ATAD 2021/22

Algoritmos e Tipos Abstratos de Dados

Enunciado do Projeto da Época de Recurso

Voos domésticos americanos

(Processamento de dados, atrasos e rotas)

1. Objetivo do Projeto

Pretende-se desenvolver um programa em C para extrair/apresentar informação útil de ficheiros com dados sobre voos domésticos no território americano, nos primeiros dias de janeiro de 2015.

O programa consiste num interpretador de comandos que o utilizador usa para obter diversos tipos de informação, principalmente informação estatística.

1.1 Representação dos dados em Memória

Cada voo registado, é representado, **obrigatoriamente**, pela estrutura de dados `Flight` apresentada na Figura 1, os aeroportos pela estrutura de dados `Airport` e as companhias aéreas pela estrutura `Airline`.

```
typedef struct flight{
    int day;
    int dayOfWeek;
    char airline[3];
    int flightNumber;
    char originAirport[4];
    char destinationAirport[4];
    Time scheduledDeparture;
    Time departureTime;
    int realTravelTime; // in minutes
    int distance;            // in miles
    Time scheduledArrival;
    Time arrivalTime;
    int totalDelay;       // in minutes (sum of arrival and departure delays)
                          // departing or arriving before the scheduled time is not a delay
} Flight;

typedef struct airport{
    char iatacode[4];
    char airport[100];
    char city[35];
    char state[3];
    int timezome;
} Airport;


typedef struct airline{
   char code[3];
   char airline[100];
   int nFlights;
} Airline;

typedef struct time{
   int hour, min, sec;
} Time;
```

Figura 1 – Definição de Tipos de dados.

Na implementação dos comandos descritos neste enunciado podem definir/utilizar outros tipos de dados auxiliares que achem úteis para a resolução dos problemas.

1.2 Dados de entrada

São disponibilizados 3 ficheiros de entrada para testes:

- airlines.csv - Dados sobre as companhias aéreas;
- airports.csv - Dados com informações sobre os aeroportos;
- flights.csv - Registo dos voos domésticos dos primeiros dias de janeiro de 2015 nos Estados Unidos;

Todos os ficheiros se encontram no formato CSV; a primeira linha dos ficheiros é uma linha com os *cabeçalhos* e não contém dados.

Ficheiro dos voos (cada linha corresponde a informação sobre um determinado voo)

```
<day>;<day_of_week>;<airline >;<flight_number>;<origin_airport >;<destination_airport>;<scheduled_departure>;
<departure_time>;<distance>;<scheduled_arrival>;<arrival_time>
...
```

O campo *day* guarda o dia do mês e o *day_of_week* representa o dia da semana, sendo que segunda-feira equivale ao valor 1.

As horas dos campos *scheduled_departure, departure_time, scheduled_arrival e arrival_time* são representadas por um número inteiro, com 4 dígitos. A oclusão dos dígitos à esquerda representa o valor 0.
Ex: o valor 30 equivale às horas 00:30.

Ficheiro com os dados dos aeroportos

```
<iata_code>;<airport>;<city>;<state>;<latitude;<longitude>;<timezone>
...
```

O campo *timezone*, representa um inteiro com a diferença horária em relação à hora no meridiano de Greenwich (MGT), em janeiro.
Ex: New York tem o valor de -5 e em Los Angeles -8.

Ficheiro com os dados das companhias aéreas

```
<iata_code>;<airline>
...
```

1.3 Utilização de ADTs

É obrigatória a manutenção em memória da informação importada:

- Dos voos- exclusivamente numa instância do ADT List, sendo **ListElem** o tipo **Flight** (definido em 1.1)

- Dos aeroportos - exclusivamente numa instância de ADT Map, sendo **ValueElem** do tipo **Airport** (definido em 1.1) e o **KeyElem** de um tipo apropriado que permita guardar uma *string* (código do aeroporto, a chave do dicionário);
- Das companhias aéreas – como ~~são poucos dados~~estas são fixas e imutáveis, deve usar um array (estático ou dinâmico) para guardar essa informação.

Não é permitido (nem necessário!!) alterar as interfaces lecionadas dos ADT, nomeadamente os ficheiros list.h e map.h. Estas instâncias serão designadas doravante por "coleções". Se achar apropriado/útil utilizar o ADT Queue ou ADT Stack como coleção auxiliar para uma qualquer operação, pode adicionar estes ADTs ao projeto.

1.4 Comandos

Há exatamente **15 comandos que o programa deve implementar**, que serão apresentados de seguida; 3 comandos para carregamento de dados, 10 comandos para mostrar resultado de cálculos sobre os dados, 1 comando para sair da aplicação e 1 comando para limpeza dos dados em memória.
Os comandos têm o seguinte grau de dificuldade previsto: BAIXA e MÉDIA

Notas:

- Cada comando é representado por uma palavra que pode ser escrita pelo utilizador em maiúsculas ou em minúsculas, não importa.
- Sempre que um comando necessitar de algum input, e.g., nome da companhia aérea, este deve ser solicitado ao utilizador.
- Sempre que um comando necessitar de informação que não está carregada, o comando deve indicar que informação está em falta, i.e., **`"No flight data available…"`**. e/ou **`"No airport data available…"`**.

A forma exata como os resultados devem ser mostrados no ecrã será descrita em seguida. **Não se deve assumir que os ficheiros de entrada estão ordenados**.

A. Os *comandos base* são os seguintes:

✓ **LOADAP**

- Abre o ficheiro "airports.csv" e carrega-o em memória (ver 1.2), mostrando o número de aeroportos importados, e.g., **`"<N> airport records imported"`**.

  Se o ficheiro não puder ser aberto, escreve "**`File not found"`** e a coleção respetiva fica vazia.

✓ **LOADF**

- Abre o ficheiro "flights.csv" e carrega-o em memória (ver Secção 1.2), mostrando o número de dados de voos importados, e.g., **`"<N> flight records imported"`**.

  Este comando necessita que o ficheiro "airports.csv" esteja carregado, para poder calcular o ***realTravelTime*** da estrutura **Flight** (Secção 1.1).

  Se o ficheiro não puder ser aberto, escreve "**`File not found"`** e a coleção fica vazia. Se os registos dos aeroportos ainda não tiverem sido lidos, apresenta a mensagem "**`Please load airport data first"`**.

✓ **LOADAR**

- Abre o ficheiro "airlines.csv" e carrega-o em memória (ver 1.2), mostrando o número de companhias aéreas importadas, e.g., **"<N> airline records imported"**.

  Este comando necessita que o ficheiro "flight.csv" esteja carregado, para poder calcular o ***nFlights*** da estrutura **Airline** (Secção 1.1).

  Se o ficheiro não puder ser aberto, escreve "**File not found"** e a coleção fica vazia. Se os registos dos voos ainda não tiverem sido lidos, apresenta a mensagem "**Please load flight data first"**.

✓ **CLEAR**

- Limpa a informação atualmente em memória. Deverá indicar o número de registos que foram descartados, e.g., **"<N> records deleted from <Flights | Airports | Airlines>"**

✓ **QUIT**

- Sai do programa, libertando toda a memória alocada para as coleções.

B. Os comandos de indicadores simples são os seguintes:

✓ **SHOWALL**

- Mostra todos os dados dos voos disponíveis nos registos.

Deve solicitar ao utilizador uma escolha; se pretender visualizar:

- **ALL** – Apresenta todos os registos de forma paginada. Cada página deverá, no máximo, 30 voos e deverá ser possível navegar para páginas seguintes.
- **SAMPLE** – Apresenta uma amostragem aleatória de 80 registos (será sempre diferente).

✓ **SHOWF**

- Mostra os dados de todos os voos que têm como rota (partida e chegada) a rota indicada solicitada ao utilizador **(solicitar aeroportos de partida e de chegada)**. Caso a rota que inseriu não tenha dados disponíveis na coleção, escreve "**Flight data not available for route <Airport Code> ==> <Airport Code>"**.

  Deve calcular primeiro uma lista com os dados dos voos do aeroporto selecionado numa função e mostrá-los noutra função.

✓ **LISTAR**

- Mostra a lista de companhias aéreas (nomes únicos) existentes com registos de voos.

✓ **LISTAP**

- Mostra a lista de aeroportos (nomes únicos) existentes com registos de voos, quer sejam em partidas ou chegadas.
- Os atributos a mostrar são os seguintes:
  "**<iata_code>: <airport> <city> <state> <utc>".**
  Onde <utc> é o *timezone*, mas apresentado no formato oficial:
  https://en.wikipedia.org/wiki/Time_zone
  Ex: Timezone -3 ➔ UTC-03:00 , Timezone 10 ➔ UTC+10:00

✓ **DELAYS**

- Mostra, para cada companhia aérea:
  - O número de voos que partiram fora de horas.
  - O número de voos que chegaram fora de horas.

O utilizador deve indicar um número de 0 a 30 (minutos) até ao qual o voo é considerado "dentro da hora". Partir ou chegar antes da hora não é considerado "fora de horas".

✓ **AVERAGEDELAY**

- Para todos os dias, só dias de semana e só fins de semana, calcular:
  - A média global dos atrasos em todos os voos.
  - A média dos atrasos de todos os voos de uma dada **companhia aérea, solicitada ao utilizador**.

Partir antes da hora não é atraso e não deve ser somado, qualquer valor depois da hora é atraso.

Os resultados são apresentados numa matriz 3x2, sendo que as 3 linhas representam, respetivamente, todos os dias, só dias de semana e só fins de semana.

A primeira coluna da matriz corresponde ao cálculo das médias globais e a segunda às médias calculadas para o aeroporto introduzido.

No cálculo das duas médias anteriores para cada uma delas, deve implementar uma função que calcule os valores e outra que as mostre.

✓ **SHOWAP**

- Mostra para cada um dos aeroportos a lista de companhias cujos voos passam por esse aeroporto (partidas e chegadas).

Se o aeroporto não tiver voos não deve ser mostrado nada para esse aeroporto

✓ **TOPDISTN**

Mostra os N voos com maior distância percorrida, ordenados crescentemente. **O valor N deve ser solicitado ao utilizador**.

Em caso de empate, ordene pelo número de voo de forma **decrescente**.

C. Os comandos de indicadores complexos são os seguintes:

✓ **AIRPORT_S**

- Mostra os dados dos aeroportos ordenados por um atributo (dos abaixo apresentados), **escolhido pelo utilizador**. Deverá ser possível ordenar por:
  - Estado (ordem crescente e decrescente).
  - Dado uma time zone, apresentar os aeroportos por ordem de proximidade crescente de time zone (valor absoluto da diferença entre time zones). Desempate por ordem crescente de IATA code.
    - **Ex:** Dado a time zone -6, serão mostrados primeiro os aeroportos de time zone -6 (valor absoluto da diferença é zero), depois os de time zone -5 e -7 (diferença de 1 em valor absoluto), depois -4 e -8 (diferença de 2 em valor absoluto), etc.

✓ **AIRLINES**

Mostra, **agregados por companhia aérea**, os seguintes dados de voos disponíveis:

- *Iata code*
- *Nome*
- *Número de voos da companhia*
- *Distância mínima, média e máxima dos voos operados pela companhia*.

No final deverá listar, **para todas as companhias aéreas**, os seguintes cálculos:

- *Média do número de voos por companhia*
- *Média das distâncias de todos os voos*

As companhias aéreas que não tenham registo de voos não devem ser listadas.

1.5 Git Classroom e repositório template

Todos os projetos deverão ser **obrigatoriamente** versionados através do Git Classroom. O link do *assignment* encontra-se no Moodle junto com este enunciado.

2 Relatório e Documentação

2.1 Documentação

Todo o código deve ser documentado utilizando a **documentação Doxygen**.

A mesma deve ser gerada para formato HTML e entregue a respetiva pasta "html" junto com o projeto.

2.2 Relatório

No relatório deverão constar as seguintes secções (para além de capa com identificação dos alunos e índice):

a) **ADTs Utilizados** - Descrição breve dos ADTs utilizados, qual a implementação escolhida e porquê (comparação de eficiências para o problema de aplicação).

b) **Algoritmos e complexidades** - Escolha de 5 funcionalidades do tipo B e C, onde apresentam o algoritmo implementado em pseudo-código e fazem a análise da complexidade algorítmica respetiva, levando em conta as complexidades algorítmicas das funções utilizadas dos ADTs – identificadas em a).

c) **Limitações** - Quais os comandos que apresentam problemas ou não foram implementados;

d) **Conclusões** - Análise crítica do trabalho desenvolvido.

3 Tabela de Cotações e Penalizações

A avaliação do trabalho será feita de acordo com os seguintes princípios:

- **Estruturação:** o programa deve estar estruturado de uma forma modular e procedimental;
- **Correção:** o programa deve executar as funcionalidades, tal como pedido.
- **Legibilidade e documentação:** o código deve ser escrito, formatado e comentado de acordo com o standard de programação definido para a disciplina.
- **Desempenho:** Os algoritmos implementados devem ter em conta a complexidade do mesmo, valorizando-se a implementação de algoritmos com menor complexidade. A gestão da memória deverá ser feita corretamente, garantindo que a mesma é libertada quando não está a ser utilizada. Utilização da ferramenta Valgrind, para validar a correta gestão de memória.

A nota final obtida, cuja tabela de cotações se apresenta a seguir, será ponderada de acordo com os princípios acima descritos.

Descrição	Cotação (valores)
Leitura de comandos, tratamento de situação de ficheiro inexistente/vazio, limpeza de memória e saída do programa (QUIT)	1
Importação de dados (comandos LOADAR, LOADAP e LOADF)	2
Comando SHOWALL	1,5
Comando SHOWF	1
Comando LISTAR	1
Comando LISTAP	1,25
Comando DELAYS	1,5
Comando AVERAGEDELAY	1,75
Comando SHOWAP	1,25
Comando TOPDISTN	1,5
Comando AIRPORT_S	1,5
Comando AIRLINES	1,75
Relatório e Documentação Doxygen (1,5 + 1,5)	3
TOTAL	20

NOTA: Ver **Regras** para significado das funcionalidades a amarelo.

A seguinte tabela contém penalizações a aplicar:

Descrição	Penalização (val.)
Uso de variáveis globais	até 2
Não utilização do github classroom (commits)	Pode resultar na cotação de 0 (zero) ao estudante que não seja possível comprovar trabalho efetuado
Não separação de funcionalidades em funções/módulos	até 2
Má gestão de memória dinâmica	até 2
Não utilização dos ADTs obrigatórios	Anulado

4 Instruções e Regras Finais

O não cumprimento das regras a seguir descritas implica uma penalização na nota do trabalho prático. Se ocorrer alguma situação não prevista nas regras a seguir expostas, essa ocorrência deverá ser comunicada ao respetivo docente de laboratório de ATAD.

Regras:

a) O Projeto deverá ser elaborado por **dois alunos do mesmo docente de laboratório**. Exceções têm de ser aprovadas previamente pelo RUC.

b) Só serão discutidos os projetos considerados funcionais, i.e., os que cumprirem no mínimo as funcionalidades assinaladas a amarelo na tabela de cotações; caso contrário o projeto é considerado "reprovado".

c) A nota do Projeto será atribuída individualmente a cada um dos elementos do grupo após a discussão. As discussões poderão ser orais e/ou com perguntas escritas. As orais poderão ser feitas com todos os elementos do grupo em simultâneo ou individualmente. E poderão ser feitas presencialmente ou remotamente.
   - Os *commits* no repositório individual do grupo serão tidos em conta na avaliação individual.

d) **A apresentação de relatórios ou implementações plagiadas leva à imediata atribuição de nota zero a todos os trabalhos com semelhanças, quer tenham sido o original ou a cópia**.
   - Todos os projetos serão submetidos a deteção automática e cruzada de plágio, via MOSS.

e) No rosto do relatório e nos ficheiros de implementação deverá constar o número, nome e turma dos autores e o nome do docente a que se destina.

f) O trabalho deverá ser submetido no Moodle, no link do respetivo docente de laboratórios criado para o efeito, até às **10:00 do dia 18 de julho de 2022**.

   Para tal o leader do grupo terá de submeter uma **pasta compactada em formato ZIP** contendo:
   - Ficheiro **AUTHORS.txt –** que identifica os membros do grupo (número e nome completo);
   - O relatório em formato **PDF,** e;
   - Uma pasta com o projeto VS Code (cópia do repositório, versão para submissão).

     Apenas será permitido submeter um ficheiro (o arquivo zip).

g) As datas das discussões serão publicadas após a entrega dos trabalhos.

(fim de enunciado)