PRÀCTICA FINAL

INTRODUCCIÓ

Els objectius d'aquesta pràctica són:

- Implementar un avaluador d'expressions matemàtiques.
- Implementar i valorar un algorisme recursiu.
- Utilitzar adequadament el disseny descendent.
- Documentar el codi fent servir precondicions i postcondicions.
- Fer un joc de proves adequat.

La pràctica la realitzareu en grups de tres estudiants. Utilitzareu el Code:Blocks com a entorn de desenvolupament. En l'entrevista haureu de mostrar al professor que el programa implementat pels experiments us funciona i enteneu com va. **La pràctica té un pes del 60% en la nota de pràctiques, que haurà de ser igual o superior a 5 per poder aprovar l'assignatura**. La qualificació d'aquesta pràctica serà NO ACCEPTADA (0), ACCEPTADA (5) o EXCEL·LENT (10).

Per tenir la qualificació d'ACCEPTADA cal haver assolit tot el que marquem a l'enunciat. Si proposeu millores, feu una presentació ben feta, argumenteu correctament durant l'entrevista... augmenteu les possibilitats de tenir un EXCEL·LENT. Al final de l'enunciat hi ha alguns suggeriments de millora.

L'avaluador d'expressions matemàtiques

L'objectiu final d'aquesta pràctica és que dissenyeu, implementeu i valoreu un avaluador d'expressions com ara les següents:

35-(12*8), (1000-500)*(2*1000), (((10+2)+5)+5)*(20-2), 20/(35/7)...

Aquesta solució estarà basada en el **disseny recursiu de funcions** (precisament per permetre la resolució d'expressions amb l'ús de parèntesis).

Noteu que es tracta de resoldre recursivament expressions del tipus **A op B** on A i B poden ser números o bé expressions matemàtiques expressades entre parèntesis, i op és un operador matemàtic.

Per afrontar aquests problemes de forma correcta i formal, es podria utilitzar la definició de gramàtiques, concepte que estudiareu més endavant a l'assignatura de Llenguatges Formals. En aquesta pràctica volem que ho solucioneu amb les eines de què disposeu fins el moment!

Per a resoldre el problema diferenciarem tres fases. La **primera fase** consisteix en llegir una expressió des de fitxer. Cada línia del fitxer tindrà una expressió de com a molt una longitud de

256 símbols (no s'hi inclouen els bytes que conformen el salt de línia). Recomanem utilitzar la funció **fgets**, tenint en compte que les línies acaben amb un salt de línia i caldrà "menjar-se'l".

La **segona fase** consisteix en una anàlisi d'errors. Amb aquest objectiu, heu de dissenyar:

- Una funció que comprovi que els símbols introduïts són vàlids. Els vàlids són 0123456789+-*/()
- Una funció que comprovi que el nombre de parèntesis d'obertura i tancament és coherent.
- Una funció que comprovi que els símbols tenen un veïnatge correcte.

Exemple de símbols invàlids: (4;(5+(a-8]))

Exemple de problema amb els parèntesis: (4(5+(a-8))*

Exemple de veïnatge incorrecte: (4(5+(a-8)))

Si es troba un error, s'informarà per pantalla i es llegirà la següent expressió. Si no es troba error, s'arriba a la **tercera fase**, que consisteix en avaluar l'expressió de forma recursiva. Caldrà fer un disseny recursiu que permeti llegir la terna A op B, diferenciant els casos directes i els recursius. Com que hem fet comprovacions d'error, la precondició és que l'expressió a avaluar ja és correcta.

Molt important: L'entrada del programa serà un fitxer. **No s'acceptaran pràctiques on l'expressió a avaluar s'hagi d'introduir de forma interactiva.** També caldrà tenir molt clar com procediu i quines estratègies fareu servir. En aquest sentit, primer hi haurà una etapa de disseny on haureu de discutir quins camins emprendre.

Proves

Se us proporcionarà a Moodle un fitxer proves.txt el qual contindrà les diferents proves a executar. Per acceptar la pràctica **és essencial que aquestes proves es superin amb èxit**: és a dir que el programa no generi errors d'execució, es detectin els errors anteriorment descrits i, en cas que l'expressió sigui correcta, que el resultat sigui l'esperat.

LLIURAMENT

La darrera sessió de la pràctica està dedicada a l'entrevista. **Com a resultat de la pràctica heu de lliurar el codi font i una presentació en PowerPoint.** En la presentació heu de fer èmfasi en les decisions de disseny, problemes que us hagueu trobat i resultats obtinguts. És obligatori presentar un fitxer de joc de proves propi, que contempli diferents casos. El codi estarà degudament comentat, seguint el que ja vau treballar en la Pràctica 2.

Per tenir més nota

Per optar a l'excel·lent podeu considerar les següents opcions:

- Fer una segona versió del programa iterativa (dos codis font: la versió recursiva, la iterativa i les biblioteques de funcions en principi compartides)
- O bé fer una versió que permeti expressions del tipus A op B op C op D ...

Per assolir l'excel·lent és condició *sine qua non* tenir un codi font ben estructurat i que plantegi solucions eficients, així com una presentació de qualitat. PENSEU QUE ÉS LA PRÀCTICA FINAL I ÉS LA QUE COMPTA MÉS!! Endavant les atxes!