Instituto Superior de Engenharia

Politécnico de Coimbra

Programação

2022 / 2023

Trabalho Prático – Metro Mondego

Bruno Tiago Ferreira Martins - 2022147149

11 de junho de 2023

Índice

Introdução

Descrição genérica da organização do programa:

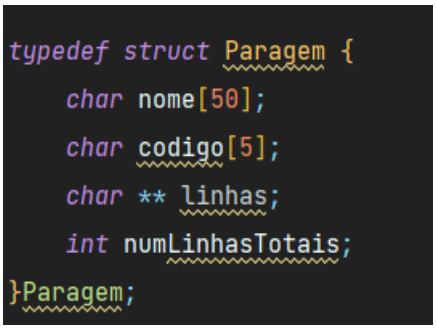
Este programa foi desenvolvido na linguagem C, respeitando no âmbito da disciplina de programação e permite efetuar a gestão do sistema de mobilidade do Metro Mondego. O ambiente escolhido foi o Clion(2023.1.2), pois tem mais auxílios e torna a escrita mais fácil como a execução do programa mais limpa.

Ele permite que os utilizadores realizem várias coisas, como adicionar paragens, adicionar linhas, adicionar e remover paragens de linhas e guardar os dados em arquivos de texto ou binários.

O programa utiliza um menu, onde o utilizador pode escolher as opções desejadas. O menu é estruturado em diferentes funções para cada tipo de opções. Essas funções são chamadas a partir da função principal (main) com base nas escolhas do utilizador.

O programa possui diferentes estruturas de dados para armazenar as informações das linhas e paragens. As linhas são representadas pela estrutura "Linha", que armazena o nome da linha e um array de paragens. Cada paragem é representada pela estrutura "Paragem", que possui o nome da paragem e um código alfanumérico.

Estruturas dinâmicas implementadas

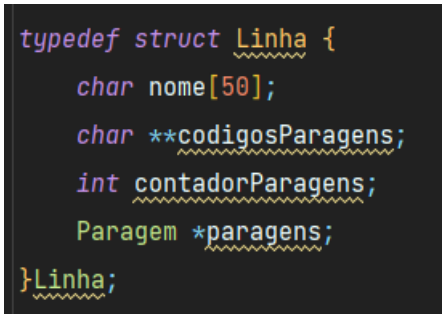


```
typedef struct Paragem {
    char nome[50];
    char codigo[5];
    char ** linhas;
    int numLinhasTotais;
}Paragem;
```

Figura 1 - Estrutura (Paragem)

Esta estrutura e responsável pelas paragens no sistema do metro.

Ela possui várias strings, uma para o nome, outra para o código alfanumérico e outra para armazenar os nomes das linhas em que a paragem está incluída. A utilização de ponteiros permite uma alocação dinâmica de memória para armazenar os nomes das linhas. Isso significa que a quantidade de linhas pode variar, e a estrutura pode armazenar paragens com diferentes números de linhas sem precisar de um espaço fixo.



```
typedef struct Linha {
    char nome[50];
    char **codigosParagens;
    int contadorParagens;
    Paragem *paragens;
}Linha;
```

Figura 2 - Estrutura (Linha)

Esta estrutura é responsável pelas linhas. É composta por uma string que guarda o nome da linha, um ponteiro que aponta para uma string que possui o código da paragem (array dinâmico de strings) permitindo armazenar os códigos alfanuméricos das paragens que pertencem à linha. Também é composta por um contador(inteiro) que server para indicar o número total de paragens existentes na linha. Para alem disso esta presente um ponteiro (paragens) que aponta para um array de estruturas (Paragem) e tem como função armazenar as paragens que pertencentes à linha.

Organização de ficheiros

- linhas.c – Neste ficheiro foram implementadas as funções relativas às linhas
- linhas.h – Este é o header do ficheiro linhas em que foi implementada a estrutura da Figura 2, variáveis externas e declaradas as funções implementadas no ficheiro linhas.c

- paragens.c – Neste ficheiro foram implementadas as funções relativas às paragens
- paragens.h - Este é o header do ficheiro paragens em que foi implementada a estrutura da Figura 1 e declaradas as funções implementadas no ficheiro paragens.c

- main.c- Neste ficheiro foram implementadas funções de guardar dados e a função main a parte principal que faz o programa executar.
- main.h - Este é o header do ficheiro main em que foram declaradas as funções implementadas no ficheiro main.c
- gui.c- Neste ficheiro esta a interface do menu.

Funções do programa

paragens.c

```
void gerarCodigo(char * codigo) {
```

Esta função gera um código alfanumérico de 4 caracteres para a nova paragem. O código é armazenado no array de caracteres codigo, que deve ser passado para a função.

```
void obterCodigoParagem(char * codigo, const char * nomeParagem) {
```

Esta função procura uma paragem com o nome fornecido no array de paragens. Se a paragem for encontrada, seu código é copiado para o array de caracteres código, isto é utilizado para a parte em que se adiciona paragens a linhas ou nas paragens predefinidas a linha predefinida (pois pelo nome da paragem ela encontra o código e atribui-o).

```
void registarParagem() {
```

Esta função é responsável por registar uma nova paragem e funciona da seguinte forma (pede ao utilizador o nome da paragem, verifica se o nome já existe, gera um código único para a nova paragem, cria a paragem e adiciona-a ao array de paragens).

```
void eliminarParagem() {
```

Esta função é responsável por eliminar uma paragem, é pedido ao utilizador o nome da paragem e depois a função procura a paragem pelo nome e verifica se a paragem está associada a linhas e se estiver não pode ser eliminada (existe outras verificações como ver se a paragem existe). Caso não esteja em nenhuma linha, elimina a paragem, move todas as paragens para a esquerda e de seguida efetua um decremento no contador de paragens e realoca o array de paragens para libertar a memoria da paragem que foi eliminada.

```
void adicionarParagensPreDefinidas() {
```

Esta função e responsável por adicionar paragens predefinidas ao sistema.

```
void visualizarParagens() {
```

Esta função mostra todas as paragens registadas e mostra o nome, o código e as linhas associadas. Se não houver paragens registadas, mostra uma mensagem a indicar.

linhas.c

```
Linha *encontrarLinha(char *nome)
```

Esta função e responsável por encontrar uma linha pelo nome.

```
void addParagempLinha(char *nomeLinha, char *nomeParagem) {
```

Esta função é responsável por adicionar uma paragem a uma linha específica, dado os nomes da linha e da paragem.

```
void addLinhasPreDefinidas()
```

Esta função adiciona linhas predefinidas com as paragens que foram predefinidas também na função adicionarparagenspredefinidas.

```
void addLinha(char *nome)
```

Esta função e responsável por criar uma linha com o nome fornecido pelo utilizador.

```
void addParagem(Linha *linha, char *nomeParagem, char **codigosParagens)
```

Esta função adiciona uma nova Paragem a uma Linha existente.

```
void removeParagemDeLinha(char *nomeLinha, char *nomeParagem)
```

Esta função remove uma Paragem de uma linha existente que o utilizador pretender.

```
void atualizarNomeLinha() {
```

Esta função e capaz de trocar o nome de uma linha existente consoante a escolha do utilizador.

```
void removeLinha()
```

Esta função remove uma Linha do sistema.

```
void mostrarLinhas()
```

Esta função mostra todas as linhas existentes no sistema, bem como as paragens associadas a cada Linha.

main.c

```
void calcularPercursos()
```

```
void addLinhaDeFicheiro(const char *nomeFicheiro)
```

Estas funções não foram implementadas

```
void guardarDados()
```

Esta função é responsável por interagir com o utilizador para decidir o tipo de ficheiro em que quer guardar os dados (txt ou bin)

```
void guardarDadosTexto()
```

Esta função guarda os dados no formato .txt.

```
void guardarDadosBinario()
```

Esta função guarda os dados no formato .bin.

```
int main()
```

Para finalizar esta é a função principal que é responsável por articular tudo e fazer com que o programa funcione (funciona como o controlo para todas as opções que o utilizador pode escolher).