Enunciado do Projecto 2 - IAED 2018/19

Data de entrega: 17 de Maio de 2019, às 17h59m

LOG alterações

1. Introdução

O objectivo deste projeto é o desenvolvimento, em linguagem C, de um sistema de gestão de contactos. 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. Os possíveis comandos são listados na tabela seguinte e indicam as operações a executar.

Comando	Acção
a	Adiciona um novo contacto.
l	Lista todos os contactos pela ordem em que foram introduzidos.
p	Procura um contacto.
r	Apaga um contacto.
e	Altera o endereço do email de um contacto.
c	Conta o número de ocorrências de um domínio de email.
x	Termina o programa.

2. Especificação do problema

O objectivo do projecto é ter um sistema de gestão de contactos que permita associar um endereço de email e um número de telefone a um nome.

Cada contacto tem:

- Um *nome*, uma string não vazia, sem espaços, com no máximo de 1023 caracteres (`[0-9a-zA-Z-_]+`).
- Um *email*, uma string não vazia, sem espaços, com no máximo de 511 caracteres, com uma única ocorrência do carácter `@`; em particular, o email pode ser dividido entre a parte *local* e *domínio* separados pelo carácter `@` (`[0-9a-zA-Z-_.]+@[0-9a-zA-Z-_.]+`).
- Um *número de telefone*, uma string não vazia, sem espaços, com no máximo de 63 caracteres, (`[0-9\-]+`).

Nota: Como os nomes/endereços/telefones longos são raros, as estruturas de dados utilizadas só deverão alocar a memória suficiente para as strings dadas.

3. Dados de Entrada

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

Os comandos disponíveis são descritos de seguida. Cada comando indica uma determinada acção que se passa a caracterizar em termos de formato de entrada, formato de saída, e erros a retornar:

- **a** - Adiciona um novo contacto.
  - Formato de entrada: `a nome email telefone`
  - Formato de saída: NADA (excepto erro).
  - Erros:
    - `Nome existente.` no caso de o nome já existir no sistema.
- **l** - Lista os contactos introduzidos.
  - Formato de entrada: `l`
  - Formato de saída: Uma linha por contacto no formato abaixo

```
<nome> <email> <telefone>
```

Os contactos deverão ser listados pela ordem em que foram introduzidos.

    - Erros: Não aplicável.

- **p** - Procura um contacto dado um nome.
    - Formato de entrada: `p nome`
    - Formato de saída: Igual ao formato de saída do comando `l`.
    - Erros:
        - `Nome inexistente.` no caso de o nome não existir.
- **r** - Apaga um contacto dado um nome.
    - Formato de entrada: `r nome`
    - Formato de saída: NADA (excepto erro).
    - Erros:
        - `Nome inexistente.` no caso de o nome não existir.
- **e** - Altera o endereço de email de um contacto dado o nome.
    - Formato de entrada: `e nome novo-email`
    - Formato de saída: NADA (excepto erro).
    - Erros:
        - `Nome inexistente.` no caso de o nome não existir.
- **c** - Conta o número de ocorrências de um domínio de e-mail dado.
    - Formato de entrada: `c dominio`
    - Formato de saída: `dominio:<número-de-ocorrências>`
    - Erros: Não aplicável.
    - O domínio é uma string sem espaços com um máximo de 511 caracteres (veja a descrição de contacto).

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.

5. Exemplos (Input/Output)

Ver os exemplos de input e respectivos outputs na pasta `testes_enunciado/`.

6. Compilação do Programa

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

```
$ gcc -Wall -Wextra -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).

7. 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
```

7.1. Testes Auxiliares

Para testar o seu programa poderá executar os passos indicados acima ou usar os scripts `test.sh` e `test-vg.sh` distribuídos nesta pasta. Para isso **deverão ter a vossa solução nesta pasta** e

1. Para executarem todos os testes de uma `pasta` deverão executar
2. `$ ./test.sh <pasta> <vossos_ficheiros_c>`
3. Para executarem todos os testes de uma `pasta` com o *valgrind* deverão executar
4. `$ ./test-vg.sh <pasta> <vossos_ficheiros_c>`

Estes scripts compilam os ficheiros indicados e comparam o resultado obtido com o resultado esperado.

8. 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
  ```
  Se só tiver um único ficheiro (e.g., `proj2.c`), bastará escrever:
  ```
  $ zip proj2.zip proj2.c
  ```

- 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 grupo. 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: **17 de Maio de 2019, às 17h59m**. 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.

9. Avaliação do Projecto

9.1. Componentes da Avaliação

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, alocação dinâmica, 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. Nesta componente será também utilizado a
  ```

ferramenta *valgrind* de forma a detectar fugas de memória ("memory leaks") ou outras incorrecções no código, que serão penalizadas. Aconselha-se por isso que os alunos utilizem esta ferramenta para fazer debugging do código e corrigir eventuais incorrecções, antes da submissão do projecto. Para utilizar o *valgrind*, comece por compilar o seu código com o `gcc` mas com a flag adicional `-g`:

3.
```
$ gcc -g -Wall –ansi –pedantic -o proj2 *.c
```

Esta flag adiciona informação para debugging que será posteriormente utilizada pelo *valgrind*. Depois disso, basta escrever:

```
$ valgrind --tool=memcheck --leak-check=yes ./proj2 < test01.in
```

Neste exemplo, estaria a usar o `test01.in` como input. Deverá estar particularmente atento/a a mensagens como:

```
Invalid read e Invalid write
```

que indicam está a tentar ler ou escrever fora da área de memória reservada por si. O *valgrind* também detecta a utilização de variáveis não inicializadas dentro expressões condicionais. Nesse caso receberá a mensagem:

```
Conditional jump or move depends on uninitialised value(s).
```

Por fim, o *valgrind* oferece no final a preciosa informação sobre a quantidade de memória alocada e libertada no heap, indicando quando essas duas quantidades não são iguais. Nesse caso o programa tem *memory leaks*. Aqui fica um exemplo de output do *valgrind* quando tudo corre bem:

```
HEAP SUMMARY:
in use at exit: 0 bytes in 0 blocks
total heap usage: 1,024,038 allocs, 1,024,038 frees, 28,682,096 bytes
allocated
All heap blocks were freed -- no leaks
```

9.2. Atribuição Automática da Classificação

- 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, até um máximo de 8MB, e no tempo total disponível para execução, sendo o tempo limite distinto para cada teste.

- 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.