

ESTRUTURAS DE DADOS AVANÇADAS (EDA)

PROJETO DE AVALIAÇÃO - FASE I

Ivo Fernandes (8819)

Docente

Luís Gonzaga Ferreira

30 Março, 2025

ÍNDICE

Introdução	3
Fase I	4
Análise e Especificação	4
Requisitos Funcionais:	4
Requisitos Não Funcionais:	4
Arquitetura Lógica / Funcional	4
Estrutura dos Dados	5
Definição da Estrutura Antena	5
Implementação	5
Assinatura das Funções	5
Carregamento de Dados	6
Funções implementadas	6
Modularização	7
Fase II	7
Análise e Especificação	8
Requisitos Funcionais:	8
Requisitos Não Funcionais:	8
Arquitetura Lógica / Funcional	8
Estrutura dos Dados	9
Funções implementadas	9
Modularização	10
Funções implementadas	10
Implementação da função adicionarLigacao:	11
Conclusão	11
Dificuldades Encontradas	12

Repositório do projeto

12

INTRODUÇÃO

Este projecto tem como objectivo implementar uma solução em linguagem C que vá ao encontro dos requisitos definidos pelo Grupo I do enunciado da unidade curricular. A proposta insere-se na primeira fase do semestre e visa aplicar, de forma prática, os conteúdos abordados nas aulas iniciais, com especial enfoque em estruturas de dados dinâmicas, nomeadamente apontadores e listas ligadas.

O contexto deste trabalho enquadra-se no desenvolvimento de competências fundamentais no âmbito da programação em C, com ênfase na utilização de memória dinâmica, manipulação de estruturas e organização modular do código. Ao longo do semestre, a unidade curricular propõe a resolução de diferentes grupos de problemas, sendo o Grupo I direccionado para a consolidação dos conceitos básicos, através da implementação de listas ligadas simples, apontadores e operações que permitam aplicar esses conceitos.

O plano de trabalho definido consistiu, numa fase inicial, na análise detalhada do enunciado e dos exemplos disponibilizados pelo docente, os quais serviram como base de apoio à compreensão dos conceitos lecionados. Seguiu-se o planeamento da solução, a implementação das funcionalidades e, por fim, a documentação do processo de desenvolvimento.

Este projecto tem, assim, vários objectivos de aprendizagem, entre os quais:

- Aprofundar o conhecimento sobre apontadores e a sua utilização;
- Compreender o funcionamento e a implementação de listas ligadas;
- Desenvolver a capacidade de estruturar código em C;
- Seguir boas práticas de modularização das soluções;
- Documentar todo o código de forma poder ser compreendido por terceiros.

Durante o desenvolvimento foram tomadas algumas decisões técnicas, motivadas, em grande parte, por dificuldades encontradas na implementação de determinadas funcionalidades. Estas decisões encontram-se devidamente justificadas na documentação da solução, de forma a evidenciar o processo de aprendizagem e as estratégias adoptadas para a superação dos desafios propostos.

FASE I

ANÁLISE E ESPECIFICAÇÃO

A aplicação desenvolvida teve como base os seguintes requisitos funcionais e não funcionais:

REQUISITOS FUNCIONAIS:

- Carregamento de dados a partir de ficheiros de texto (representando uma grelha de antenas).
- Inserção de novas antenas com coordenadas e frequência.
- Remoção de antenas existentes a partir das coordenadas.
- Listagem de todas as antenas registadas na estrutura de dados.
- Detecção de efeito nefasto provocados por antenas na mesma linha e com a mesma frequência.
- Gravação dos dados num ficheiro binário.

REQUISITOS NÃO FUNCIONAIS:

- Utilização de estruturas de dados dinâmicas (listas ligadas).
- Modularização do código com separação por ficheiros `.h` e `.c`.
- Leitura e escrita de ficheiros com verificação mínima de erros.
- Simplicidade na execução e fácil interpretação dos resultados no terminal.

ARQUITETURA LÓGICA / FUNCIONAL

A arquitetura do projeto foi organizada de forma modular, com três componentes principais:

- **dados.h**: Definição das estruturas de dados utilizadas, nomeadamente a estrutura `Antena` e a lista ligada `ListaLigadaAntenas`.
- **antenas.h** / **antenas.c**: Declaração e implementação das funções responsáveis pela manipulação das antenas, como criação da lista, inserção, remoção, carregamento de ficheiro, listagem, deteção de conflitos e gravação de dados.
- **ficheiro de entrada (e.g., antenas.txt)**: Contém a representação das antenas numa grelha bidimensional, onde cada linha representa uma coordenada Y e cada coluna uma coordenada X.

ESTRUTURA DOS DADOS

A estrutura de dados adotada para representar as antenas será uma lista ligada simples. Cada item da lista contém:

- Frequência;
- Coordenadas da antena (int x e y);
- Apontador para o próximo item da lista.

DEFINIÇÃO DA ESTRUTURA ANTENA

```
typedef struct Antena {  
  
    char frequencia;  
  
    int x, y;  
  
    struct Antena* next;  
  
} Antena;
```

IMPLEMENTAÇÃO

ASSINATURA DAS FUNÇÕES

```
ListaLigadaAntenas* criarLista();  
  
void inserirAntena(ListaLigadaAntenas* lista, char frequencia,  
    int x, int y);  
  
void removerAntena(ListaLigadaAntenas* lista, int x, int y);  
  
void listarAntenas(ListaLigadaAntenas* lista);  
  
void carregarFicheiro(ListaLigadaAntenas* lista, const char*  
    nomeFicheiro);
```

```
void detectarEfeitoNefasto(ListaLigadaAntenas* lista);

bool gravarDadosFicheiro(const char* nomeFicheiro,
    ListaLigadaAntenas* lista);
```

CARREGAMENTO DE DADOS

Os dados das antenas serão lidos de um ficheiro binário contendo uma matriz de caracteres de qualquer dimensão. Um exemplo de formato de ficheiro:

```
B   A
B       A
C
D   D
E   E
F
```

Assumindo-se de a linha representa a coordenada X e a coluna representa a coordenada Y. Para efeitos de teste, o projeto contém o ficheiro antenas.txt, sendo este um ficheiro de demonstração com uma estrutura de exemplo para uma lista de antenas.

FUNÇÕES IMPLEMENTADAS

As principais operações foram desenvolvidas no ficheiro antenas.c e organizadas com as suas declarações em antenas.h. Abaixo apresenta-se uma descrição resumida das funcionalidades:

- **criarLista()**: Inicializa uma nova lista ligada, com o apontador head a NULL.
- **inserirAntena(lista, f, x, y)**: Insere uma nova antena no início da lista, atribuindo-lhe os valores recebidos.
- **removerAntena(lista, x, y)**: Remove uma antena da lista com base nas coordenadas fornecidas.

- **listarAntenas(lista)**: Percorre a lista e apresenta no terminal todas as antenas registadas, com respetiva frequência e posição.
- **carregarFicheiro(lista, nomeFicheiro)**: Lê um ficheiro de texto onde cada linha representa uma coordenada Y e cada carácter representa uma posição X. Os caracteres válidos são inseridos na lista como antenas.
- **detectarEfeitoNefasto(lista)**: Verifica se existem duas ou mais antenas com a mesma frequência na mesma linha (coordenada X), sinalizando o conflito.
- **gravarDadosFicheiro(nomeFicheiro, lista)**: Tenta escrever os dados existentes num ficheiro binário. Esta funcionalidade encontra-se ainda com limitações e será revista numa fase posterior.

MODULARIZAÇÃO

O projeto foi modularizado em três ficheiros principais:

- **dados.h**: Define as estruturas de dados utilizadas.
- **antenas.h**: Contém as declarações das funções públicas utilizadas para manipulação das antenas.
- **antenas.c**: Implementa todas as operações sobre a lista ligada de antenas.

FASE II

ANÁLISE E ESPECIFICAÇÃO

A aplicação desenvolvida teve como base os seguintes requisitos funcionais e não funcionais:

REQUISITOS FUNCIONAIS:

- Carregamento de dados a partir de ficheiros de texto (representando uma grelha de antenas).
- Inserção de novas antenas com coordenadas e frequência.
- Remoção de antenas existentes a partir das coordenadas.
- Listagem de todas as antenas registadas na estrutura de dados.
- Detecção de efeito nefasto provocados por antenas na mesma linha e com a mesma frequência.
- Gravação dos dados num ficheiro binário.

REQUISITOS NÃO FUNCIONAIS:

- Utilização de estruturas de grafos com listas ligadas.
- Modularização do código com separação por ficheiros `.h` e `.c`
- Leitura e escrita de ficheiros com verificação mínima de erros.
- Simplicidade na execução e fácil interpretação dos resultados no terminal.

ARQUITETURA LÓGICA / FUNCIONAL

A arquitetura do projeto foi organizada de forma modular, com três componentes principais:

- **grafo_antenas.h /grafo_antenas.c:** Declaração e implementação das funções responsáveis pela manipulação das antenas, como criação do grafo, criação da antena, adicionar antena, adicionar ligações e mostrarGrafo.

ESTRUTURA DOS DADOS

A estrutura de dados adotada para representar as antenas será um grafo com listas ligadas, com a respetiva estrutura:

```
typedef struct Ligacao {  
    . . . Antena* destino;  
    . . . float distancia;  
    . . . Ligacao* seguinte;  
} Ligacao;  
  
typedef struct Antena {  
    . . . char frequencia;  
    . . . float latitude;  
    . . . float longitude;  
    . . . Antena* seguinte;  
    . . . Ligacao* ligacoes;  
} Antena;
```

FUNÇÕES IMPLEMENTADAS

As principais operações foram desenvolvidas no ficheiro antenas.c e organizadas com as suas declarações em antenas.h. Abaixo apresenta-se uma descrição resumida das funcionalidades:

- **InicializarGrafo:** Inicializa o grafo.
- **CriarAntena:** Cria uma Nova antena.

- **AdicionarAntena**(frequencia, coor x, coord y): Adiciona uma nova antena, recebendo a frequencia, coordenada X e coordenada Y.
- **AdicionarLigacao**: Adiciona a ligação entre antenas, considerando apenas uma inserção de 2 antenas de igual frequência,
- **MostrarGrafo**: Lê um ficheiro de texto onde cada linha representa uma coordenada Y e cada carácter representa uma posição X. Os caracteres válidos são inseridos na lista como antenas.

MODULARIZAÇÃO

O projeto foi modularizado em dois ficheiros principais:

- **grafo_antenas.h**: Define as estruturas de dados utilizadas.
- **grafo_antenas.c**: Implementa todas as operações sobre o grafo antenas.

FUNÇÕES IMPLEMENTADAS

```
void inicializarGrafo(Grafo* g);
Antena* criarAntena(float freq, float lat, float lon);
bool adicionarAntena(Grafo* g, Antena* a);
bool adicionarLigacao(Grafo* g, float freqOrigem, float freqDestino, float distancia);
void mostrarGrafo(const Grafo* g);
```

IMPLEMENTAÇÃO DA FUNÇÃO ADICIONARLIGACAO:

```
/*
 * Esta função cria uma ligação entre duas antenas especificadas
 * pelas suas frequências. A ligação inclui a distância entre as antenas.
 */
/*
 * @param g Apontador para o grafo onde as antenas estão localizadas.
 * @param freqOrigem Frequência da antena de origem.
 * @param freqDestino Frequência da antena de destino.
 * @param distancia Distância entre as antenas.
 * @return true se a ligação foi adicionada com sucesso, false caso contrário.
 */
Ligacao* adicionarLigacao(Grafo* g, float freqOrigem, float freqDestino, float distancia) {
    Antena* origem = procurarAntena(g, freqOrigem);
    Antena* destino = procurarAntena(g, freqDestino);
    // verifica se origem ou destino são nulos
    if (!origem || !destino) return false;

    // verificar que frequencia origem/destino são as mesmas
    if (origem->frequencia == destino->frequencia){

        Ligacao* novaLig = (Ligacao*)malloc(sizeof(Ligacao));
        novaLig->destino = destino;
        novaLig->distancia = distancia;
        novaLig->seguinte = origem->ligacoes;
        origem->ligacoes = novaLig;

        // ligação inversa
        Ligacao* ligInv = (Ligacao*)malloc(sizeof(Ligacao));
        ligInv->destino = origem;
        ligInv->distancia = distancia;
        ligInv->seguinte = destino->ligacoes;
        destino->ligacoes = ligInv;
    }

    return true;
}
```

CONCLUSÃO

O presente trabalho teve como principal objetivo a implementação de uma solução para gestão de antenas recorrendo a estruturas de dados dinâmicas, em particular listas ligadas, para representar e organizar de forma eficiente um conjunto de elementos com atributos distintos.

Ao longo do desenvolvimento, foram abordadas operações essenciais como a inserção, remoção, listagem e deteção de conflitos entre antenas com base na frequência e na sua posição no espaço.

A solução desenvolvida permitiu aplicar, de forma prática, os conhecimentos adquiridos ao longo da licenciatura, nomeadamente no que diz respeito à utilização de memória dinâmica, à organização modular do código e à manipulação de ficheiros. A funcionalidade de detecção de

interferências revelou-se especialmente desafiante, e foram assumidas abordagens que o aluno tem consciência de não estarem alinhadas com o resultado esperado.

DIFICULDADES ENCONTRADAS

Durante o desenvolvimento do projeto, surgiram diversas dificuldades que limitaram a concretização plena de alguns objetivos inicialmente definidos:

- **Formatação dos dados inseridos:** Não foi possível garantir que a inserção manual de antenas respeitasse o mesmo formato utilizado na leitura do ficheiro de entrada. Assim, não se conseguiu manter a estrutura de espaçamento (representada por ‘.’) entre os caracteres, o que inviabilizou a correspondência exata entre os valores de coordenadas e a sua representação no ficheiro. Como exemplo, ao inserir uma antena com o comando “A, (4,6)”, seria expectável que o carácter ‘A’ surgisse na linha 4, coluna 6 do ficheiro, o objetivo que não foi possível atingir.
- **Gestão da lista ligada:** A implementação das operações associadas à estrutura dinâmica revelou-se mais complexa do que o previsto, exigindo um maior domínio das técnicas de ligação e controlo dos elementos ao longo da estrutura.
- **Gravação de dados em ficheiro binário:** A funcionalidade destinada à escrita dos dados em ficheiro binário não produziu os resultados esperados, pelo que será necessário proceder a uma revisão cuidada do seu funcionamento numa fase futura do projeto.
- Relativamente à fase II, as dificuldades foram ainda superiores, sendo que não foi possível concluir alguns dos requisitos apresentados

REPOSITÓRIO DO PROJETO

<https://github.com/ivofernandes8819/trabalho-eda-g1>