Enunciado do Projecto 2 - IAED 2021/2022

Data de entrega: 20 de Abril de 2022, às 19h59m

LOG alterações

- 4-abr-22 - Publicação do enunciado.
- 7-abr-22 - Clarificação do código de reserva e do comando **e**.

1. Introdução

O objectivo deste projeto é o desenvolvimento, em linguagem C, de funcionalidades adicionais às criadas no primeiro projeto. 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 comandos do primeiro projeto são listados na tabela seguinte e mantêm as operações a executar e obedecem aos limites impostos, exceto o limite superior de passageiros por voo que é eliminado.

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

Além dos comandos do primeiro projeto, são adicionados os comandos listados na tabela seguinte, bem como as operações a executar.

Comando	Acção
r	adiciona uma reserva ou lista as reservas de um voo
e	elimina voos ou reserva

2. Especificação do problema

Não existem limites no número de reservas, nem na dimensão dos códigos das reservas, logo deve procurar utilizar a memória estritamente necessária. Para facilitar a introdução dos dados, pode assumir que cada instrução não excede 65535 carateres. Se a memória se esgotar, o programa deve terminar de forma controlada, imprimindo a mensagem `No memory.` Antes de terminar, o programa deve libertar toda a memória reservada.

3. Dados de Entrada

Durante a execução do programa as instruções devem ser lidas do standard input na forma de um conjunto de linhas iniciadas por uma palavra, que se passa a designar por *comando*, seguido de um número de informações dependente do comando a executar. Os comandos e os argumentos são separados por espaços ou tabuladores.

Cada comando indica uma determinada ação que se passa a caracterizar em termos de formato de entrada, formato de saída, e erros. No caso de múltiplos erros para o mesmo comando

deverá retornar apenas o primeiro desses erros. Os comandos adicionais são:

- **r** - adiciona uma reserva ou lista as reserva de um voo:
    - Formato de entrada: `r <códigoVoo> <data> [<códigoReserva> <númeroPassageiros>]`
    - Formato de saída: `<códigoReserva> <númeroPassageiros>` para cada reserva no voo com o código `<códigoVoo>` na data `<data>`. Uma reserva por linha por ordem lexicográfica do código de reserva.
    - Erros:
        - `invalid reservation code` no caso do código da reserva não ser uma string composta apenas por maíusculas e digitos ou se for composto por menos de 10 caracteres.
        - `<códigoVoo>: flight does not exist` no caso de não existir um voo com o código na data indicada.
        - `<códigoReserva>: flight reservation already used` no caso de já existir uma reserva com o `<códigoReserva>` indicado.
        - `too many reservations` no caso da reserva, a ser criada, exceda a capacidade do voo.
        - `invalid date` no caso de ser uma data no passado ou mais de um ano no futuro.
        - `invalid passenger number` no caso do número de passageiros não seja um inteiro superior a 0.
- **e** - elimina voos ou reserva:
    - Formato de entrada: `e <código>`
    - Formato de saída: Apaga todos os voos ou a reserva com o `<código>` indicado. Caso se trate de um código de voo também devem ser eliminadas todas as reservas associadas aos voos removidos.
    - Erros:
        - `not found` no caso de não existir o `<código>`.

4. Dados de Saída

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 o número de espaços entre elementos do seu output, assim como os espaços no final de cada linha. Procure respeitar escrupulosamente as indicações dadas.

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 proj2 *.c
```

Este comando deve ter como resultado a geração do ficheiro executável `proj2`, caso não haja erros de compilação. **A execução deste comando não deverá escrever qualquer resultado no terminal. Caso a execução deste comando escreva algum resultado no terminal, considera-se que o programa não compilou com sucesso**. Por exemplo, durante a compilação do programa, o compilador não deverá escrever mensagens de aviso (warnings).

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

5. Execução do Programa

O programa deve ser executado da forma seguinte:

```
$ ./proj2 < 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/`. na pasta `tests/`. Para executar todos os testes com o *valgrind* poderá executar `make valgrind` na pasta `tests/`.

6. 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. Instruções acerca da forma de acesso ao sistema serão oportunamente fornecidas.
- Efectue o upload de um ficheiro 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 proj2.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.
- Data limite de entrega do projecto: **20 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. Deverá portanto verificar cuidadosamente que a última submissão corresponde à versão do projecto que pretende que seja avaliada. Não existirão excepções a esta regra.

7. Avaliação do Projecto

Na avaliação do projecto serão consideradas as seguintes componentes:

1. A primeira componente será feita automaticamente e 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, alocação dinâmica de memória, estruturação, modularidade e divisão em ficheiros, abstracção de dados, 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. Na segunda componente serão utilizadas as ferramentas *lizard*, *valgrind*, e a opção *fsanitize* por forma a detectar a complexidade de código, fugas de memória ("memory leaks") ou outras incorrecções no código, que serão penalizadas. Aconselha-se que utilizem estas ferramentas para fazer debugging do código e corrigir eventuais incorrecções, antes da submissão do projecto. Algumas dicas para degugging podem ser encontradas aqui.

4. 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 descritos 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.

5. 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.