INSTITUTO SUPERIOR TÉCNICO

Introdução aos Algoritmos e Estruturas de Dados

2016/2017 - 2º semestre

Enunciado do 1º Projecto (v1.2)

Data de entrega: 29 de Abril de 2017 (23h59m)

1. Introdução

Nos nossos dias, a informação é um bem precioso e a monitorização, gestão, e análise desta uma área em franca expansão.

O objectivo deste projeto é o desenvolvimento, em linguagem C, de um programa para gestão e monitorização de um fórum de mensagens. Mais especificamente, a interacção com o programa deverá ocorrer através de um conjunto de linhas compostas por uma letra (comando) e eventualmente um ou dois argumentos. Os possíveis comandos são listados na Tabela seguinte e indicam as operações a executar.

Comando	Descrição
`A`	Adiciona uma mensagem.
`L`	Lista todas as mensagens.
`U`	Lista todas as mensagens introduzidas por um dado utilizador.
`O`	Lista as mensagens mais longas.
`T`	Escreve o utilizador mais activo na escrita de mensagens.
`S`	Lista todas as mensagens por ordem alfabética.
`C`	Escreve o número de ocorrências de uma palavra.
`X`	Termina o programa

2. Especificação do programa

Uma mensagem num fórum é composta por um *identificador de utilizador* (um inteiro) e uma *frase* (uma sequência de caracteres). Assuma que o fórum nunca terá mais de 1000 participantes, 10.000 mensagens, e que cada frase tem no máximo 140 caracteres. Assuma também que o identificador de um participante nunca será superior a 1000.

O sistema de gestão de mensagens permite adicionar e listar mensagens, bem como procurar as frases mais ~~comuns~~ longas e utilizadores mais activos, ou contar o número de ocorrências de uma dada palavra.

3. Dados de Entrada

O programa deverá ler os dados de entrada a partir do standard input, na forma de um conjunto de linhas iniciadas por um carácter, que se passa a designar por *comando*, seguido de um número variável de informações; o comando e cada uma das informações são separados por um espaço.

Os comandos disponíveis são descritos de seguida e correspondem às letras `A, L, U, O, T, C, S, X`. Cada comando indica uma determinada acção que se passa a caracterizar em termos de objectivo, número de informações por linha (n.i.l.), sintaxe e output[1]:

`A`: adiciona uma nova mensagem (n.i.l. = 2)[2]

```
A user frase
```

- `user`: número inteiro entre 0 e 999;
- `frase`: sequência de caracteres.

Output: Não tem qualquer output.

Adiciona a *frase* do utilizador *user* ao fórum de mensagens.
Sugestão: Uma frase poderá conter várias palavras. Para a sua leitura, poderá utilizar a função `LeLinha` discutida nas aulas ou, se preferir, a função `fgets`[3].

`L`: lista todas as mensagens do fórum (n.i.l. = 0)

```
L
```

Output: Lista todas as mensagens do fórum pela ordem em que foram introduzidas no sistema. Deverá ter uma linha inicial

```
*TOTAL MESSAGES:<total_mensagens>
```

seguida de uma mensagem por cada linha na forma

```
<user>:<frase>
```

(substituir "`<user>`" e "`<frase>`" pelos respectivos elementos)

`U`: lista todas as mensagens submetidas pelo utilizador (n.i.l. = 1)

```
U user
```

- `user` :número inteiro.

Output: Lista todas as mensagens do `user` no fórum pela ordem em que foram introduzidas no sistema. Deverá ter uma linha inicial

```
*MESSAGES FROM USER:<user>
```

[1] Se encontrar dificuldades, veja p.f. o exemplo de linha de comandos dado nas aulas teóricas.
[2] Poderá criar um vector de estruturas "mensagem" onde guarda o id do utilizador e a própria mensagem.
[3] Ver http://www.cplusplus.com/reference/cstdio/fgets/. Note que o fgets guarda o `\n` final na string.

Seguida de uma frase por linha na forma

```
<frase>
```

O: frase mais longa (n.i.l. = 0)

```
O
```

Output: Lista a frase mais longa do fórum[4]. Se existir mais do que uma entrada, deverá listar todas as entradas com igual comprimento, pela ordem que foram introduzidas no sistema, uma por cada linha.

```
*LONGEST SENTENCE:<user>:<frase>
```

Observações: Notem que a mesma frase pode ser submetida por vários utilizadores, ou várias vezes pelo mesmo. Nesse caso, deverão ser impressas todas as ocorrências, mesmo que iguais.

T: utilizador mais activo (n.i.l. = 0)[5]

```
T
```

Output: Lista o `user` mais activo, i.e., aquele que enviou mais mensagens. Se houver mais do que um, deverá listá-los por ordem crescente do seu identificador, e na forma

```
*MOST ACTIVE USER:<user>:<N_frases>
```

C: conta ocorrências de palavra (n.i.l. = 1)

```
C palavra
```

- `palavra`: sequência de caracteres sem espaços (uma única palavra).

Output: Imprime no ecran o número de ocorrências da `palavra` em todas as frases do fórum. Por palavra entende-se qualquer sequência de caracteres limitadas por caracteres brancos (e.g., `';'`, `','` , `' '` , `'\t'`, `'\n'`,`'.'`) ou pelo início ou fim da frase (`'\0'`). Deverá escrever

```
*WORD <palavra>:<N_ocorrencias>
```

Observações: a palavra `da` não ocorre em `dar` nem em `dado`.

S: lista ordenada (n.i.l. = 0)

[4] Sugestão: Poderá recorrer à função `strlen` (disponível em `string.h`) ou adaptar o retorno da função `LeLinha` discutida nas aulas teóricas.

[5] Sugestão: Poderá tirar partido do facto de os ids estarem limitados entre 0 e 999 por forma a registar o número de mensagens associadas a cada user.

`S`

Output: Lista todas as mensagens enviadas, ordenadas por ordem alfabética[6]. Caso existam 2 mensagens com a mesma frase, deverá ordenar pelo identificador do utilizador, por ordem crescente. Se mesmo assim forem iguais, deverá ser respeitada a ordem de introdução no sistema. Deverá ter uma linha inicial

`*SORTED MESSAGES:<total_mensagens>`

seguida de uma mensagem por cada linha na forma

`<user>:<frase>`

`X`: sair do programa (n.i.l. = 0)
`X`

Output: `N_mensagens`

Termina o programa imprimindo ~~retornando o número total de utilizadores e~~ o número total de mensagens no sistema.

4. Dados de Saída

O programa deverá escrever no standard output as respostas a certos 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)

Exemplo 1

Dados de Entrada:

```
A 123 Mensagem de teste
L
A 123 Outra mensagem de teste
A 567 Ainda outra mensagem de teste
L
X
```

Dados de Saída:

```
*TOTAL MESSAGES:1
123:Mensagem de teste
*TOTAL MESSAGES:3
123:Mensagem de teste
123:Outra mensagem de teste
567:Ainda outra mensagem de teste
3
```

Exemplo 2

[6] Poderá utilizar a função `int strcmp(char s1[], char s2[])` disponível em `string.h`. Esta função retorna 0 se duas strings são iguais, e retorna um valor positivo se `s1` é lexicograficamente (i.e., alfabeticamente) superior a `s2` e um número negativo se `s2` é superior a `s1`.

Dados de Entrada:

```
A 123 Mensagem de teste
A 123 Outra mensagem de teste
A 567 Ainda outra mensagem de teste
U 123
U 890
O
A 890 A roda do autocarro roda roda
O
X
```

Dados de Saída:

```
*MESSAGES FROM USER:123
Mensagem de teste
Outra mensagem de teste
*MESSAGES FROM USER:890
*LONGEST SENTENCE:567:Ainda outra mensagem de
teste
*LONGEST SENTENCE:567:Ainda outra mensagem de
teste
*LONGEST SENTENCE:890:A roda do autocarro roda
roda
4
```

Exemplo 3

Dados de Entrada:

```
A 123 Mensagem de teste
A 123 Outra mensagem de teste
A 567 Ainda outra mensagem de teste
A 890 A roda do autocarro roda roda
T
A 890 A roda do autocarro roda roda roda
T
S
A 890 A roda do autocarro roda
A 123 A roda do autocarro roda
S
X
```

Dados de Saída:

```
*MOST ACTIVE USER:123:2
*MOST ACTIVE USER:123:2
*MOST ACTIVE USER:890:2
*SORTED MESSAGES:5
890:A roda do autocarro roda roda
890:A roda do autocarro roda roda roda
567:Ainda outra mensagem de teste
123:Mensagem de teste
123:Outra mensagem de teste
*SORTED MESSAGES:7
123:A roda do autocarro roda
890:A roda do autocarro roda
890:A roda do autocarro roda roda
890:A roda do autocarro roda roda roda
567:Ainda outra mensagem de teste
123:Mensagem de teste
123:Outra mensagem de teste
7
```

Exemplo 4

Dados de Entrada:

```
A 123 Mensagem de teste
A 123 Outra mensagem de teste
A 567 Ainda outra mensagem de teste
A 890 A roda do autocarro roda roda
C roda
C carro
C auto
A 890 A roda do autocarro roda roda roda
A 890 A roda do autocarro roda
A 123 A roda do autocarro roda
C roda
C carro
C auto
X
```

Dados de Saída:

```
*WORD roda:3
*WORD carro:0
*WORD auto:0
*WORD roda:11
*WORD carro:0
*WORD auto:0
7
```

6. Compilação do Programa

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

```
$ gcc -Wall -o proj1 *.c
```

o qual deve ter como resultado a geração do ficheiro executável `proj1`, 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 deve escrever mensagens de aviso (warnings).

7. Execução do Programa

O programa deve ser executado da forma seguinte:

```
$ ./proj1 < test01.in > test01.myout
```

Posteriormente poderá comparar o seu output com o output previsto usando o comando **`diff`**

```
$ diff test01.out test01.myout
```

7.1. Testes Auxiliares

Para testar o seu programa poderá executar os passos indicados acima ou usar os scripts **`run.sh`** e **`runall.sh`** distribuídos no ficheiro **`Exemplos.zip`**.

Se quiserem executar apenas o **`teste01.in`** deverão executar

```
$ ./run.sh <vosso_ficheiro_c> teste01.in
```

Para executarem todos os testes deverão executar

```
$ ./runall.sh <vosso_ficheiro_c>
```

Estes scripts compilam o ficheiro indicado e comparam o resultado obtido com o resultado esperado. Se apenas indicar o tempo de execução é porque o comando **`diff`** não encontrou nenhuma diferença. Caso indique mais informação, então é porque o resultado obtido e o resultado esperado diferem. Para obter a informação detalhada das diferenças poderá remover a opção **`-q`** da linha 10 do ficheiro **`run.sh`**.

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

  Se só tiver um único ficheiro (e.g., `proj1.c`), bastará escrever:
  ```
  $ zip proj1.zip proj1.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: **29 de Abril de 2017 (23h59m)**. 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

a. 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, 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 na discussão final do projecto.

3. Durante a discussão final do projecto será averiguada a participação de cada elemento do grupo na realização do projecto, bem como a sua compreensão do trabalho realizado. A respectiva classificação será ponderada em conformidade, isto é, elementos do mesmo grupo podem ter classificações diferentes. Elementos do grupo que se verifique não terem participado na realização do respectivo projecto terão a classificação de 0 (zero) valores.

b. 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 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, até um máximo de 64 Mb, 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.
- Em caso algum será disponibilizado qualquer tipo de informação sobre os casos de teste utilizados pelo sistema de avaliação automática. A totalidade de ficheiros de teste usados na avaliação do projecto serão disponibilizados na página da disciplina após a data de entrega.

Versão 1.2: 21Apr2017: Acrescenta o separador '.' ao comando C. Corrige a palavra "longas" no segundo parágrafo da secção 2.

Versão 1.1: 19Apr2017: Corrige o comando `X` removendo do texto a necessidade de devolver o número de utilizadores (cf. o Output indicado e os Exemplos).

Adiciona secção 7.1 explicando a utilização dos scripts **`run.sh`** e **`runall.sh`**.