Relatório de Trabalho - 1ª Fase

José Rebelo (31516)
Professor Luís Ferreira

March 30, 2025

Contents

1 Introdução

Este trabalho foi realizado no âmbito da unidade curricular Estruturas de Dados Avançadas. Ele apresenta um sistema de gestão de antenas, desenvolvido para facilitar o controlo da sua distribuição. O sistema permite criar, remover e listar antenas de forma eficiente, fornecendo apenas informações sobre a sua posição, representada pelas coordenadas (x, y), e sua frequência de operação, que pode ser do tipo A ou O. Dessa forma, busca-se otimizar a organização e gestão das antenas, o que proporciona maior precisão e eficiência na administração delas.

2 Ferramentas Utilizadas

Durante a realização do projeto foram utilizadas diversas ferramentas que desempenharam um papel crucial em cada etapa do trabalho. Cada ferramenta tem um propósito diferente

2.1 Visual Studio

O Visual Studio é um ambiente de desenvolvimento integrado (IDE) desenvolvido pela Microsoft. Ele oferece um conjunto abrangente de ferramentas para criar aplicações para diversas plataformas, incluindo Windows, web, dispositivos móveis e jogos. Com suporte para várias linguagens de programação, como C, C++, Python e JavaScript. Recorri ao uso do Visual Studio porque é uma ferramenta no qual estou confortável a usar.

2.2 Overleaf

O Overleaf é uma plataforma online destinada à criação, edição e colaboração em documentos LaTeX. Permite que múltiplos utilizadores possam editar simultaneamente o mesmo documento, com as alterações sendo refletidas em tempo real. Além disso, preconiza a compilação automática e a integração com serviços como o GitHub, tornando a elaboração de documentos técnicos e académicos muito mais rápida e prática. Recorri ao Overleaf para a criação do relatório em LaTex.

2.3 Linguagem C

A linguagem C é uma linguagem de programação que oferece controlo direto sobre o hardware por meio de ponteiros, manipulação de memória e acesso a recursos do sistema. Como adição, a linguagem é estruturada e suporta modularização e organização do código, o que a torna adequada para o desenvolvimento de projetos de média e grande escala. Para a realização deste projeto foi solicitado o uso desta linguagem de programação.

2.4 Doxygen

O Doxygen é uma ferramenta de documentação de código aberto que gera documentação a partir do código-fonte. Ele suporta diversas linguagens de programação e produz documentação em diversos formatos, como HTML, LaTeX e PDF. O Doxygen é muito utilizado para documentar APIs e projetos de software. Eu optei pela utilização do Doxygen para fazer a documentação do projeto

2.5 GitHub

O GitHub é uma plataforma de hospedagem de código para controlo da versão e colaboração. Ele permite que desenvolvedores trabalhem juntos em projetos, rastreiem alterações e partilhem código. Recorri ao uso do GitHub para armazenar o meu projeto.

2.6 ChatGPT

ChatGPT é uma Inteligência Artificia com modelo de linguagem avançado desenvolvido pela OpenAI. Ele pode gerar texto, responder a perguntas e conversar sobre diversos tópicos. O ChatGPT é usado para criar chatbots, gerar conteúdo e muito mais. Recorri

ao ChatGPT como uma ferramenta de pesquisa para os conteúdos necessários face à realização do projeto.

3 Código

3.1 Data

Na pasta Data encontra-se o ficheiro com as estruturas de dados das antenas e das zonas com efeito nefasto.

3.1.1 Data/dados.h

```
/**
* @file dados.h
* @author José Rebelo
* @brief Ficheiro com as estruturas das antenas e do nefasto
*/

/**
 *@brief Estrutura de dados das antenas
 */
typedef struct Antena {
    int x; //posicao x
    int y; //posicao y
    char frequencia; //frequencia (A ou 0)
    struct Antena* prox; //apontador para a proxima antena
} Antena;

/**
* @brief Estrutura de dados do efeito nefasto
*/
typedef struct Nefasto{
    int x; //posicao x
    int y; //posicao y
    struct Nefasto* prox; // apontador para o proximo efeito
} Nefasto;
```

3.2 Functions

Na pasta Functions encontram-se os ficheiros de declaração das funções do projeto.

3.2.1 Functions/funcoes.h

```
/**
 * @file funcoes.h
 * @author José Rebelo
 * @brief Ficheiro com as definições das funções do menu
*/
#include "../Data/dados.h"


#pragma region Funções do menu
```

```
/**
 * @brief Função de criar novas antenas
 */
Antena* criarAntena(int x, int y, char frequencia);


/**
 * @brief Função de remover as antenas do ficheiro e da lista
 */
void removerAntena(Antena** lista, const char* caminhoFicheiro, int x, int y);


/**
 * @brief Função de carregar as antenas do ficheiro
 */
Antena* carregarAntenasDoFicheiro(const char* caminhoFicheiro);
#pragma endregion
```

3.2.2 Functions/funcoes.c

```
    /**
 * @file funcoes.c
 * @author José Rebelo
 * @brief Ficheiro com as funções do menu
*/

/**
 * @brief Inclusão das bibliotecas necessárias
 */
#include "funcoes.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>


/**
 * @brief Função de criar novas antenas
 * @param x posição x
 * @param y posição y
 * @param frequencia frequência (A ou 0)
 * @return Antena* apontador para a nova antena
 */
#pragma region Criar Antena
    Antena* criarAntena(int x, int y, char frequencia){

        Antena* novaAntena = (Antena*) malloc(sizeof(Antena)); // Aloca memória para a

        // Intrução dos dados
        novaAntena->x = x;
```

```
        novaAntena->y = y;
        novaAntena->frequencia = frequencia;
        novaAntena->prox = NULL;

        return novaAntena;
    }
#pragma endregion



/**
 * @brief Função de carregar as antenas do ficheiro
 * @param caminhoFicheiro Caminho do ficheiro
 * @param lista Lista ligada
 * @param y Coordenada y (linhas)
 * @param x Coordenada x (colunas)
 */
#pragma region Carregar Antenas do Ficheiro

    Antena* carregarAntenasDoFicheiro(const char* caminhoFicheiro) {

        /**
         * @brief Abrir o ficheiro em modo leitura ("r -> read")
         */
        FILE* ficheiro = fopen(caminhoFicheiro, "r");

        if (ficheiro == NULL) { // Se o ficheiro não existir

            printf("Erro ao abrir o ficheiro, ficheiro inexistente.\n");
            return NULL;
        }

        Antena* lista = NULL; // Lista vazia no inicio
        char linha[256]; // Tamanho maximo da linha, em caso de ser pretendido aumenta
        int y = 0; // Coordenada y de começo

        // Ler o ficheiro linha a linha
        while (fgets(linha, sizeof(linha), ficheiro)) { // fgets le cada linha do fich

            int x = 0; // Após mudar de linha x volta a 0, esta linha tem de ficar aqu

            // Percorrer cada caracter da linha
            while (linha[x] != '\0' && linha[x] != '\n') { // Enquanto não chegar ao f

                if (linha[x] != '.') { // Se não for um ponto

                    Antena* novaAntena = criarAntena(x, y, linha[x]); // Cria uma nova
```

```
                    if (novaAntena != NULL) {
                        novaAntena->prox = lista; // Adiciona no início da lista ligad
                        lista = novaAntena;
                    }
                }
                x++;
            }
            y++; // Incrementa a coordenada Y (próxima linha)
        }

        fclose(ficheiro);
        return lista;
    }
#pragma endregion



/**
 * @brief Função de remover as antenas da lista
 * @param x posição x da antena no ficheiro
 * @param y posição y da antena no ficheiro
 */
#pragma region Remover Antena
    void removerAntena(Antena** lista, const char* caminhoFicheiro, int x, int y) { //

        if (*lista == NULL) { // Se a lista estiver vazia, imprime uma mensagem de err
            printf("Lista vazia. Nenhuma antena para remover.\n");
            return;
        }

        Antena* atual = *lista; // Apontador onde percorre a lista indicando a antena
        Antena* anterior = NULL;

        // Percorre a lista para encontrar a antena com as coordenadas (x, y) introduz
        while (atual != NULL && (atual->x != x || atual->y != y)) {
            anterior = atual;
            atual = atual->prox;
        }

        // Se encontrou a antena, remove-a da lista ligada
        if (atual != NULL) {

            if (anterior == NULL) {
                *lista = atual->prox; // Remover do início da lista
            }

            else {
                anterior->prox = atual->prox; // Remover do meio ou fim
```

```
            }
            free(atual); // Liberta a memória da antena removida
        }
        else {
            printf("Antena não encontrada na lista ligada.\n");
            return;
        }
    }
#pragma endregion



/**
 * @brief Função de listar as antenas
 * @param lista lista ligada de antenas
 */
#pragma region Listar Antenas
```

3.3 Main

Na pasta Main encontram-se os ficheiros principais do código do projeto.

3.3.1 Main/main.c

```
/**
 * @file main.c
 * @author José Rebelo
 * @brief Ficheiro principal (inclui o menu)
*/

/**
 * @brief Inclusão das bibliotecas necessárias
 */
#include <stdio.h>
#include  "menu.h"

/**
 * @brief Função principal
 */
#pragma region menu
int main() {

    menu();
}
#pragma endregion
```

3.3.2 Main/menu.h

```
/**
```

```
 * @file menu.h
 * @author José Rebelo
 * @brief Ficheiro com a função do menu
*/

/**
 * @brief Inclusão das bibliotecas necessárias
 */
#include "../Functions/funcoes.h"

/**
 * @brief Definição da função de menu
 */
void menu ();
```

3.3.3 Main/menu.c

```
    /**
 * @file menu.c
 * @author José Rebelo
 * @brief Ficheiro com as funções do menu
*/

/**
 * @brief Inclusão das bibliotecas necessárias
 */
#pragma region Inclusão de Bibliotecas

#include "menu.h"
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <string.h>

#pragma endregion

Antena* lista = NULL; // Lista ligada vazia (é assim que começa)

void menu() {

    Antena* novaAntena = NULL; //Evita erros de lixo na memória
    char frequencia;
    int x, y;
    const char* caminhoFicheiro = "./Text/dados.txt"; // Caminho para o ficheiro dos c

    /**
     * @brief Criar antena
     * @details Esta função tem como objetivo criar uma nova antena à lista ligada
     * @param x posição x (colunas)
```

```
     * @param y posição y (linhas)
     * @param frequencia frequência (A ou O)
     */
#pragma region Criar Antena

    printf("Frequência (A/O): \n");
    scanf(" %c", &frequencia); // O espaço evita o código avançar a introdução da frec

    printf("X e Y: \n");
    scanf("%d %d", &x, &y); // Introdução das coordenadas (x, y)

    novaAntena = criarAntena(x, y, frequencia); // Cria nova antena

    if (novaAntena != NULL) {
        novaAntena->prox = lista; // Aponta para a próxima antena
        lista = novaAntena; // Atualiza a lista ligada com a nova antena
        printf("Antena criada com sucesso.\n"); // Só teste depois tira
    }

    else {
        printf("Erro ao criar a antena.\n");
    }

    listarAntenas(lista); // A função aplica-se na variável da lista ligada

#pragma endregion

    /**
     * @brief Guardar dados das antenas
     * @details Esta função tem como objetivo guardar os dados das antenas do ficheiro
     * @param caminhoFicheiro caminho do ficheiro (mantém-se sempre o mesmo)
     * @param lista lista ligada
     * @param temp variável temporária para guardar a lista ligada
     */
#pragma region Guardar Dados Das Antenas

    Antena* listaFicheiro = carregarAntenasDoFicheiro(caminhoFicheiro); // Carrega as

    // Agora junta as duas listas
    if (listaFicheiro != NULL) {

        Antena* temp = listaFicheiro; // Variável temporária para percorrer a lista ca
        while (temp->prox != NULL) {

            temp = temp->prox; // Vai percorrer a lista até ao fim
        }
        temp->prox = lista;  // Liga a lista carregada à lista existente
```

```
        lista = listaFicheiro; // Atualiza a lista principal
    }

    printf("Antenas adicionadas com sucesso.\n"); // Só teste depois tira

    listarAntenas(lista); // A função aplica-se na variável da lista ligada

#pragma endregion

    /**
     * @brief Remover antena
     * @details Esta função tem como objetivo remover uma antena da lista ligada
     * @param caminhoFicheiro caminho do ficheiro (mantém-se sempre o mesmo)
     * @param x posição x (colunas)
     * @param y posição y (linhas)
     */
#pragma region Remover Antena

    printf("X e Y: \n");
    scanf("%d %d", &x, &y); // Introdução das coordenadas (x, y)

    removerAntena(&lista, caminhoFicheiro, x, y);

    printf("Antena removida com sucesso.\n"); // Só teste depois tira

    listarAntenas(lista); // A função aplica-se na variável da lista ligada

#pragma endregion

    /**
     * @brief Listar antenas
     * @details Esta função tem como objetivo listar as antenas registadas na lista li
     * @param lista lista ligada
     */
#pragma region Listar Antenas

    listarAntenas(lista); // A função aplica-se na variável da lista ligada

#pragma endregion
}
```

4 Dificuldades

Durante o desenvolvimento deste projeto, a parte mais desafiadora foi a implementação das funções relacionadas ao efeito nefasto. A lógica necessária para detetar e representar este efeito exigiu um esforço significativo, especialmente na manipulação do ficheiro de dados. A complexidade do algoritmo e a necessidade de coordenar esta funcionalidade

com as outras partes do trabalho foram os principais obstáculos encontrados.

5 Webgrafia

References

[1] Repositório GitHub. https://github.com/JoseRebel0/trabalho-pratico-parte1

[2] Conversa com o ChatGPT. https://chatgpt.com/share/67dc8d82-a9e4-8011-b39c-af3d9cf3a206.

[3] Playlist de manipulação de ficheiros em C. https://youtube.com/playlist?list=PLa75BYTPDNKY6q
oD65MtoBbU3BBejTmKsi=pVJ9BSHtNk-$_w vxS$.