Estruturas de Dados I
Enunciado do Trabalho Prático

Licenciatura em Engenharia Informática
Universidade de Évora

2001/2002

1 Objectivo

Pretende-se implementar uma base de dados sobre as viagens efectuadas ao estrangeiro pelos amigos (e pelos amigos dos amigos), durante o ano de 2001.

Em traços gerais, poder-se-á inserir amigos, destinos e viagens (indicando, neste último caso, a sequência de deslocações e os amigos envolvidos); cancelar viagens; registar a desistência de um amigo de uma viagem; e efectuar pesquisas sobre viagens (realizadas num período do ano ou por um amigo).

2 Conceitos e Princípios

Cada *viagem* é efectuada por um *grupo* de (um ou mais) *amigos* e consiste numa sequência de, pelo menos, duas *deslocações*. Uma deslocação ocorre numa *data* (caracterizada pelo *mês* e pelo *dia*) e tem um *destino*, que é identificado por uma *localidade* e um *país*.

Por exemplo, uma viagem de ida e volta a Paris, realizada pela Maria e pelo Manel, seria associada a um grupo de dois amigos (a Maria e o Manel) e a uma sequência de duas deslocações: no dia 7 de Abril o destino seria Paris, em França; no dia 15 de Abril o destino seria Lisboa, em Portugal.

Para simplificar, assume-se que todas as viagem têm origem em Portugal, e exige-se que o destino também seja no nosso País. Considera-se que qualquer deslocação é totalmente efectuada num dia, ocupando completamente esse dia (i.e., uma pessoa parte e chega no mesmo dia, mas não pode realizar mais do que uma deslocação por dia). As datas de início e de conclusão de uma viagem são as datas da primeira e da última deslocação, respectivamente.

O sistema deverá garantir: que todos os amigos e destinos referidos nas diferentes primitivas foram previamente inseridos (excepto nas respectivas primitivas de inserção, nas quais se exige o contrário); que o percurso induzido pela sequência de deslocações é cronologicamente possível; que não existem viagens com destinos intermédios em Portugal (nem com destino final no estrangeiro); e que ninguém está simultaneamente em dois lugares distintos. Para este efeito, o sistema deverá ignorar qualquer primitiva de inserção que tente violar estas regras.

3 Entrada e Saída

O funcionamento da aplicação deverá ser o seguinte. Depois de arrancar, começará a ler comandos da entrada padrão (`System.in`), processando-os um a um e enviando as respostas para a saída padrão (`System.out`). A terminação ocorrerá quando for atingido o fim de ficheiro na entrada padrão.

3.1 Sintaxe

Pretende-se que a interface da aplicação seja muito simples, de modo a poder ser utilizada em ambientes diversos (em batch, interactivamente, dentro de um script, executada por um servidor WWW, etc.) e, no caso da saída, de forma a permitir automatizar o processo de teste. Por estes motivos, a entrada e a saída deverão respeitar o formato rígido que se indica na Secção 4. Convém, no entanto, prestar alguns esclarecimentos sobre a notação utilizada.

O símbolo $\hookleftarrow$ representa uma mudança de linha.

Em todas as primitivas apresentadas na Secção 4.1, a saída é uma mensagem, especificando-se quais as mensagens possíveis em cada caso. A mensagem a devolver deverá ser a primeira (da sequência dada) que corresponde a uma afirmação verdadeira. Note que nenhuma destas primitivas deve produzir qualquer efeito na base de dados, se a mensagem devolvida não for a última (que informa que a operação foi efectuada com sucesso).

No caso das pesquisas (c.f. Secção 4.2), as linhas que correspondem à entrada deverão sempre ocorrer na saída. As restantes linhas dependem da existência da informação pesquisada.

As duas primitivas de pesquisa produzem uma listagem de viagens, cujo formato deverá ser sempre igual.

- Se a listagem for vazia, ou seja, quando não existirem viagens nas condições especificadas, a saída deverá ser exactamente igual à entrada.
- Se a listagem não for vazia, dever-se-á começar por repetir as linhas da entrada (à excepção da última, que só tem um símbolo de mudança de linha), terminando a informação referente a cada viagem com uma "linha em branco".

Estas regras serão exemplificadas apenas na primeira primitiva.

3.2 Tipo e Dimensão dos Dados

As entidades *amigo*, *localidade* e *país* são cadeias de, no máximo, vinte caracteres. O *mês*, o *mês-inicial* e o *mês-final* são números entre `1` e `12`. O *dia*, o *dia-inicial* e o *dia-final* são números entre `1` e `31`.

Sabe-se que o número de amigos e o número de destinos (i.e., o número de pares <localidade, país>) inseridos na base de dados nunca excedem mil. O número de países não ultrapassa cem.

Relativamente aos restantes números (total de viagens, de amigos que viajam juntos (dimensão dos grupos), de deslocações numa viagem, de viagens efectuadas por uma pessoa, etc.), não existe qualquer restrição. Contudo, prevê-se que, em noventa e cinco por cento dos casos, os grupos não serão compostos por mais do que seis pessoas, as viagens não terão mais do que vinte deslocações e uma pessoa não efectuará mais do que cinco viagens.

3.3 Correcção da Entrada

Poderá admitir que a entrada está sintacticamente correcta, mas não deverá assumir que ela está logicamente correcta. Por exemplo, poder-se-á pedir a listagem das viagens efectuadas entre 15 de Novembro e 31 de Fevereiro (com duas fontes de erro) ou a listagem das viagens realizadas em Janeiro por um amigo inexistente.

4 Primitivas a Implementar

4.1 Inserções e Remoções

- Inserção de um Amigo
  - Sintaxe da Entrada
    `IA` *`amigo`*↩
    ↩
  - Sintaxe da Saída
    `mensagem-de-inserção-de-amigo`↩
    ↩
  - A *mensagem-de-inserção-de-amigo* deve ser uma das seguintes.
    * `Amigo existente.`
    * `Inserção de amigo com sucesso.`

  Inserção de um novo amigo, com o nome *amigo*.

- Inserção de um Destino
  - Sintaxe da Entrada
    `ID` *`localidade`*↩
    `país`↩
    ↩
  - Sintaxe da Saída
    `mensagem-de-inserção-de-destino`↩
    ↩
  - A *mensagem-de-inserção-de-destino* deve ser uma das seguintes.
    * `Destino existente.`
    * `Inserção de destino com sucesso.`

  Inserção de um novo destino, identificado pela localidade *localidade* no país *país*.

- Inserção de uma Viagem
  - Sintaxe da Entrada
    `IV` *`mês dia localidade`*↩
    `país`↩
    `mês dia localidade`↩
    `país`↩

```
.........
mês dia localidade↩
país↩
↩
amigo↩
amigo↩
.........
amigo↩
↩
```

- Sintaxe da Saída
  mensagem-de-inserção-de-viagem↩
  ↩
- A *mensagem-de-inserção-de-viagem* deve ser uma das seguintes.
  * `Data(s) inexistente(s).`
  * `Percurso cronologicamente impossível.`
    Sequência de deslocações não ordenada por ordem estritamente crescente.
  * `Percurso inválido.`
    Número insuficiente de deslocações, ou Portugal é o país de destino de uma deslocação que não é a última, ou Portugal não é o país de destino da última deslocação.
  * `Destino(s) inexistente(s).`
  * `Amigo(s) inexistente(s).`
  * `Amigo(s) indisponível(eis).`
    Sobreposição com outra viagem.
  * `Inserção de viagem com sucesso.`

Inserção de uma nova viagem, composta pela sequência de deslocações especificada e efectuada pelo grupo de amigos indicados.

Garante-se que as sequências de deslocações e de amigos nunca são vazias.

- Cancelamento de uma Viagem
  - Sintaxe da Entrada
    `CV` *mês dia amigo*↩
    ↩
  - Sintaxe da Saída
    mensagem-de-cancelamento-de-viagem↩
    ↩
  - A *mensagem-de-cancelamento-de-viagem* deve ser uma das seguintes.
    * `Data inexistente.`
    * `Amigo inexistente.`
    * `Viagem inexistente.`
    * `Cancelamento de viagem com sucesso.`

Remoção da viagem com data de início no dia *dia* do mês *mês*, efectuada pelo grupo de amigos ao qual o amigo *amigo* pertence.

- Desistência de um Amigo de uma Viagem

  - Sintaxe da Entrada
    `DA` *`mês dia amigo`*↩
    ↩
  - Sintaxe da Saída
    `mensagem-de-desistência-de-amigo`↩
    ↩
  - A *mensagem-de-desistência-de-amigo* deve ser uma das seguintes.
    * `Data inexistente.`
    * `Amigo inexistente.`
    * `Viagem inexistente.`
    * `Desistência inaceitável.`
      Ausência de companheiros de viagem.
    * `Desistência de viagem com sucesso.`

  Relativamente à viagem com data de início no dia *dia* do mês *mês*, efectuada pelo grupo de amigos ao qual o amigo *amigo* pertence, remoção do amigo *amigo* (desse grupo de amigos).

4.2 Pesquisas

- Pesquisa das Viagens Efectuadas num Intervalo de Tempo

  - Sintaxe da Entrada
    `VI` *`mês-inicial dia-inicial mês-final dia-final`*↩
    ↩
  - Sintaxe da Saída (igual à da entrada ou da forma seguinte)
    `VI` *`mês-inicial dia-inicial mês-final dia-final`*↩
    `mês dia localidade`↩
    `país`↩
    `mês dia localidade`↩
    `país`↩
    .....
    `mês dia localidade`↩
    `país`↩
    `amigo`↩
    `amigo`↩
    .....
    `amigo`↩
    ↩
    `mês dia localidade`↩
    `país`↩
    `mês dia localidade`↩
    `país`↩
    .....
    `mês dia localidade`↩

```
país↩
amigo↩
amigo↩
.........
amigo↩
↩
.........
mês dia localidade↩
país↩
mês dia localidade↩
país↩
.........
mês dia localidade↩
país↩
amigo↩
amigo↩
.........
amigo↩
↩
```

Lista toda a informação sobre as viagens realizadas (ou seja, iniciadas e concluídas) entre o dia *dia-inicial* do mês *mês-inicial* e o dia *dia-final* do mês *mês-final* (inclusive).

As viagens deverão aparecer ordenadas cronologicamente pela data de início e, em caso de igualdade desta, pela data de conclusão. No âmbito de cada viagem, a sequência de deslocações deverá estar ordenada cronologicamente e a sequência de amigos deverá respeitar a ordem alfabética.

- Pesquisa das Viagens Efectuadas num Mês por um Amigo

  – Sintaxe da Entrada
  `VA` *mês amigo*↩
  ↩

  – Sintaxe da Saída
  (semelhante ao caso anterior — c.f. final da Secção 3.1)

  Lista toda a informação sobre as viagens realizadas (ou seja, iniciadas e concluídas) durante o mês *mês*, pelo amigo *amigo*.

  As viagens deverão aparecer ordenadas cronologicamente pela data de início. No âmbito de cada viagem, a sequência de deslocações deverá estar ordenada cronologicamente e a sequência de amigos deverá respeitar a ordem alfabética.

5 Desenvolvimento e Avaliação

O trabalho deverá ser implementado em Java, nomeadamente, em JDK 1.2 (Linux). Não é permitido fazer uso do *package java.util*.

O trabalho deverá ser entregue até às 20 horas do dia 11 de Janeiro de 2002. Deverá ser entregue, *em papel*, uma descrição das estruturas de dados utilizadas, incluindo a justificação das escolhas feitas. Deverão, também, ser entregues todos os ficheiro com o código

do programa, com instruções para a sua compilação e execução. Os ficheiros poderão ser enviados para `vp@di.uevora.pt` e a descrição das estruturas de dados pode ser entregue, em mão, a qualquer um dos docentes ou deixada no cacifo do docente das teóricas, sito frente ao gabinete CLV-226. Um trabalho só será considerado entregue quando o forem todos os elementos referidos.

O trabalho, a ser realizado em grupos de dois ou três alunos, deverá executar correctamente todas as operações indicadas. Só serão considerados os trabalhos que funcionem correctamente para o exemplo de interacção fornecido, que é constituído por dois ficheiros. Um contém uma sequência de comandos[1] a ler e executar pelo programa, e no outro é apresentada a correspondente sequência de respostas[2]. Se o método `main` pertencer à classe `Viagens` do *package* `Trabalho`, se o ficheiro com os comandos for chamado `teste.entrada` e o ficheiro com as respostas for chamado `teste.saida` e se estes residirem na directoria que contém a directoria `Trabalho`, a correcção do programa, em relação a estes testes, pode ser testada através do comando

```
java Trabalho.Viagens < teste.entrada | diff teste.saida -
```

Se o funcionamento da aplicação estiver de acordo com o enunciado, o comando terminará sem nada ser escrito no terminal, caso contrário, mostrará as diferenças entre as respostas correctas e as dadas pela aplicação.

O trabalho será classificado entre 0 e 5 valores. A avaliação incidirá sobre todos os aspectos: concepção, qualidade e eficiência da solução, modularidade, estrutura e documentação do código, qualidade do relatório, etc.

Bom Trabalho.

[1] Ver em `http://www.di.uevora.pt/~vp/ed/teste.entrada`.
[2] Ver em `http://www.di.uevora.pt/~vp/ed/teste.saida`.