INSTITUTO SUPERIOR TÉCNICO

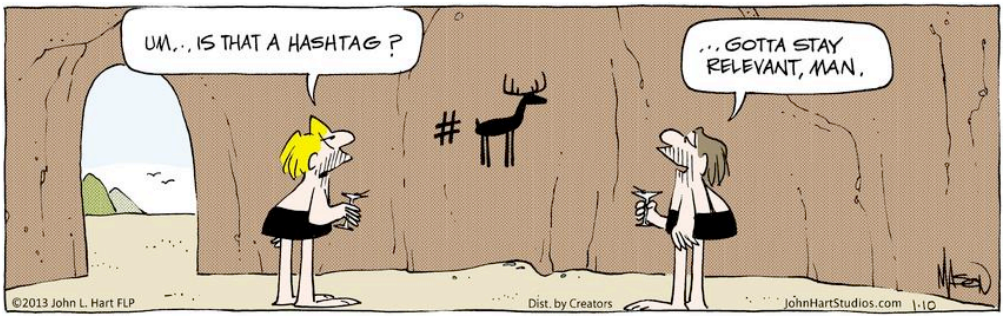
Introdução aos Algoritmos e Estruturas de Dados
2015/2016

Enunciado do 2º Projecto

Data de entrega: 14 de Maio de 2016 (23h59)

1 Introdução

As *hashtags* são hoje um popular termo para denominar palavras-chave ou referências a uma informação, tópico ou discussão que se deseja indexar de forma explícita. Popularizadas pelo *Twitter*, e posteriormente pelo *Facebook*, *Google+* e *Instagram*, entre outros, as *hashtags* são compostas pela palavra-chave do assunto antecedida por um cardinal (#).



O objectivo deste projeto é o desenvolvimento, em linguagem C, de um programa para contagem do número de ocorrências de cada *hashtag* num conjunto de mensagens dadas. Mais especificamente, a interacção com o programa deverá ocorrer através de um conjunto de linhas compostas por uma letra (comando) e um eventual argumento. Os possíveis comandos são listados na Tabela seguinte e indicam as operações a executar.

Comando	Descrição
`a <mensagem>`	Processa uma `mensagem` incrementando para cada *hashtag* encontrada o contador respectivo.
`s`	Mostra o número de *hashtags* distintas e o número total ocorrências de todas as *hashtags*.
`m`	Mostra a *hashtag* mais popular.
`l`	Lista todas as *hashtags* por ordem decrescente do número de ocorrências. Em caso de igualdade, as *hashtags* deverão ser ordenadas alfabeticamente.
`x`	Termina o programa

Uma vez que as funcionalidades pretendidas são conceptualmente muito simples, pretende-se que os alunos dediquem particular esforço à implementação de soluções computacionalmente eficientes, o que será valorizado em termos da avaliação. Assim, não será apenas tida em consideração a correção do código produzido, mas também a eficiência do mesmo.

2 Dados de entrada e de saída

O programa deverá ler os dados de entrada a partir do *standard input*. As opções da Tabela anterior têm a seguinte sintaxe.

- A leitura de uma mensagem é efectuada através do comando (comando `a`) com a seguinte sintaxe:

  ```
  a <mensagem>
  ```

  onde `<mensagem>` representa a mensagem a ler, contendo, no máximo, 140 caracteres (incluindo espaços). Esta operação não tem qualquer output. Cada mensagem é apresentada numa só linha, terminando com um carácter de mudança de linha `'\n'`. Para cada *hashtag* previamente encontrada deverá incrementar o respectivo contador, criando um novo registo na primeira ocorrência de uma *hashtag*.

- O comando `s` mostra o número total de *hashtags* distintas (`Ntags`) utilizadas até então e a soma do total de ocorrências de todas as *hashtags* (`Ntotal`). Não recebe argumentos. O output deverá ter a seguinte sintaxe:

  ```
  Ntags Ntotal
  ```

- O comando `m` mostra a *hashtag* mais popular (i.e., a `hashtag` com o maior número `N` de ocorrências). Não recebe argumentos. O output deverá ter a seguinte sintaxe:

  ```
  #hashtag N
  ```

  Nota: Em caso de igualdade deverá ser mostrada a *hashtag* com menor ordem lexicográfica (ordem alfabética).

- O comando `l` lista todas *hashtags* referidas nas mensagens por ordem decrescente segundo o número de ocorrências `N`. Em caso de igualdade, deverá seguir a ordem lexicográfica. O output deverá ter a seguinte sintaxe (em que `hashtag1` é a *hashtag* com maior número de ocorrências):

  ```
  #hashtag1 N
  #hashtag2 N
  #hashtag3 N
  ...
  ```

- O comando `x` termina o programa. Não recebe argumentos e não tem qualquer output.

Notas e sugestões adicionais:

- Para a leitura da mensagem, poderá utilizar a função `LeLinha` discutida nas aulas ou a função `fgets`[1]. Note que tanto a função `LeLinha` como o `fgets` lêem todos os caracteres até ao `'\n'` final, inclusive. Assim, dependendo de como o processamento dos comandos é

[1] http://www.dummies.com/how-to/content/how-to-use-the-fgets-function-for-text-input-in-c-.html

efectuado poderão ter de ler manualmente o `'\n'` nos comandos `m` e `l` à imagem do que ocorreu em todos os comandos do 1º projecto.

- Para processar cada palavra da linha lida poderá utilizar uma das duas versões da função `split` sugeridas na secção 4.
- Quando processar uma *hashtag*, as letras maiúsculas devem ser consideradas equivalentes às respectivas letras minúsculas (*case insensitive*). Da mesma forma, quando impressas no *standard output* as *hashtags* deverão ser escritas em letras minúsculas e precedidas de um cardinal (#).
- Cada *hashtag* terá no máximo 140 caracteres (incluindo o cardinal '#'), é precedida por um espaço, e não contém qualquer espaço ou carácter "branco" (ver listagem de separadores de palavras na secção 4). Cada *hashtag* terá no mínimo um carácter, para além do '#'.
- Uma mensagem poderá conter, por exemplo, duas ocorrências da mesma hashtag. Nesse caso, deverá incrementar duas vezes o respectivo contador.

3 Exemplos

- **#exemplo1**

Input:

```
a Four more years
a The North Pole is moving - and we're to blam  #science #climatechange
a If only Bradley's arm was longer. Best photo ever. #oscars
a pick the right data structure #iaed
a suggestion #iaed : try this test #project2 #iaed
l
m
x
```

Output:

```
#iaed 3
#climatechange 1
#oscars 1
#project2 1
#science 1
```
Output do comando `l`

```
#iaed 3
```
Output do comando `m`

- **#exemplo2**

Input:

```
a Four more years
a The North Pole is moving - and we're to blam  #science #climatechange
a If only Bradley's arm was longer. Best photo ever. #oscars
a pick the right data structure #iaed
a suggestion #iaed : try this test #project2 #iaed
s
x
```

Output:

```
5 7
```

- **#exemplo3**

Input:

```
a A #computer would deserve to be called #intelligent if it could deceive a human
into believing that it was human. #turing
a The first principle is that you must not fool #yourself and you are the easiest
person to fool. #feynman
m
a Prediction is very difficult, especially about the #future. #bohr
a #Science is a differential equation. #Religion is a boundary condition. #turing
m
a A man provided with paper, pencil, and rubber, and subject to strict
discipline, is in effect a #universal #machine. #turing
l
s
x
```

Output:

```
#computer 1
#turing 2
#turing 3
#bohr 1
#computer 1
#feynman 1
#future 1
#intelligent 1
#machine 1
#religion 1
#science 1
#universal 1
#yourself 1
11 13
```

Output dos 2 comando `m`

Output do comando `l`

Output do comando `s`

4 Parsing

Apresenta-se aqui uma sugestão de código para efectuar a divisão de cada mensagem em palavras.

```
#define NUMSEP 11
static const char separators[] = {'
','\t',',',';','.','?','!','"','\n',':','\0'};

void split(char *line)
{
  int i, j, k;
  char buffer[MAXLINESIZE];
  for(i = 0, k = 0; line[i] != '\0'; i++, k++) {
    buffer[k] = tolower(line[i]);
    for(j = 0; j < NUMSEP; j++) {
      if(line[i] == separators[j]) {
        if(k != 0) {
          buffer[k] = '\0';
          /* processar aqui palavra guardada em buffer */
        }
        k = -1;
      }
    }
  }
}
```

Nota: A função `tolower`[2] converte para minúscula qualquer carácter maiúsculo dado como argumento.

Em alternativa, poderá utilizar a função `strtok`[3] disponível em *string.h*, assim como as restantes funções deste *header* da *C standard library*. Usando a função `strtok` a função `split` toma a forma (mais simples):

```
void split(char *line){
    char *token = strtok(line, separators);
    while(token!=NULL) {
          /* processar aqui a palavra guardada em token */
        token = strtok(NULL, separators);
    }
}
```

5 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 proj2 *.c
```

o qual 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 deve escrever mensagens de aviso (*warnings*).

6 Execução do Programa

O programa deve ser executado da forma seguinte:

```
$ ./proj2 < test1.in > test1.myout
```

Posteriormente poderá comparar o seu output (`test1.myout`) com o output previsto (`test1.out`) usando o comando `diff`[4].

```
$ diff test1.out test1.myout
```

No *output* do comando `diff`, as linhas diferentes em ambos os ficheiros serão listadas, sendo as linhas esperadas precedidas por < e as linhas erradas no seu output precedidas por >.

[2] http://www.tutorialspoint.com/c_standard_library/c_function_tolower.htm
[3] http://man7.org/linux/man-pages/man3/strtok.3.html
[4] http://www.computerhope.com/unix/udiff.htm ou http://man7.org/linux/man-pages/man1/diff.1.html

7 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 os ficheiros fonte (`.c`) e cabeçalho (`.h`) que constituem o programa.
- 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`, 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 grupo. Exemplos de casos de teste serão oportunamente fornecidos.
- Data limite de entrega do projecto: **14 de Maio de 2016 (23h59m)**. Até à data limite poderá efectuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a última entrega efectuada. Deverá portanto verificar cuidadosamente que a última entrega realizada corresponde à versão do projecto que pretende que seja avaliada. Não serão abertas excepções.

8 Avaliação do Projecto

8.1 Componentes da Avaliação

Na avaliação do projecto serão consideradas as seguintes componentes:

1. A primeira componente avalia o desempenho e 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. Nesta componente será também utilizado o sistema ***valgrind***[5] 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 este sistema para fazer debugging do código e corrigir eventuais incorrecções, antes da submissão 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, sendo a respectiva classificação 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.

[5] Para uma pequena introdução ao Valgrind, veja (por exemplo): http://cs.ecs.baylor.edu/~donahoo/tools/valgrind/

8.2 Atribuição Automática da Classificação

- A classificação da primeira componente da avaliação do projecto é obtida automaticamente através da execução de um conjunto de testes executados 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 podem 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 disponibilizada 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.