendar um serviço para o seu veículo, este deve fornecer os seus dados: Nome, NIF, contacto telefónico e matrícula do veículo a ser intervencionado. A aplicação deverá permitir:

- Gerir clientes (criar, atualizar e remover).
- Listar clientes por ordem alfabética de nome
- Apresentar as informações, dado o seu NIF.

A aplicação deverá permitir gerir os serviços mecânicos com duração de uma hora. Cada serviço deverá armazenar a identificação do cliente, do mecânico, a data e hora do serviço, estado do serviço (1 – marcado, 2 – realizado, 3 – Não compareceu), relatório de intervenção e o preço a pagar. A aplicação deverá permitir:

- Agendar um serviço
- Listar todos os serviços de hoje e de um determinado dia ordenados por hora ascendente.
- Listar todos os serviços de hoje e de um determinado dia para um determinado mecânico.
- Listar todos os serviços de hoje e de um determinado dia por tipo de serviço.
- Saber o histórico de serviços de um determinado veículo pela matrícula.
- Simular a realização de um serviço: informar qual o mecânico e a data e hora do serviço
- Listar os serviços em que o cliente não compareceu para um determinado dia.
- Saber por cada tipo de serviço, quando foi faturado hoje e este mês.
- Saber quantos serviços realizou cada mecânico este mês.
- Gerar um relatório por dia e por mês de contas. O relatório deverá organizar a informação por tipo de serviço.

Caso o cliente feche a aplicação, ou esta feche inesperadamente, o cliente deve conseguir continuar a utilizar a aplicação sem qualquer perda de dados.

3 Projeto

3.1 projeto.c

Neste trecho de código, nós temos a função ‘MenuInicial()‘, que é a função principal do programa. Ela é responsável por exibir um menu para o usuário e tomar ações de acordo com a opção escolhida. Dentro dessa função, declaramos várias variáveis e arrays que serão utilizados ao longo do programa. Por exemplo, temos o array ‘mecanica‘ para armazenar informações sobre os mecânicos, o array ‘servicos‘ para armazenar informações sobre os serviços, e variáveis como ‘numservicos‘, ‘idmecanico‘, ‘idservico‘, entre outras, para controlar o número de serviços e identificadores. Em seguida, utilizamos um loop ‘do-while‘ para exibir o menu e solicitar a opção ao usuário. Essa estrutura garante que o menu seja exibido pelo menos uma vez e continua a exibição até que uma opção válida seja escolhida. Dentro do loop, utilizamos um switch-case para executar as ações correspondentes a cada opção escolhida. Por exemplo, se o usuário escolher a opção 1, chamamos a função ‘AdicionarServicos()‘ para adicionar serviços. Se o usuário escolher a opção 2, chamamos a função ‘AdicionarMecanicos()‘ para adicionar mecânicos. E assim por diante. Cada opção do menu tem um trecho de código correspondente, onde são chamadas as funções apropriadas para executar as tarefas desejadas. Além disso, há um caso padrão (default) para lidar com opções inválidas. No final, temos a função ‘main()‘, que é o ponto de entrada do programa. Nela, chamamos a função ‘MenuInicial()‘ para iniciar a execução do programa.

3.2 structs.h

Este trecho de código contém definições de estruturas utilizando a diretiva de pré-processador ifndef e define. Essas diretivas de pré-processador são usadas para evitar a inclusão repetida do arquivo de cabeçalho "structs.h". Elas verificam se o identificador "STRUCTS$_H$" $já foi definido anteriormente. Senão ativ$ $processador ignora o conteúdo. Aqui, definimos uma estrutura chamada Cliente, que contém campos para armaz$

3.3 Clear.h

Neste trecho de código, temos a definição de duas funções relacionadas à limpeza do buffer e do terminal:

clearBuffer(): Esta função é responsável por limpar o buffer de entrada. Ela utiliza um laço de repetição while para ler os caracteres do buffer até encontrar um caractere de nova linha (") ou o final do arquivo (EOF). Isso é feito chamando a função getchar(). Dessa forma, todos os caracteres indesejados no buffer de entrada são descartados. ClearTerminal(): Esta função é responsável por limpar o terminal. Ela utiliza diretivas de pré-processador para verificar se o sistema operacional é o Windows ($_W IN32$). $Se for, a função system("cls") é chamada, o que limpa o terminal no Windows. Caso contrário, a função$

3.4 funcoes.h

altera as cenas em cima direito Esse trecho de código é um exemplo de um arquivo de cabeçalho em C. Nele, são declarados diversos protótipos de funções que serão implementadas em outros arquivos do projeto. Vou explicar o código para você. O trecho começa com as diretivas de pré-processador ifndef FUNCOES$_H$ $e define FUNCOES_H. Essas diretivas são utilizadas para evitar múltiplas inclusões do mes$

3.5 Servicos.c

Esse código C é um arquivo de cabeçalho chamado "Serviços.h"que contém várias funções relacionadas à manipulação de serviços. Vou explicar cada função em detalhes:

AdicionarServicos: Essa função permite adicionar um novo serviço à lista encadeada de serviços. Ela solicita ao usuário o nome, preço, ID e modelo da matrícula do serviço a ser adicionado. Em seguida, verifica se o ID do serviço já existe na lista antes de adicionar o novo serviço. EditarServicos: Essa função permite editar as informações de um serviço existente. O usuário precisa fornecer o ID do serviço que deseja editar. Em seguida, a função percorre a lista de serviços e procura pelo serviço com o ID fornecido. Se o serviço for encontrado, o usuário pode editar o nome, preço, ID e modelo da matrícula do serviço. RemoverServicos: Essa função permite remover um serviço da lista de serviços com base no ID fornecido pelo usuário. A função percorre a lista de serviços e procura pelo serviço com o ID fornecido. Se o serviço for encontrado, ele é removido da lista, e os outros serviços são reorganizados para preencher o espaço vago. ListarServicos: Essa função exibe na tela a lista de serviços existentes. Ela percorre a lista de serviços e imprime o nome, preço, ID e modelo da matrícula de cada serviço. Essas funções são definidas no arquivo "Serviços.h"e são usadas em outros arquivos do seu projeto para manipular os serviços. O arquivo de cabeçalho inclui outros arquivos de cabeçalho, como "structs.h", "funcoes.h"e "Clear.h", para obter as definições necessárias para os tipos de dados e funções utilizadas.

4 Centro$_S ervicos.h$

Inicialmente, são incluídos os cabeçalhos das bibliotecas necessárias para o funcionamento do programa. A função agendarServico() é definida. Essa função representa uma opção do menu (no caso, a opção 4) que permite ao usuário agendar um serviço no centro mecânico. Dentro da função agendarServico(), é criada uma nova instância da estrutura aplicacoes, que provavelmente contém informações sobre o serviço a ser agendado. Não é fornecida a definição completa dessa estrutura no trecho de código fornecido. O programa solicita ao usuário que digite o ID do cliente, o ID do mecânico, a data e hora do serviço, o estado do serviço (marcado, realizado ou não compareceu),

o relatório de intervenção e o preço a pagar. As informações fornecidas pelo usuário são lidas e armazenadas nas variáveis correspondentes da estrutura aplicacoes.

5 (

5.1 Funções Implementadas

O código implementa as seguintes funções:

5.2 AdicionarMecanicos

A função "AdicionarMecanicos"é responsável por adicionar um novo mecânico à lista de mecânicos. Ela solicita ao usuário informações como nome, NIF, número de telefone, morada, data de nascimento e ID do mecânico. Em seguida, cria um novo nó "gestao"com essas informações e o adiciona à lista.

5.3 EditarMecanicos

A função "EditarMecanicos"permite ao usuário editar as informações de um mecânico existente na lista. Ela solicita o ID do mecânico que se deseja editar e percorre a lista até encontrar o mecânico correspondente. Em seguida, permite ao usuário editar as informações desse mecânico.

5.4 OrdenarMecanicosPorIdade

A função "OrdenarMecanicosPorIdade"é responsável por ordenar a lista de mecânicos com base na data de nascimento em ordem ascendente. No entanto, a implementação dessa função está incompleta, pois a função "TrocarMecanicos"está comentada. É necessário revisar e implementar essa função para obter a ordenação correta.

5.5 ListarMecanicos

A função "ListarMecanicos"exibe na tela as informações de todos os mecânicos presentes na lista. Ela percorre a lista e exibe os dados de cada mecânico. Além disso, oferece a opção de ordenar os mecânicos por idade em ordem ascendente.

5.6 Mecanicodisponibilidade

A função "Mecanicodisponibilidade"verifica e registra a disponibilidade de um mecânico. Ela solicita ao usuário o ID do mecânico e pergunta se ele está disponível (resposta "S"ou "N"). Em seguida, atualiza o campo de disponibilidade do mecânico correspondente.

5.7 RemoverMecanicos

A função "RemoverMecanicos"permite ao usuário remover um mecânico da lista com base no seu ID. Ela solicita o ID do mecânico que se deseja remover e percorre a lista até encontrar o mecânico correspondente. Em seguida, remove o mecânico da lista.

5.8 gravarMecanicosArquivo

A função "gravarMecanicosArquivo"permite gravar as informações dos mecânicos em um arquivo. Ela recebe o nome do arquivo como parâmetro e cria um novo arquivo no modo de escrita. Em seguida, percorre a lista de mecânicos e escreve as informações de cada mecânico no arquivo.

5.9 PrintGestaoData

A função "PrintGestaoData"exibe na tela as informações dos mecânicos presentes na lista. Ela percorre a lista e exibe os dados de cada mecânico, incluindo o número, ID, nome, NIF, morada, telefone, data de nascimento e disponibilidade.

6 (

Centro$_M ecanicos.h$)

6.1 Função menuOpcoesMecanico

A função "menuOpcoesMecanico"é responsável por exibir um menu de opções relacionadas aos mecânicos em um centro mecânico. Ela utiliza um loop "do-while"para continuar exibindo o menu até que o usuário escolha a opção de voltar para o menu inicial.

As opções disponíveis no menu são as seguintes:

6.2 Opção 1: Remover Mecânico

Essa opção permite ao usuário remover um mecânico do centro mecânico. Ela solicita ao usuário o ID do mecânico que se deseja remover e chama a função "RemoverMecanicos"para realizar a remoção.

6.3 Opção 2: Alterar Mecânico

Essa opção permite ao usuário alterar as informações de um mecânico do centro mecânico. Ela solicita ao usuário o ID do mecânico que se deseja alterar e chama a função "EditarMecanicos"para realizar a alteração.

6.4 Opção 3: Listar Mecânico

Essa opção permite ao usuário listar os mecânicos do centro mecânico. No código fornecido, essa opção está apenas exibindo uma mensagem de sucesso, mas é possível chamar a função "ListarMecanicos"para listar efetivamente os mecânicos.

6.5 Opção 4: Voltar para o Menu Inicial

Essa opção permite ao usuário voltar para o menu inicial do centro mecânico. Ela exibe uma mensagem indicando que está voltando para o menu inicial.

7 clientes.c

adicionarCliente: Essa função permite adicionar um novo cliente à lista de clientes. Ela verifica se o limite máximo de clientes foi atingido e, em seguida, solicita ao usuário o nome, NIF, contato e matrícula do veículo do novo cliente. Os dados do cliente são armazenados em um array de estruturas chamado totalClientes. listarClientesPorNome: Essa função lista os clientes por ordem alfabética de nome. Ela usa o algoritmo de ordenação chamado selection sort para reorganizar o array de clientes totalClientes com base no nome dos clientes. Em seguida, imprime na tela o nome, NIF, contato e matrícula de cada cliente. exibirClientePorNIF: Essa função permite exibir as informações de um cliente com base no NIF fornecido pelo usuário. Ela percorre o array de clientes totalClientes e procura por um cliente com o NIF fornecido. Se o cliente for encontrado, exibe o nome, NIF, contato e matrícula do veículo. Caso contrário, exibe uma mensagem informando que o cliente não foi encontrado. removerCliente: Essa função permite remover um cliente com base no NIF fornecido pelo usuário. Ela percorre o array de clientes totalClientes e procura por um

cliente com o NIF fornecido. Se o cliente for encontrado, remove-o do array e ajusta a contagem de clientes (numClientes). Em seguida, exibe uma mensagem informando que o cliente foi removido. Caso contrário, exibe uma mensagem informando que o cliente não foi encontrado. Essas funções operam em um array chamado totalClientes, que pode armazenar até 100 clientes. O número atual de clientes é mantido na variável numClientes. Os dados dos clientes são armazenados em uma estrutura chamada Cliente, que contém campos para nome, NIF, contato e matrícula.

É importante mencionar que algumas definições, como a estrutura Cliente, não estão incluídas no código fornecido, pois são provenientes de um arquivo de cabeçalho chamado "structs.h". Portanto, é necessário verificar o conteúdo desse arquivo para entender completamente a estrutura Cliente e outras definições relevantes.

8 Conclusão

Em resumo, o código apresentado é uma parte de um programa em linguagem C que trata do agendamento de serviços em um centro mecânico. O trecho de código em questão é responsável por definir uma função chamada agendarServico(), que permite ao usuário inserir informações sobre um serviço a ser agendado, como ID do cliente, ID do mecânico, data e hora do serviço, estado do serviço, relatório de intervenção e preço a pagar.

Embora a definição completa da estrutura aplicacoes não seja fornecida no código fornecido, pode-se inferir que ela é usada para armazenar e manipular os dados relacionados aos serviços agendados.

É importante destacar que o trecho de código fornecido não está completo, e, portanto, não é possível analisar seu funcionamento global ou entender como essas informações são utilizadas posteriormente no programa.

No entanto, com base no que foi apresentado, podemos concluir que o código tem como objetivo fornecer uma funcionalidade para agendar serviços em um centro mecânico, permitindo o registro de informações importantes relacionadas aos serviços agendados.