Pràctica obligatòria – Docència repetida

Programació 1 – Grau en Enginyeria Informàtica

Curs 2018-2019

Exercici 1 (5 punts) – Missatges de SPAM amb paraules ponderades

Introducció

Des de fa temps, els missatges de correu no desitjat (SPAM) són un problema per als que utilitzem habitualment el correu electrònic. El que tractarem de fer en aquest exercici és implementar un detector de missatges de SPAM senzill, que reconegui i compti certes paraules en un text donat i que finalment, amb els resultats obtinguts, identifiqui el missatge com a SPAM o com a correu ordinari.

Enunciat

La pràctica consistirà en 3 fases:

1. Obtenir les paraules a identificar: Una de les tècniques que fan servir els programes anti-SPAM és analitzar el text buscant una sèrie de paraules que es fan servir sovint en aquest tipus de correu, i d'aquesta forma poder decidir si el correu s'ajusta a les característiques de SPAM. En aquesta fase, el que farem és obtenir les paraules que volem identificar en el text introduïdes des de l'entrada estàndard (teclat o fitxer en cas de fer un redireccionament). El format a seguir per a l'entrada d'aquestes paraules serà el següent:
   - Les paraules vindran separades per un o més espais o canvis de línia.
   - Una seqüència de paraules a identificar s'acabarà amb un punt.
   - Hi ha dos seqüències de paraules, les paraules de la primera seqüència tenen més pes que les de la segona a l'hora de detectar SPAM.
2. Identificar les paraules en el text: Una vegada sabem les paraules que volem buscar al text, passarem a l'obtenció del text que, com les paraules a identificar, ens vindrà introduït per l'entrada estàndard. L'únic requisit a seguir per a l'entrada d'aquest text és que ha d'acabar amb 2 punts simples, que poden estar separats per espais. No es permet emmagatzemar tot el text en memòria, per tant, el que es pretén és llegir el text paraula a paraula i tractar-les a mesura que les anem llegint.
3. Decidir si el missatge obtingut és SPAM: finalment, una vegada obtinguts els resultats de buscar les paraules dins el text, el que hem de fer es decidir si el missatge llegit és SPAM. Per prendre aquesta decisió, ho farem de la següent forma: si apareixen 1 o més vegades el 50% o més de les paraules de la primera seqüència buscades en el text, i si apareixen 2 o més vegades el 50% o més de les paraules de la segona seqüència buscades en el text, llavors considerarem que el missatge rebut és SPAM, altrament considerarem que el missatge és de correu ordinari.

Consideracions

- Com a molt tindrem 2 llistes de paraules.
- Com a molt tindrem 20 paraules a identificar per cadascuna de les llistes.
- Cada paraula tindrà com a molt 30 caràcters.
- El programa no ha de diferenciar entre majúscules i minúscules, és a dir, per al nostre programa és el mateix “HoLa” que “hola”.

- Juntament amb l'enunciat de la pràctica trobareu fitxers de prova (ex1-jp?.txt) que us poden donar una idea de com us poden venir les dades redireccionant l'entrada des d'un fitxer.
- Els resultats que haurem d'obtenir del nostre programa són els següents: (i) El número de vegades que apareix cada paraula (ii) El rati de paraules que apareixen 1 o més cops de la primera seqüència, o 2 o més cops de la segona seqüència en el text (iii) El resultat de si el missatge és SPAM o no. Podeu veure varis exemples de com ha de ser la sortida del vostre programa dins el fitxer "exemple-sortida.txt" que trobareu junt amb l'enunciat.
- Als jocs de proves (ex1-jp?.txt), en alguns casos, al tractar-se de casos reals, s'ha procurat separar les paraules de símbols com parèntesis, cometes, etc... amb espais per a facilitar una mica la lectura de les paraules. Per tant, només cal tenir en compte que una paraula esta formada únicament per caràcters alfabètics i s'acaba quant es troba un caràcter no alfabètic.

Exercici 2 (5 punts) – Comprovació de Sudokus grans

Introducció

Un Sudoku gran és un trencaclosques de col·locació que requereix paciència i una certa habilitat lògica. En el nostre cas serà un Sudoku més gran que el normal, on el joc es composa d'una graella de 16x16 cel·les subdividida en 16 subgraelles de 4x4 anomenades regions. Donats uns quants números inicials, l'objectiu és col·locar un número del 0 al 9 i una lletra de l'A (representa el nombre 10) a l'F (representa el nombre 16) en cada cel·la de tal manera que mai coincideixin dos números iguals en cada línia horitzontal, vertical o en cada regió. A continuació podeu veure un Sudoku resolt:

```
0c19 7d6f 8e5b 3a24
8462 b530 dacf 7e19
a35f 81e4 9672 bcd0
deb7 9ac2 3041 58f6

3da8 4906 f1be c275
c72e f3ab 5864 190d
10f5 d87e c2a9 643b
4b96 5c21 7d03 ef8a

6aed 2058 1937 4bcf
9f4c 67d3 0be8 25a1
5280 ab19 64fc d3e7
b173 e4fc a52d 9068

280a 1ebd 4396 f75c
76cb 024a efd5 8193
e5d1 3f97 bc8a 0642
f934 c685 2710 adbe
```

Enunciat

L'exercici consistirà en 3 fases:

1. Obtenir el Sudoku: El nostre programa rebrà des de l'entrada estàndard (teclat o fitxer en cas de fer un redireccionament) la seqüència de números que formaran el nostre Sudoku. Els números seran enters del 0 al 9 i de l'A fins a l'F, separats per un o més espais, tabuladors o canvis de línia. Rebrem un mínim de 256 números o lletres per poder completar el nostre Sudoku. El primer número que rebrem serà el número de més a l'esquerra de la primera fila,

la de dalt. El següent número serà el que va immediatament a la dreta de l'anterior i així successivament fins al novè número que formarà la primera fila. Els següents nou números formaran la segona fila i així successivament fins a formar el Sudoku complert quant haguem llegit els 256 números que el formen. Una vegada llegit tot el Sudoku, el mostrarem per pantalla tal com surt en l'exemple anterior.

2. Comprovar que el Sudoku és correcte: El segon pas consistirà en comprovar que el Sudoku és correcte, és a dir, que cada una de les seves files conté tots els números del 0 al 9 i de l'A a l'F sense repeticions, el mateix per cada una de les seves columnes i per cada una de les seves regions de 4x4. Quant trobeu un error, heu de dir en quina fila (de la 0 a la F), columna (de la 0 a la F) o regió (de la superior esquerra 0,0 a la inferior dreta 4,4) l'heu trobat. Haureu d'aturar-vos al trobar el primer error, amb un sol error el Sudoku ja és incorrecte.
3. Si la solució és correcta, contar el nombre de vegades que l'índex de fila o columna coincideix amb el valor de la casella, com també si el valor coincideix amb l'índex de regió, on l'índex de regió va d'esquerra a dreta i de dalt a baix.
4. Mostrar el resultat: Si en el pas anterior heu trobat algun error, haureu de dir que el Sudoku és incorrecte. En cas contrari, haureu de dir que el Sudoku és correcte.

Exemple de Sudoku incorrecte:

```
0c19 7d6f 8e5b 3a24
8462 b530 dacf 7e19
a35f 81e4 9672 bcd0
deb7 9ac2 3041 58f6

3da8 4906 f1be c275
c72e f3ab 5864 190d
10f5 d87e c2a9 643b
4b96 5c21 7d03 ef8a

6aed 2068 1937 4bcf
9f4c 67d3 0be8 25a1
5280 ab19 64fc d3e7
b173 e4fc a52d 9068

280a 1ebd 4396 f75c
76cb 024a efd5 8193
e5d1 3f97 bc8a 0642
f934 c685 2710 adbe

El Sudoku te un error a la columna 7.
La solucio al Sudoku es INCORRECTA!
```

Jocs de proves

Junt amb l'enunciat es deixaran una sèrie de fitxers per a que pugueu provar el funcionament dels exercicis i que es faran servir per a l'avaluació. Podeu utilitzar-los fent un redireccionament de l'entrada estàndard de la següent forma:

```
$ ./ex1 < ex1-jp0.txt
```

Hi ha diferents jocs de proves (exX-jpX.txt) i els resultats que hauria de donar per cada un d'ells (exX-jpX.txt.out).

El programa ha d'estar implementat com si les dades fossin introduïdes per teclat. Per a provar els

jocs de proves no cal fer cap modificació en el vostre programa, sempre i quant estigui ben implementat.

Avaluació

- La pràctica s'ha d'implementar en el llenguatge ANSI C++ i s'ha d'executar correctament en una plataforma Linux. Per garantir que satisfà l'estàndard ANSI compileu afegint l'opció `-ansi`. És a dir, la comanda de compilació ha de ser:

```
$ g++ programa.cpp -o executable -ansi
```

- S'han d'implementar els exercicis utilitzant les tècniques de disseny descendent vistes a classe.
- La pràctica es puntua sobre 10 punts i el seu pes a l'avaluació final és del 40%.
- La data límit per al lliurament de la pràctica és el 17 de maig.
- La pràctica es lliurarà via el Campus Virtual (CV), dins l'apartat Activitats.
- Es recomana posar comentaris a les parts importants del programa dins els fitxers .cpp dels exercicis que ajudin a interpretar l'algorisme implementat.
- La pràctica s'ha de resoldre **individualment**.
- Com a comentaris de l'activitat comenteu breument l'estratègia emprada per resoldre cada problema.
- El dia de l'entrega es farà la validació de la pràctica a classe. Aquesta **validació** és **obligatòria** per a ser avaluada.

Aspectes a tenir en compte

Alguns aspectes que heu de tenir en compte a l'hora de realitzar les vostres pràctiques i que puntuaran negativament si no els teniu en compte:

- Nitidesa en el codi (tabulació correcta, no fer càlculs innecessaris, no repetir codi, utilització correcta dels recursos de la màquina...).
- Utilitzar estructures algorísmiques adients per als problemes a resoldre.
- Utilitzeu noms de variables entenedors.
- Llegiu atentament l'enunciat i no implementeu funcionalitats diferents a les que us demana.
- Queda prohibit utilitzar la instrucció de salt `goto`, o instruccions per alterar el funcionament normal d'un bucle com `continue` o `break` (l'únic cas en que es pot fer servir el `break` és en el cas que facis anar la instrucció `switch`).
- No es poden utilitzar llibreries no estàndard com la `conio.h`.
- En cas de realitzar pràctiques de forma "col·laborativa" entre diferents grups, esmentar-ho en el moment de l'entrega o en els comentaris, encara que finalment s'entreguin les pràctiques per separat o individualment.
- Per al primer exercici, no es permès emmagatzemar tot el text en memòria i després tractar-lo, ja que el text pot ser infinit. L'incompliment d'aquest requeriment significarà un 0 a l'exercici.
- Si es detecta que la pràctica és copiada, la nota és un 0. Tant pel que còpia com pel que deixa copiar.

Mètode de correcció

Per a la validació del funcionament de la pràctica s'utilitzarà el Makefile i els corresponents jocs de proves. Tant els jocs de proves com el fitxer Makefile que s'utilitzarà per a la correcció els podreu trobar a la carpeta de la pràctica a l'apartat de Recursos del Campus Virtual. A l'hora de presentar la pràctica a través del CV, haureu d'enviar els següents fitxers:

- Makefile: El mateix fitxer "Makefile" que trobareu al CV.

- ex?-jp?.txt: Els fitxers de proves que trobareu al CV.
- ex?.cpp: Els fitxers on implementareu els vostres programes, un per cada exercici.

Per poder enviar tots els fitxers sense problemes els podeu comprimir utilitzant la comanda 'tar' de la següent forma:

Suposant que teniu els exercicis resolts dins la carpeta 'prac2' podeu fer:

```
$ cd  prac2
$ tar  cvfz  prac2.tgz  *
```

El fitxer resultant és el fitxer prac2.tgz que conté tots els fitxers de la carpeta 'prac2'.

Comandes que se seguiran per la correcció:

1. $ make clean

Per eliminar els possibles fitxers binaris que hi pugui haver.

2. $ make all

Per compilar els exercicis seguint l'estàndard ANSI.

3. $ make test

Per provar el correcte funcionament dels programes d'acord amb l'enunciat.

Podeu veure un exemple d'una sortida correcta després d'executar aquestes 3 comandes dins el fitxer “exemple-sortida.txt” que trobareu junt amb l'enunciat.