Enunciado do Projecto 2 - IAED 2022/2023

Data de entrega: 14 de abril de 2023, às 19h59m

LOG alterações

- 27-mar-23 - Publicação do enunciado.

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.

Comando	Acção
q	termina o programa
c	adiciona e lista as carreiras
p	adiciona e lista as paragens
l	adiciona ligações entre paragens numa carreira
i	lista os nós de interligação

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

Comando	Acção
r	remove uma carreira
e	elimina uma paragem
a	apaga todos os dados

2. Especificação do problema

Não existem limites no número de paragens ou ligações, nem na dimensão dos nomes das carreiras ou paragens, logo deve procurar utilizar a memória estritamente necessária. Não podem existir variáveis globais. 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** - remove uma carreira:
    - Formato de entrada: `r <nome-de-carreira>`
    - Formato de saída: Nada. Apaga a carreira indicada e todas as ligações por ela usadas.
    - Erros:
        - `<nome-de-carreira>: no such line.` no caso de não existir uma carreira com o nome indicado.
- **e** - elimina uma paragem:
    - Formato de entrada: `e <nome-de-paragem>`
    - Formato de saída: Apaga a paragem indicada, passando todas as carreiras que nela paravam a saltar a paragem mas mantendo o restante percurso; o custo e a duração resulta da soma dos dois percursos colapsados pela eliminação da paragem; se a paragem for um extremo do percurso de uma carreira, o percurso fica encurtado.
    - Erros:
        - `<nome-de-paragem>: no such stop.` no caso de não existir uma paragem com o nome indicado.
- **a** - apaga todos os dados:
    - Formato de entrada: `a`
    - Formato de saída: Liberta toda a memória reservada desde o innício da execução (ver *valgrind*).

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 instrução `goto`, da declaração `extern`, nem da função `qsort` nativa do C e nenhum destes *nomes* deve aparecer no vosso código.

Exemplos de utilização dos comandos

Considere que estão definidas as carreiras da figura do enunciado do 1º projeto.

Comando `r`

```
r middle
```

O comando `r` seguido do nome de uma carreira remove do sistema a carreira indicada e todas as suas ligações.

Comando `e`

```
e D2
```

O comando `e` seguido de uma paragem existente no sistema remove a paragem do sistema. As ligações das carreiras com a paragem a eliminar têm que ser actualizadas.

Comando `a`

```
a
```

O comando `a` remove todas as paragens, carreiras e respectivas ligações do sistema.

4. Compilação e teste

O compilador a utilizar é o `gcc` com as seguintes opções de compilação: `-O3 -Wall -Wextra -Werror -ansi -pedantic`. Para compilar o programa deve executar o seguinte comando:

```
$ gcc -O3 -Wall -Wextra -Werror -ansi -pedantic -o proj2 *.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 `public-tests/`.

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 `public-tests/`.

5. Entrega do Projeto

Será criado um repositório `git` para cada aluno desenvolver e submeter o projeto. Este repositório será criado no GitLab da RNL e será activado quando da publicação deste enunciado.

Na sua submissão do projeto deve considerar os seguinte pontos:

- Considera-se que os seus ficheiros de desenvolvimento do projeto (`.c` e `.h`) estão na raiz do repositório e não numa directoria. *Qualquer ficheiro fora da raíz não será considerado como pertencendo ao seu projeto*.
- A última versão que estiver no repositório da RNL será considerada a submissão para avaliação do projeto. Qualquer versão anterior ou que não esteja no repositório não será considerada na avaliação.

- Antes de fazer qualquer submissão para o repositório da RNL, não se esqueça que deve sempre fazer `pull` para sincronizar o seu repositório local.
- Quando actualizar os ficheiros `.c` e `.h` no seu repositório na RNL, esta versão será avaliada e será informado se essa versão apresenta a resposta esperada num conjunto de casos de teste. Tal como no repositório dos laboratórios, o resultado da avaliação automática será colocado no repositório do aluno.
- Para que o sistema de avaliação seja executado, tem que esperar pelo menos 10 minutos. Sempre que fizer uma actualização no repositório, começa um novo período de espera de 10 minutos. Exemplos de casos de teste serão oportunamente fornecidos.
- Data limite de entrega do projeto: **14 de abril de 2023, às 19h59m**. Até à data limite poderá efectuar o número de submissões que desejar, sendo utilizada para efeitos de avaliação a última versão. Deverá portanto verificar cuidadosamente que a última versão no repositório GitLab da RNL corresponde à versão do projeto que pretende que seja avaliada. Não existirão excepções a esta regra.

6. Avaliação do Projeto

Na avaliação do projeto 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 em guidelines.md.
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 em debugging.md.

- A classificação da primeira componente da avaliação do projeto 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. Projetos 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.
- Note-se que o facto de um projeto passar com sucesso o conjunto de testes disponibilizado na página da disciplina não implica que esse projeto 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.
- 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 projeto serão disponibilizados na página da disciplina após a data de entrega.