Enunciado do Projecto 1 - IAED 2021/22

Data de entrega: 01 de Abril de 2022, às 19h59

LOG alterações

- 14mar22 - Publicação do enunciado.
- 18mar22 - Adicionados esclarecimentos sobre o formato de output

1. Introdução

Pretende-se a construção de um sistema de gestão de voos entre aeroportos. Para tal, o seu sistema deverá permitir a definição de aeroportos e voos, assim como a sua consulta.

A interacção com o programa deverá ocorrer através de um conjunto de linhas compostas por uma letra (comando) e um número de argumentos dependente do comando a executar. Pode assumir que todo o *input* fornecido respeitará os tipos indicados, por exemplo onde é esperado um valor inteiro decimal nunca será introduzida uma letra. Os possíveis comandos são listados na tabela seguinte e indicam as operações a executar.

Comando	Acção
q	termina o programa
a	adiciona um novo aeroporto ao sistema
l	lista os aeroportos
v	adiciona um voo ou lista todos os voos
p	lista os voos com partida de um aeroporto
c	lista os voos com chegada a um aeroporto
t	avança a data do sistema

2. Especificação do problema

O objectivo do projecto é ter um sistema de gestão de voos comerciais diários entre aeroportos. Por forma a fazer a simulação da gestão dos voos e reservas, consideramos que o passar do tempo é controlado pela aplicação através de um comando que permite definir a data actual do sistema. Quando o programa é iniciado considera-se que a data actual é 1 de Janeiro de 2022. O programa não vai ser testado com datas posteriores a 31 de Dezembro de 2023, pelo que não vão ocorrer anos bissextos.

No sistema, considera-se que uma data no input ou output refere-se sempre ao formato DD-MM-AAAA onde DD denota o dia (número entre 1 e o último dia do mês), MM o mês (número entre 1 e 12) e AAAA o ano (número positivo). As horas são também sempre no formato HH:MM onde HH denota a hora (número entre 0 e 23) e MM denota os minutos (número entre 0 e 59). Não é necessário validar se os dias, meses, horas, etc. estão dentro dos intervalos válidos.

Nas situações em que o valor é inferior a 10, então o caracter das dezenas é 0. Por exemplo, o dia 1 de Janeiro de 2022 é denotado por 01-01-2022 e as 9 horas e 5 minutos é representada por 09:05. Finalmente, os períodos de tempo como a duração dos voos segue o formato das horas.

Cada **aeroporto** é caracterizado por um identificador composto por três letras maiusculas, um país (uma *string* não vazia com um máximo de **30** carateres) e uma cidade (uma *string* não vazia com um máximo de **50** carateres)

Note-se que na *string* que descreve a cidade onde o aeroporto está localizado, podem ocorrer

carateres brancos (espaços ou tabulador `\t`). No entanto, a string que descreve o país apenas contém letras minusculas ou maíusculas. O sistema suporta no máximo **40** aeroportos, sendo que os identificadores dos aeroportos são únicos.

Cada **voo** é caracterizado por um código de voo, o aeroporto de partida e chegada, a data e hora de partida, a duração do voo e número máximo de passageiros. O código de voo é uma *string* composta por duas letras maíusculas seguido por um número entre 1 e 9999. Os números entre 1 e 9999 nunca se iniciam com um 0. Podem existir vários voos com o mesmo código, mas não podem existir dois voos com o mesmo código na mesma data de partida. Ou seja, voos com o mesmo código têm que ocorrer sempre em dias diferentes. O número máximo de passageiros pode ser qualquer inteiro entre 10 e 100 (inclusivé). A duração de um voo nunca pode ser superior a 12 horas. Podem existir no máximo **30000** voos.

3. Dados de Entrada

O programa deverá ler os dados de entrada a partir da linha de comandos do terminal.

Durante a execução do programa as instruções devem ser lidas do terminal (*standard input*) na forma de um conjunto de linhas iniciadas por um carácter, que se passa a designar por comando, seguido de um número de informações dependente do comando a executar; o comando e cada uma das informações são separados por pelo menos um carácter branco.

Os comandos disponíveis são descritos de seguida. Os caracteres < e > são utilizados apenas na descrição dos comandos para indicar os parametros. Os parametros opcionais estão indicados entre caracteres `[` e `]`. Cada comando tem sempre todos os parametros necessários à sua correcta execução. Os caracteres `...` indicam possíveis repetições de um parametro.

Cada comando indica uma determinada acção que se passa a caracterizar em termos de formato de entrada, formato de saída e erros a retornar. Se o comando gerar mais do que um erro, deverá ser indicado apenas o primeiro.

- **q** - termina o programa:
  - Formato de entrada: q
  - Formato de saída: NADA
- **a** - adiciona um novo aeroporto ao sistema:
  - Formato de entrada: `a <IDAeroporto> <país> <cidade>`
  - Formato de saída: `airport <IDAeroporto>` onde `<IDAeroporto>` é o identificador do aeroporto criado.
  - Nota: a cidade pode conter caracteres brancos.
  - Erros:
    - `invalid airport ID` no caso do identificador de aeroporto não ser uma string apenas com maíusculas.
    - `too many airports` no caso de o aeroporto, se criado, exceder o limite máximo de aeroportos permitidos pelo sistema.
    - `duplicate airport` no caso de já existir um aeroporto com o mesmo identificador.
- **l** - lista os aeroportos:
  - Formato de entrada: `l [<IDAeroporto> <IDAeroporto> ...]`
  - Formato de saída: `<IDAeroporto> <cidade> <país> #voos` por cada aeroporto, um por linha onde #voos é o número de voos com origem no aeroporto.
    - Se o comando for invocado sem argumentos, todos os aeroportos são

listados por ordem alfabética do código.
    - Se o comando for invocado com uma lista de identificadores de aeroportos, os aeroportos devem ser listados pela ordem dos respetivos identificadores no comando.
  - Erros:
    - `<IDAeroporto>: no such airport ID` no caso de não existir o aeroporto indicado.

- **v** - adiciona um voo ou lista todos os voos:
  - Formato de entrada: `v [<códigoVoo> <IDAeroportoPartida> <IDAeroportoChegada> <dataPartida> <horaPartida> <duração> <capacidade>]`
  - Formato de saída: `<códigoVoo> <IDAeroportoPartida> <IDAeroportoChegada> <dataPartida> <horaPartida>` por cada voo adicionado, pela ordem de criação ou nada, se for criado um novo voo.
  - Erros:
    - `invalid flight code` no caso do código do voo não ser uma string com duas maíusculas seguida por digitos (entre 1 a 4 digitos).
    - `flight already exists` no caso de já existir um voo com o mesmo código para o mesmo dia.
    - `<IDAeroporto>: no such airport ID` no caso de não existir o aeroporto (partida ou chegada) indicado.
    - `too many flights` no caso de o novo voo, a ser criado, exceda o limite de voos.
    - `invalid date` no caso de ser uma data no passado ou mais de um ano no futuro.
    - `invalid duration` no caso de ser um voo com duração superior a 12 horas.
    - `invalid capacity` no caso da capacidade não for um inteiro entre 10 e 100 (inclusivé).

- **p** - lista os voos com partida de um aeroporto:
  - Formato de entrada: `p <IDAeroporto>`
  - Formato de saída: `<códigoVoo> <IDAeroportoChegada> <dataPartida> <horaPartida>` por cada voo, um por linha ordenados por data e hora de partida. Os voos são ordenados da data e hora mais antigas para a mais recente. Num aeroporto não há dois voos com a mesma hora de partida no mesmo dia.
  - Erros:
    - `<IDAeroporto>: no such airport ID` no caso de não existir o aeroporto indicado.

- **c** - lista os voos com chegada a um aeroporto:
  - Formato de entrada: `c <IDAeroporto>`
  - Formato de saída: `<códigoVoo> <IDAeroportoPartida> <dataChegada> <horaChegada>` por cada voo, um por linha ordenados por data e hora de chegada. Os voos são ordenados da data e hora mais antigas para a mais recente. Num aeroporto não há dois voos com a mesma hora de chegada no mesmo dia.
  - Erros:
    - `<IDAeroporto>: no such airport ID` no caso de não existir o aeroporto indicado.

- **t** - avança a data do sistema:

- Formato de entrada: `t <data>`
- Formato de saída: `<data>` onde `<data>`é novo valor da data atual.
- Erros:
    - `invalid date` no caso de ser uma data no passado ou mais de um ano no futuro.

Só poderá usar as funções de biblioteca definidas em `stdio.h`, `stdlib.h`, `ctype.h` e `string.h`

Nota importante: não é permitida a utilização da função `qsort` nativa do C e este *nome* não deve aparecer no vosso código.

O compilador a utilizar é o `gcc` com as seguintes opções de compilação: `-Wall -Wextra -Werror -ansi -pedantic`. Para compilar o programa deve executar o seguinte comando:

```
$ gcc -Wall -Wextra -Werror -ansi -pedantic -o proj1 *.c
```

O programa deverá escrever no *standard output* as respostas aos comandos apresentados no *standard input*. As respostas são igualmente linhas de texto formatadas conforme definido anteriormente neste enunciado. Tenha em atenção ao número de espaços entre elementos do seu output, assim como a ausência de espaços no final de cada linha. Procure respeitar escrupulosamente as indicações dadas.

Ver os exemplos de input e respectivos output na pasta `tests/`.

O programa deve ser executado da forma seguinte:

```
$ ./proj1 < test.in > test.myout
```

Posteriormente poderá comparar o seu output (`*.myout`) com o output previsto (`*.out`) usando o comando `diff`,

```
$ diff test.out test.myout
```

Para testar o seu programa poderá executar os passos indicados acima ou usar o comando `make` na pasta `tests/`.

5. Entrega do Projecto

A entrega do projecto deverá respeitar o procedimento seguinte:

- Na página da disciplina aceda ao sistema para entrega de projectos. O sistema será activado uma semana antes da data limite de entrega. As instruções acerca da forma de acesso ao sistema serão oportunamente fornecidas.
- Efectue o upload de um ficheiro de arquivo com extensão `.zip` que inclua todos os ficheiros fonte que constituem o programa.
- Se o seu código tiver apenas um ficheiro o zip conterá apenas esse ficheiro.
- Se o seu código estiver estruturado em vários ficheiros (`.c` e `.h`) não se esqueça de os juntar também ao pacote.
- Para criar um ficheiro arquivo com a extensão `.zip` deve executar o seguinte comando

na directoria onde se encontram os ficheiros com extensão `.c` e `.h` (se for o caso), criados durante o desenvolvimento do projecto:

```
$ zip proj1.zip *.c *.h
```

- Como resultado do processo de upload será informado se a resolução entregue apresenta a resposta esperada num conjunto de casos de teste.
- O sistema não permite submissões com menos de 10 minutos de intervalo para o mesmo aluno. Tenha especial atenção a este facto na altura da submissão final. Exemplos de casos de teste serão oportunamente fornecidos.
- Data limite de entrega do projecto: **01 de Abril de 2022, às 19h59m**. Até à data limite poderá efectuar o número de submissões que desejar, sendo utilizada para efeitos de avaliação a última submissão efectuada. Entre cada submissão haverá um período de 5 minutos entre os quais não poderá fazer nova submissão. Deverá portanto verificar cuidadosamente que a última submissão corresponde à versão do projecto que pretende que seja avaliada. Não existirão excepções a esta regra.

6. Avaliação do Projecto

Na avaliação do projecto serão consideradas as seguintes componentes:

1. A primeira componente avalia o desempenho da funcionalidade do programa realizado. Esta componente é avaliada entre 0 e 16 valores.
2. A segunda componente avalia a qualidade do código entregue, nomeadamente os seguintes aspectos: comentários, indentação, estruturação, modularidade, abstracção, entre outros. Esta componente poderá variar entre -4 valores e +4 valores relativamente à classificação calculada no item anterior e será atribuída posteriormente. Algumas *guidelines* sobre este tópico podem ser encontradas aqui.
3. A classificação da primeira componente da avaliação do projecto é obtida através da execução automática de um conjunto de testes num computador com o sistema operativo GNU/Linux. Torna-se portanto essencial que o código compile correctamente e que respeite o formato de entrada e saída dos dados descrito anteriormente. Projectos que não obedeçam ao formato indicado no enunciado serão penalizados na avaliação automática, podendo, no limite, ter 0 (zero) valores se falharem todos os testes. Os testes considerados para efeitos de avaliação poderão incluir (ou não) os disponibilizados na página da disciplina, além de um conjunto de testes adicionais. A execução de cada programa em cada teste é limitada na quantidade de memória que pode utilizar, e no tempo total disponível para execução, sendo o tempo limite distinto para cada teste.
4. Note-se que o facto de um projecto passar com sucesso o conjunto de testes disponibilizado na página da disciplina não implica que esse projecto esteja totalmente correcto. Apenas indica que passou alguns testes com sucesso, mas este conjunto de testes não é exaustivo. É da responsabilidade dos alunos garantir que o código produzido está correcto.
5. Em caso algum será disponibilizado qualquer tipo de informação sobre os casos de teste utilizados pelo sistema de avaliação automática. A totalidade dos ficheiros de teste usados na avaliação do projecto serão disponibilizados na página da disciplina após a data de entrega.

7. Dicas para Desenvolvimento do Projecto

Abaixo podem encontrar algumas dicas simples que facilitam o despiste de erros comuns no desenvolvimento do projecto. Sugerimos que **desenvolvam os vossos projectos de forma incremental e que testem as vossas soluções localmente antes de as submeterem no Mooshak**. Também é desejável que tenham o projeto funcional à data da abertura do Mooshak por forma a poderem tirar completo partido deste.

Sugerimos que sigam os seguintes passos:

1. Desenvolva e corrija o código de forma incremental garantindo que compila sem erros nem *warnings*. Não acumule uma série de erros pois o *debug* é tanto mais complexo quanto a dimensão da base de código a analisar.
2. Garanta que está a ler o *input* e a escrever o *output* correctamente, em particular garanta que as *strings* não ficam com espaços extra, \n no final, que a formatação está correcta e de acordo com o que está no enunciado, *etc*.
3. Procure desenvolver os comandos pela ordem apresentada.
4. Teste isoladamente cada comando e verifique que funciona correctamente.