Técnicas de Programação – LEEC (2023/2024)
Trabalho Prático de Projeto Final

Gestão de Rede de Metro

Objetivos

Aplicar os conhecimentos de programação em C obtidos na disciplina

- Utilização eficiente de estruturas de dados
- Ponteiros e alocação dinâmica de memória
- Escrita e leitura de ficheiros

Trabalho Prático – Gestão do Metro

1. Introdução

A empresa que gere o sistema de mobilidade de uma rede de Metro (por. Ex. o Metro de Lisboa) pretende um programa que efetue a gestão das várias linhas existentes. Este programa deverá manter informação atualizada sobre as linhas que constituem o sistema e sugerir percursos entre 2 paragens.

2. Cada linha do sistema Metro de Lisboa é constituída por várias paragens. Em cada paragem podem entrar e sair pessoas. As linhas podem ser percorridas nos 2 sentidos. Pode considerar que as linhas não são circulares e nenhuma delas passa 2 vezes pela mesma paragem.
   Exemplo:
   A *Linha Amarela* é constituído por 13 paragens, entre o Rato e Odivelas. Pode ser percorrida nos seguintes sentidos:
   Rato ⇒ Marques de Pombal ⇒ … ⇒ Senhor Roubado ⇒ Odivelas
   Odivelas ⇒ Senhor Roubado ⇒ … ⇒ Marques de Pombal ⇒ Rato

   O sistema é constituído por diversas linhas que se cruzam em algumas paragens. Nesses locais, um passageiro pode sair de uma das linhas e entrar numa outra.

2. Programa a Implementar

Pretende-se que desenvolva um programa em linguagem C que permita efetuar a gestão do sistema de mobilidade de uma rede de Metro. As funcionalidades previstas são descritas nos pontos seguintes.

2.1 Paragens

As paragens pertencentes ao sistema são identificadas por um nome e por um código alfanumérico com 4 caracteres. Tanto o nome, como o código alfanumérico são únicos, ou seja, não podem existir 2 paragens com o mesmo nome e/ou o mesmo código alfanumérico. Pode adicionar outros campos, caso julgue conveniente (por exemplo, a quantas linhas pertence uma determinada paragem ou outra informação relevante). Durante a execução do programa, esta informação está armazenada num **array de estruturas dinâmico**.

2.1.1 Operações sobre Paragens

- **Registar Paragem**: Podem ser adicionadas novas paragens ao sistema. O nome é introduzido pelo utilizador, mas o código alfanumérico único deve ser gerado automaticamente pelo programa (este campo não é especificado pelo utilizador). Quando uma nova paragem é adicionada, não fica associada a nenhuma linha. Isso será efetuado mais tarde (ver ponto 2.2.1).

- **Eliminar Paragem**: Pode ser eliminada uma paragem do sistema. O código da paragem a eliminar é introduzido pelo utilizador. Só podem ser eliminadas paragens que, nessa altura, não façam parte de nenhuma linha.

- **Visualizar Paragens:** A lista completa de paragens existentes no sistema pode ser apresentada na consola.

2.2 Gestão de Linhas

A informação completa sobre as linhas do metro deve ser mantida numa estrutura dinâmica do tipo **lista ligada**. Cada linha (lista ligada), pode estar referenciada a um **array dinâmico** que agrupa todas as linhas (pode usar outra estrutura de dados que achar mais conveniente para este efeito).

2.2.1 Operações sobre Linhas

- **Adicionar Linhas**: Podem ser adicionadas novas linhas ao sistema de mobilidade. A adição pode ser efetuada a partir de informação introduzida pelo utilizador ou lendo esses dados de um ficheiro de texto[1]. Toda a informação necessária para criar a linha deve ser especificada, incluindo o seu nome e as várias paragens. Não é possível ter duas linhas com o mesmo nome, nem incluir na linha paragens não registadas no sistema.

- **Atualizar Linha**: A sequência de paragens de uma linha pode ser atualizada. A atualização pode consistir na adição ou eliminação de uma ou mais paragens.

- **Visualizar Linhas**: O programa deve permitir a apresentação completa de todas as linhas existentes no sistema. Também deve ser possível mostrar as linhas que passem numa determinada paragem.

2.3 Cálculo de Percursos

Nesta funcionalidade, o programa deverá apresentar uma listagem com todos os percursos que liguem 2 paragens. O utilizador indica os nomes do ponto de partida e do ponto de chegada e o programa mostra quais as linhas que permitem fazer a ligação. Deve ser indicado todo o percurso, isto é, todas as paragens entre o ponto de partida e o ponto de chegada.

A listagem deve considerar 2 possibilidades:

i. Percurso efetuado numa única linha
ii. Percurso com uma mudança de linha[2]

2.4 Armazenamento em Ficheiro

Imediatamente antes de terminar a execução, o programa deve guardar a informação das paragens e das linhas num **ficheiro binário**. Esta informação deverá permitir reconstruir as estruturas dinâmicas (array dinâmico e lista ligada) quando o programa retomar a execução.

[1] O formato do ficheiro de texto é descrito no ponto 3.
[2] Os percursos entre o ponto inicial e o final poderão ter, no máximo, um transbordo

3. Ficheiro de Texto

A informação sobre uma nova linha a adicionar ao sistema pode ser especificada através de um ficheiro de texto. Cada ficheiro de texto só pode conter informação sobre uma linha.

```
Linha Amarela
Rato  #  P011
Marquês de Pombal  #  P123
Picoas  #  Q998
Saldanha  #  F554
Campo Pequeno  #  H123
…
Odivelas  #  G234
```

O formato é o seguinte:

i. - Na primeira linha do ficheiro surge o nome da nova linha (i.e., o nome do percurso) do metro
ii. - Nas linhas seguintes do ficheiro surgem o nome e o identificador alfanumérico das paragens (1 paragem por linha).

O ficheiro ao lado ilustra a informação necessária para construir a linha descrita no exemplo do ponto 1.

Rato ⇒ Marques de Pombal ⇒ … ⇒ Senhor Roubado ⇒ Odivelas
Odivelas ⇒ Senhor Roubado ⇒ … ⇒ Marques de Pombal ⇒ Rato

O caracter '#' separa o nome da paragem do seu identificador alfanumérico. Pode assumir que este caracter nunca faz parte do nome ou do identificador e atua apenas como separador.

Fases de Resolução e Entrega do Trabalho

- O trabalho terá de ser entregue até através do Moodle até ao dia:
  - Época de frequência: **13/06/2024**
  - Época de Exame: **24/06/2024**
  - Época de Recurso: **15/07/2024**
- Grupos de **2 alunos** (sendo aceites trabalhos individuais)
- Submetido através do moodle (e-learning) com o nome "**trabalho_tp_XXXXX_YYYYY.zip**"
- Os valores XXXXX e YYYYY são os números dos alunos do grupo
- Deve conter o código (com o projeto do codeblocks) e relatório (em formato pdf)

Relatório

- O relatório deve ser curto. O objectivo é explicar de forma clara (e bem resumida) o trabalho que foi realizado pelo grupo, as opções tomadas no planeamento e como este foi implementado.
- Deverá conter:
  - Identificação dos alunos
  - Introdução – Uma breve descrição do projecto (em que consiste?)
  - Trabalho desenvolvido – O que foi feito? Como foi feito?
  - Que funcionalidades implementaram? Quais destas são extra?
  - Que estruturas criaram / utilizaram? Quais as razões?
  - Como está organizado o código (bibliotecas e funções existentes)?

    - Eventuais imagens de ecrãs que pensem ser relevantes para mostrar o que fizeram
    - Conclusões – algumas notas / reflexão sobre o projecto
- Correu bem? O trabalho final atingiu os objetivos?
- Que limitações contém? Onde sentiram mais dificuldades?
- O que gostariam de ter feito mais?

Defesa

As defesas serão realizadas **presencialmente em sala a designar**, sendo obrigatória para aprovação do trabalho, durante a tarde do último dia da entrega em cada época.

O que será avaliado?

- Relatório
    - Está curto e bem escrito?
    - Explica corretamente o que fizeram?
- Estruturas de dados utilizadas
    - Que estruturas de dados utilizaram?
    - Porquê da sua escolha?
- Funcionalidade base do trabalho
    - As funcionalidades base foram implementadas?
    - Estão a funcionar?
- Existem funcionalidades adicionais?
    - Em que consistem?
    - Estão a funcionar?
- Qualidade e organização de código
    - O código está bem organizado?
    - Está devidamente comentado?

Defesa

As defesas serão realizadas **presencialmente em sala a designar**, sendo obrigatória para aprovação do trabalho.

Sugestões para a realização do trabalho

Para realizar o trabalho de forma eficiente e bem-sucedida é essencial que exista um bom planeamento do mesmo, assim como boa gestão do tempo e esforço. O plano seguinte contém algumas linhas orientadoras, para evitar mau projeto.
A ideia essencial é:
1) **Planear** antes de iniciar a implementação
2) **Dividir a implementação** em problemas menores

a) Tarefa 1: Planeamento do trabalho (em papel) – Semana #1 (máx 1s)

- Ler com atenção o enunciado do trabalho
    - Percebi a ideia geral do trabalho? Brainstorming nas aulas!
- Definir que informação querem guardar para um <entidade>
    - Por exemplo: Jogo? Jogadas? restrições?

- Que estruturas podem usar para guardar esses dados na aplicação
  - Arrays, Matrizes, linked lists, …
- Pensar na organização do código e nas funcionalidades que querem implementar
  - Identificar algumas funções mais óbvias que serão necessárias
- Para garantir uma melhor organização de código (p. ex. *criar_<entidade>*?)
  - Ter uma ideia das bibliotecas que pensam criar ou reutilizar

b) Tarefa 2: Definição das estruturas base – Semana #2 (máx 1s)
- Criar o projecto no *Code::Blocks*
- Definir as estruturas base a usar (para representar um <entidade>, outras coisas?)
- Criar primeiras funções para interagir com estas estruturas
  - Por exemplo: *imprimir_*<entidade>*?* ou *criar_*<entidade>*?*
- Testar as estruturas e funções criadas

c) Tarefa 3: Interação base com o utilizador (funções) – Semana #2 e #3 (máx 1s)
- Implementar um menu básico (modo texto) para apresentar as opções assim como navegação entre as diversas opções
- Criar / adaptar as funções necessárias para ler dados do utilizador (input)

d) Tarefa 4: Implementar estruturas adicionais – Semana #3, #4, #5 (máx 2s)
- Adicionar as bibliotecas necessárias para guardar dados (arrays, matrizes, listas, …)
  - Adaptar as mesmas conforme for necessário
- Integração destas estruturas com o código já existente
  - Ex: ao criar um <entidade> este deve ser adicionado corretamente à estrutura

e) Tarefa 5: Guardar dados, funcionalidade extra – Semana #5 e #6 (máx 1s)
- Criar funções para ler e escrever dados para um ficheiro

f) Tarefa 6: Testar, escrever relatório – Semana #6 (máx 1s)
- Testar as funcionalidades, corrigir bugs encontrados
- Escrever o relatório