IPCA INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE

Estruturas de Dados Avançadas (EDA)

Projeto de Avaliação - Fase I

Ivo Fernandes (8819)
Docente
Luís Gonzaga Ferreira

30 Março, 2025

ÍNDICE

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 direcionado 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

Requisitos Funcionais

A aplicação desenvolvida suporta os seguintes requisitos funcionais:

- Carregamento de dados a partir de um ficheiro de texto, com as posições (latitude e longitude) e frequências das antenas.
- Criação e inserção de novas antenas através da função `CriaVertice`, que aloca dinamicamente uma antena com frequência, latitude e longitude.
- Remoção de antenas existentes através da função `RemoveVertice`, que elimina um vértice com base nas coordenadas.
- Listagem das antenas registadas, com respetivas ligações, utilizando a função `mostrarGrafo`.
- Criação de ligações entre antenas com a mesma frequência e localizadas na mesma linha (longitude), através da função `adicionarLigacoes` (a ser implementada conforme estrutura).
- Gravação do grafo num ficheiro binário usando a função `GravaBinário`, que escreve os dados das antenas e respetivas ligações.

Requisitos Não Funcionais

- Utilização de estruturas de grafos com listas ligadas (estrutura `Grafo`, `Antena`, `Ligacao`).
- Código estruturado e modularizado, com separação clara entre os ficheiros de cabeçalho (`grafo_antenas.h`) e de implementação (`grafo_antenas.c`).
- Validação mínima de erros em operações de leitura e escrita de ficheiros.
- Simplicidade na execução, com foco na leitura clara de dados no terminal.

Arquitetura Lógica / Funcional

O projeto foi organizado de forma modular, com dois ficheiros principais:

- `grafo_antenas.h`: Define as estruturas de dados principais, como `Grafo`, `Antena` e `Ligacao`, além dos protótipos das funções.
- `grafo_antenas.c`: Implementa as operações de manipulação do grafo, inserção, remoção, ligação, leitura de ficheiros e visualização do grafo.

Estrutura dos Dados

A estrutura de dados utilizada é um grafo dirigido implementado com listas ligadas, contendo:

- A estrutura `Grafo`, com o Apontador `listaAntenas` que aponta para o início da lista de antenas.
- A estrutura `Antena`, que guarda a frequência (`char`), latitude, longitude e uma lista ligada de ligações (`Ligacao* ligacoes`).
- A estrutura `Ligacao`, que representa as conexões entre antenas (destino e Apontador para a próxima ligação).

```c
#pragma region Estruturas

typedef struct Ligacao {
    // apontador para a struct antena
    // indica antena destino da ligação
    Antena* destino;
    float distancia;
    Ligacao* seguinte;
} Ligacao;

typedef struct Antena {
    char frequencia;
    int latitude;
    int longitude;
    // apontador para criar lista ligada de antenas
    // cada antena aponta para a prox da lista
    Antena* seguinte;
    // apontador que referencia inicio da lista ligada de ligações
    Ligacao* ligacoes;
}Antena;

typedef struct {
    // início da lista ligada de antenas
    Antena* listaAntenas;
} Grafo;

#pragma endregion
```

Funções Implementadas

As principais funções implementadas no ficheiro `grafo_antenas.c` são:

- `inicializarGrafo`: Inicializa a estrutura `Grafo`, definindo o início da lista como `NULL`.
- `CriaVertice`: Cria dinamicamente uma nova antena com dados fornecidos.
- `InsereVertice`: Insere uma antena na lista de forma ordenada por coordenadas.
- `RemoveVertice`: Remove uma antena com base nas suas coordenadas.
- `mostrarGrafo`: Percorre todas as antenas e imprime os dados e ligações associadas.
- `carregaGrafoficheiro`: Lê um ficheiro de texto com dados das antenas e constrói o grafo.
- `GravaBinário`: Grava as antenas e suas ligações num ficheiro binário.

Modularização

O projeto foi desenvolvido com base na separação clara entre declarações e implementações, de forma a garantir maior organização, manutenção e reutilização do código:

- **grafo_antenas.h**
  - As definições das estruturas de dados (`Grafo`, `Antena`, `Ligacao`).
  - Os protótipos das funções usadas no programa, tais como:
    - inicializarGrafo
    - CriaVertice
    - InsereVertice
    - RemoveVertice
    - mostrarGrafo
    - GravaBinário
    - carregaGrafoficheiro

- **grafo_antenas.c**
  - A implementação completa das funções declaradas no ficheiro `.h`.
  - A gestão de memória dinâmica para criação de antenas e ligações.

- A lógica para:
    - Inserção ordenada de antenas.
    - Criação e visualização do grafo.
    - Remoção de antenas.
    - Gravação e leitura a partir de ficheiros.

A estrutura segue regiões com `#pragma region` para facilitar a navegação entre blocos funcionais, como:

- #pragma region **grafos**
- #pragma region **operações antenas**
- #pragma region **exibir dados**
- #pragma region **Dados e Ficheiros**

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