Segona pràctica obligatòria

Programació 1 — Grau en Enginyeria Informàtica

Curs 2019–2020

Exercici 1 (5 punts) – Cerca d'anagrames

Introducció

Una de les funcions més comuns que ens podem trobar en un editor de text és la de buscar una paraula o frase dins el mateix. L'objectiu d'aquesta pràctica és el de dissenyar e implementar una variant de la funció de cerca en la que buscarem paraules que siguin anagrames d'una o més paraules donades. Un anagrama és una paraula formada per la transposició de les seves lletres. Aquest tipus de cerca és utilitzada en aplicacions reals en les que en cas de no trobar la paraula introduïda es busquen anagrames de la mateixa intentant preveure que l'usuari s'ha equivocat. Diversos buscadors de pàgines web utilitzen tècniques d'aquest estil per ajudar a l'usuari en la cerca.

Enunciat

L'exercici consistirà en 3 fases:

Obtenir les paraules: El nostre programa rebrà des de l'entrada estàndard (teclat o fitxer en cas de fer un redireccionament) les paraules a buscar. Aquestes paraules seran les primeres introduïdes, poden estar separades per un o varis espais i s'acabaran amb un punt simple '.'.

Buscar anagrames en el text: Una vegada sabem les paraules que volem buscar en el text, passarem a l'obtenció del text que, a l'igual que 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 un punt simple '.'. 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. Per aquesta pràctica, considerarem que hem trobat un anagrama, quant la paraula del text tingui les mateixes lletres que alguna de les paraules a buscar encara que siguin en diferent ordre, és a dir, les lletres de la paraula del text són una transposició de les de alguna de les paraules a buscar. S'ha de mostrar la paraula a buscar i l'anagrama.

Mostrar el resultat de la cerca: Finalment, una vegada obtinguts els resultats de buscar les paraules dins el text, el que hem de fer és mostrar el resultat. La informació que haurà de retornar el nostre programa és:

- Cada una de les paraules del text que són anagrames d'alguna de les paraules que estem buscant. Es podrà mostrar aquesta informació a mesura que l'anem trobant en el text.
- El número total de vegades que hem trobat anagrames de les paraules en el text.

Consideracions

- Com a molt tindrem 10 paraules a buscar.
- Cada paraula tindrà com a molt 50 caràcters.
- El programa no ha de diferenciar entre majúscules i minúscules, és a dir, per al nostre programa és el mateix "HoLa" que "hola".
- Per facilitar l'enunciat, els jocs de proves (ho veurem més endavant) es faran sobre textos en anglès, és a dir, no cal considerar els accents a les paraules. Per tant, una paraula estarà formada per lletres $\{a, \ldots, z, A, \ldots, Z\}$. Qualsevol altre caràcter es pot considerar com un espai.

Exercici 2 (5 punts) – Comprovació de sortida del laberint

Introducció

En aquest exercici haurem de decidir si la solució proposada per sortir d'un laberint és correcta o no. Exemple d'un laberint on els espais ' ' indiquen els passadissos, les 'X' indiquen les parets i la 'I' la posició inicial:

```
  XX   XXXXX    XXXXX
   XXXX        XX  X
 XX     XXX   XXXXXXX
  X XXX XXXXXXX    X
  X XXX XXI   XX X XX
 XX  XX XXXXX XX X XX
   X XX       XX X X
  X  XXXXXXXXX    XXX
  X XX      XXXXXXXXX
 XX XXXX   XXXX    X
```

Enunciat

L'exercici consistirà en 3 fases:

Obtenir el laberint i el recorregut: El laberint tindrà una mida de 10 files i 20 columnes. Ens vindrà introduït per l'entrada estàndard (teclat o fitxer) en forma de 10×20 caràcters, on les 'X' representaran parets del laberint (espai que no podem trepitjar) i les 'O' (lletra) representaran espais buits pels que podem passar. Seguidament, rebrem per la mateixa entrada un número de fila i de columna que ens indicaran la posició de la que partim per sortir del laberint. Finalment, ens vindrà la seqüència de passos que hem de comprovar si ens porta a la sortida del laberint. Aquesta seqüència de passos vindrà formada per les lletres n, s, e, o, nord, sur, est, oest respectivament. Cada pas indica que avancem en la direcció indicada una posició. Com a màxim tindrem tantes passes com caselles tingui el laberint.

Comprovar que el recorregut ens fa sortir del laberint: Considerarem que hem sortit del laberint quan, partint de la posició d'inici i seguint els passos indicats, arribem a un dels marges de la taula que forma el laberint (files 0 o 9, o columnes 0 o 19). Si la posició inicial o alguna de les passes que fem no passen per espais buits, la solució no és correcta. Si acabem de donar passes i no hem passat per cap dels marges del laberint, la solució no és correcta. Si seguint les passes passem per un dels marges del laberint, la solució és correcta independentment de si després tenim més passes o no. Podem passar tantes vegades com vulguem per una mateixa casella.

Mostrar el resultat: Per acabar el nostre programa mostrarem el laberint i el recorregut que hem seguit per sortir d'ell. Per mostrar el laberint farem servir la lletra 'X' per indicar una paret, espai en blanc ' ' per indicar una posició buida del laberint per la que no hem passat, la lletra 'I' per indicar la posició inicial de la que partim i la lletra 'i' per indicar les posicions que anem recorrent seguint les passes. Finalment, mostrarem per pantalla una frase indicant si la solució al laberint és correcta o no (veure exemple més endavant).

Consideracions

- El laberint sempre serà de 10 files i 20 columnes.
- Com a molt tindrem tantes passes com caselles té el laberint.
- Per les coordenades d'inici considerarem que tant la fila i la columna comencen per 0.
- Si durant l'entrada de dades detecteu algun caràcter erroni, podeu finalitzar el programa.
- El recorregut per sortir del laberint pot passar vàries vegades pel mateix lloc sempre i quan no trepitgi cap paret 'X'.

Exemple de sortida correcta per al laberint anterior:

```
 XX   XXXXX    XXXXX
  XXXX        XX  X
XXiiiiiXXX   XXXXXXX
 XiXXXiXXXXXXX    X
 XiXXXiXXIiiiXX X XX
XXiiXXiXXXXXiXX X XX
  XiXXiiiiiiiXX X X
 XiiXXXXXXXXX    XXX
 XiXX      XXXXXXXXX
XXiXXXX   XXXX    X

Coordenades d'inici (F, C): 4, 9
Passes del recorregut: eeessooooooonnnnooooosssessoss
La solució al problema del laberint és correcta!
```

Jocs de proves

Juntament 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 varis 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.cc -o executable -ansi`
- La pràctica es puntua sobre 10 punts i el seu pes a l'avaluació final és del 25%.
- La pràctica es lliurarà via el Campus Virtual (CV), dins l'apartat Activitats.
- Es recomana posar comentaris dins els fitxers `.cpp` dels exercicis que ajudin a entendre l'algorisme implementat.
- La pràctica s'ha de resoldre individualment o en grups de màxim 2 persones.
- A l'**informe de l'activitat** heu d'indicar si la pràctica s'ha realitzat de forma individual o en grup. A més heu d'indicar els membres que composen el grup. Aquest informe consisteix en:
  - Resum de l'estratègia emprada per resoldre cada problema (100 paraules com a màxim),
  - Pseudo-codi[1] de les funcions principals.
- El primer dia de laboratori posterior a l'entrega, es realitzarà la validació de la pràctica individualment. Per a que la pràctica sigui avaluada **caldrà superar aquesta validació**.

[1] `https://en.wikipedia.org/wiki/Pseudocode`

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, **encara que la pràctica *funcioni***:

- Nitidesa en el codi (tabulació correcta, no fer càlculs innecessaris, 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.
- No es pot utilitzar la instrucció de salt `goto`, o instruccions per alterar el funcionament normal d'un bucle com `continue` o `break`[2].
- 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 l'informe, 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 copia com pel que es 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 `prac` podeu fer:

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

El fitxer resultant és el fitxer `prac.tgz` que conté tots els fitxers de la carpeta `prac`.

Comandes que se seguiran per la correcció:

`$ make clean`

Per eliminar els possibles fitxers binaris que hi pugui haver.

`$ make all`

Per compilar els exercicis seguint l'estàndard ANSI C++.

`$ make test`

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

[2] L'únic cas en que es pot fer servir el `break` és en el cas que es faci anar la instrucció `switch`.