Estructura de Computadors II

Primera Pràctica

Data d'entrega: 20 de Novembre de 2015

Curs 2015/2016

Resum

Aquesta pràctica consta de dos apartats. En primer lloc haureu d'implementar una biblioteca de gestió de llistes. En segon lloc haureu d'implementar una interfície gràfica senzilla que permeti interactuar amb la biblioteca de gestió de llistes. Ja que la segona part haurà d'interactuar amb la primera, és molt important que es compleixin perfectament les especificacions. A més a més, se us proporcionarà part del codi de la pràctica ja fet.

1 Objectius de la pràctica

Amb la realització d'aquesta pràctica s'espera que:

- Comprengueu millor el funcionament del vector d'excepcions i l'accés a dispositius mitjançant interrupcions. En aquest cas, accedireu al ratolí.
- Comprengueu millor els mecanismes bàsics que ha de proporcionar una CPU per a facilitar la programació del *software* del sistema. En aquest cas només ens centrarem en l'ús del TRAP #15.
- Milloreu les vostres habilitats de programació en general i de programació en ensamblador en particular.
- Comprengueu millor com es tradueixen a baix nivell les estructures de dades i operacions d'alt nivell que estau acostumats a emprar en altres contextos i assignatures.
- Sigueu capaços de programar codi fàcilment utilitzable per terceres persones, complint les especificacions i emprant estructures i interfícies clares.
- Sigueu capaços d'entendre codi realitzat per terceres persones i d'integrar-lo en els vostres projectes.

La pràctica es realitzarà en ensamblador de M68000 sobre l'eina Easy68k. La pràctica es divideix en dues parts:

- Gestió de llistes. Haureu de programar part del codi d'una biblioteca de gestió de llistes. Per això haureu d'entendre les especificacions i el codi subministrat.
- Interfície gràfica. Haureu de programar el codi d'una biblioteca que permeti interactuar amb les llistes de manera gràfica. Per això haureu d'entendre les especificacions, el codi subministrat, i interactuar amb la vostra biblioteca de gestió de llistes. També haureu de realitzar tasques com accedir al ratolí per interrupció i accedir a la pantalla per a inicialitzar-la, dibuixar-la i fer doble buffer mitjançant el TRAP #15.

2 Aspectes generals

2.1 Descripció de les llistes

En el context d'aquesta pràctica, una llista és una estructura de dades capaç d'emmagatzemar varis elements. Una llista pot emmagatzemar un número màxim prefixat d'elements. Ara be, aquest número màxim pot diferir d'una llista a una altra. Per tant, la biblioteca ha d'esser capaç de gestionar llistes amb tamanys arbitraris.

A cada element de la llista s'hi poden emmagatzemar varis WORDs de dades. Aquestes dades no tenen significat per la biblioteca de gestió de llistes, simplement s'emmagatzemen. La quantitat de WORDs que s'emmagatzema a cada element és fixada per a una llista concreta, però cada llista pot emmagatzemar quantitats distintes de dades. Per tant, la biblioteca també ha d'esser capaç de gestionar llistes amb elements de tamanys arbitraris.

La llista que s'ha d'implementar és una llista enllaçada. Per tant, internament la llista està composada per un conjunt de nodes i cada node conté, a més de les dades, un punter al següent node. Notau que això implica que dos nodes consecutius de la llista no tenen perquè ocupar posicions contigües dins la memòria. Donat un node es pot saber quin node el segueix consultant l'esmentat punter.

L'estructura exacta de les llistes que heu de gestionar és la següent. El terme NULL es correspon, necessàriament, a un LONG amb un valor de 0.

1. Capçalera

   N (1 WORD) : Número de WORDs de dades en cada node.

   OCCUPIED (1 LONG) : Punter al primer node que conté dades de la llista, o be NULL si no n'hi ha cap.

   FREE (1 LONG) : Punter al primer node lliure de la llista, o be NULL si no n'hi ha cap.

2. Nodes. Cada node conté els següents camps

   NEXT (1 LONG) : Punter al següent node o NULL si no n'hi ha més.

   DATA (N WORDs) : Els N WORDs de dades.

Notau que el llistat de Nodes és únic, i contendrà tant els nodes ocupats com els lliures. En general, aquests nodes formaran part de dues sub-llistes enllaçades disjuntes (OCCUPIED i FREE), i per a accedir a cada una de les sub-llistes s'haurà de començar, necessàriament, pels punters OCCUPIED i FREE. Afegir un element a la llista significa llevar-lo de la sub-llista FREE, posar-hi les dades pertinents a dins i posar-lo dins la sub-llista OCCUPIED. Eliminar un element de la llista significa llevar-lo de la sub-llista OCCUPIED i posar-lo dins la sub-llista FREE. Notau que, si be el número de nodes de cada sub-llista pot anar canviant a mesura que es facin insercions i borrats, el número total de nodes (lliures més ocupats) es mantindrà constant.

La Figura 1 sumaritza, amb un exemple, l'estructura de les llistes. Notau, per exemple, que si be els nodes de l'exemple es troben en memòria en l'ordre (0, 1, 2, 3, 4), la sub-llista d'elements ocupats es llegiria en l'ordre (2, 4, 0) i la sub-llista d'elements lliures es llegiria en l'ordre (1, 3). Per tant, la llista contindria els elements 2, 4 i 0. Els elements 1 i 3 estarien disponibles quan s'hagués d'afegir un nou element.

La Figura 2 mostra els continguts de memòria corresponent a una llista amb un total de 5 nodes de 3 WORDs de dades cada un. A la posició $14B8 es pot veure el WORD amb valor 3, indicant que cada element conté tres WORDs de dades. A continuació apareix el punter OCCUPIED seguit del punter FREE. Després apareixen els cinc nodes, cada un amb el seu camp NEXT i els 3 WORDs de dades.

2.2 Subrutines de biblioteca

Ja que heu d'implementar dues biblioteques, les subrutines que se us demanen han d'esser subrutines de biblioteca. Si heu d'implementar alguna subrutina auxiliar, no té per què ser de biblioteca.

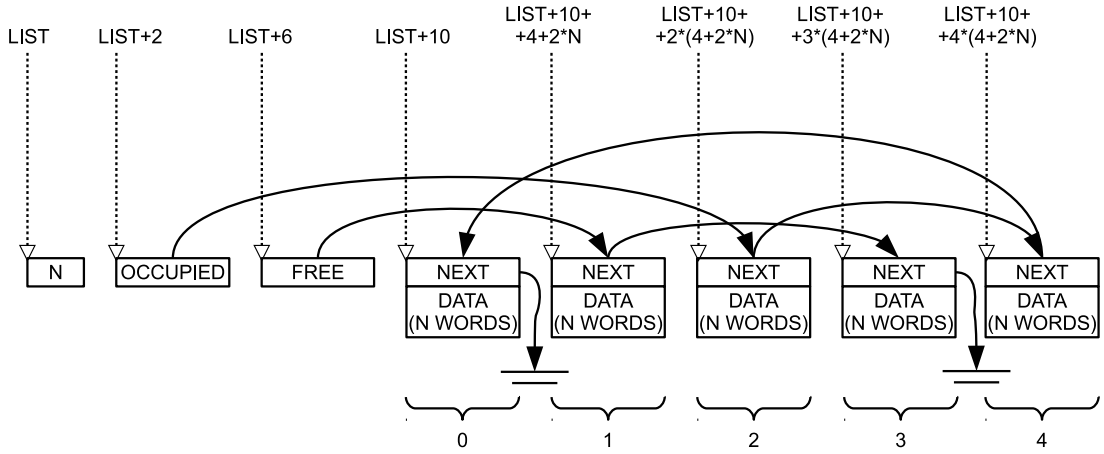


Figura 1: Exemple de llista. A la part de dalt es mostren adreces de memòria. Els números mostrats a la part d'abaix s'empraran per a facilitar la descripció a l'enunciat. Les flextes representen punters.

```
$ Address:
000014B0  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
000014B0: 58 8F 10 3C 00 09 4E 4F 00 03 00 00 14 EA 00 00
000014C0: 14 E0 00 00 00 00 00 02 00 01 00 00 00 00 14 C2
000014D0: 00 05 00 04 00 03 00 00 00 00 00 08 00 07 00 06
000014E0: 00 00 14 D6 00 0B 00 0A 00 09 00 00 14 CC 00 0E
000014F0: 00 0D 00 0C FF FF FF FF FF FF FF FF FF FF FF FF
```

Figura 2: Continguts de la memòria corresponents a una llista amb 5 nodes de 3 WORDs de dades cada un. La llista comença a \$14B8.

L'objectiu de les subrutines de biblioteca és que siguin fàcilment utilitzables per altres programes. Un problema freqüent quan es fa servir una biblioteca és l'aparició de *conflictes d'etiquetes*: és possible que una etiqueta que heu definit en la vostra llibreria també s'empri en el programa que fa ús de la vostra llibreria. Per tal d'evitar aquests problemes *heu de seguir aquests dos criteris*:

- Emprau etiquetes locals sempre que sigui possible.
- Per a les etiquetes globals, emprau sempre el prefix de dues o tres lletres que se us indiqui. Per a les etiquetes globals de la biblioteca de llistes heu d'emprar sempre el prefix LS_ (de *LiSt*). Per a les etiquetes globals de la biblioteca de la interfície gràfica heu d'emprar el prefix LP_ (de *List Plot*). Les etiquetes globals de programa principal empraran el prefix TST_ (de *TeST*) i els noms de les macros relacionades amb la sortida per pantalla començaran amb SC_ (de *SCreen*). Per exemple, si a la biblioteca de gestió de llistes voleu crear una subrutina anomenada AUXILIAR, l'haureu d'anomenar LS_AUXILIAR.

Ja heu vist a Estructures de Computadors I el concepte de subrutina de biblioteca. També teniu disponible a Campus Extens un document on s'expliquen, entre d'altres, els conceptes principals de les subrutines de biblioteca. Els aspectes més rellevants a complir són:

- Les subrutines de biblioteca reben els parèmetres a través de la pila i retornen paràmetres a través de la pila.
- Les subrutines de biblioteca que necessitin variables temporals per als seus càlculs, empraran la pila com a espai d'emmagatzematge d'aquestes variables.
- Les subrutines de biblioteca no modifiquen cap paràmetre d'entrada a no ser que el paràmetre de sortida l'hagi de sobreescriure.
- En sortir d'una subrutina de biblioteca tots els registres han de contenir els mateixos valors que tenien en entrar-hi.
- No és responsabilitat de la subrutina de biblioteca posar els paràmetres d'entrada dins la pila ni de buidar la pila una vegada acabada la seva execució.

Consultau la documentació en Campus Extens per a obtenir més informació sobre les subrutines de biblioteca.

3 Biblioteca de gestió de llistes

A continuació es descriuen les subrutines de la biblioteca de gestió de llistes. Recordau que algunes d'elles ja se us proporcionen implementades. Per a cada una d'elles s'indica com han d'estar els paràmetres d'entrada dins la pila *abans* de cridar a la subrutina (Pre) i com han d'estar els paràmetres de sortida, si n'hi ha, *després de retornar* de la subrutina (Post). Recordau que dins la subrutina, la pila contindrà també l'adreça de retorn. *Aquest és el motiu de que les especificacions de l'enunciat i les que apareixen als comentaris del codi difereixin en 4 bytes.*

En línies generals, afegir un element a la llista significa ajustar els diferents punters per tal de traspassar un node de la sub-llista FREE a la sub-llista OCCUPIED. Eliminar un element de la llista significa, a grans trets, ajustar els punters per tal que l'element a eliminar passi de la sub-llista OCCUPIED a la sub-llista FREE. El moviment de nodes d'una sub-llista a una altra es fa ajustant els punters, no copiant tot el node. *Trobareu informació més detallada sobre què ha de fer cada subrutina als propis comentaris del codi que se us subministra.*

LS_INIT Inicialitza una llista com a una llista buida.

Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista
(A7+4)	WORD	Número d'elements de la llista (M)
(A7+6)	WORD	Número de WORDs de dades per element (N)

Post : Cap.

Observacions : El Punter a la Llista ha d'apuntar a una zona de memòria on s'hagi reservat espai suficient per a emmagatzemar la llista. Seguint la nomenclatura anterior, l'espai requerit per una llista és $2 + 4 + 4 + M \times (4 + 2 \times N)$ bytes. *La memòria per a emmagatzemar la llista no s'ha de reservar dins la biblioteca.*

LS_PUT Posa les dades especificades com a paràmetre d'entrada dins de la llista si hi ha espai. Retorna el punter a l'element on s'han introduit les dades o be $FFFFFFFF si no hi havia espai a la llista.

Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista
(A7+4)	N WORDs	Dades a introduir dins la llista

Post :

Posició	Tipus	Descripció
(A7)	LONG	Punter a l'element o $FFFFFFFF si no hi havia espai

Observacions : Es suposa que el punter a la llista apunta a una llista amb el format correcte.

LS_REMOVE Elimina de la llista l'element especificat, o be no fa res si l'element especificat no existeix.

Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista
(A7+4)	LONG	Punter a l'element a eliminar

Post : Cap

Observacions : Es suposa que el punter apunta a un element de la llista. No és necessari que faceu cap comprovació de si el punter a l'element és correcte o no.

LS_FIRST Retorna el punter al primer element ocupat de la llista, o be $FFFFFFFF si la llista és buida.

Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista

Post :

Posició	Tipus	Descripció
(A7)	LONG	Punter al primer element ocupat o be $FFFFFFFF si la llista és buida

Observacions : Es suposa que el punter a la llista apunta a una llista amb el format correcte

LS_NEXT Donat el punter a un element de la llista, retorna el punter al següent element dins la llista o bé $FFFFFFFF si l'element indicat és el darrer de la llista.

Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista
(A7+4)	LONG	Punter a l'element actual

Post :

Posició	Tipus	Descripció
(A7+4)	LONG	Punter al següent element ocupat o be $FFFFFFFF si no hi ha més elements

Observacions : Es suposa que el punter a la llista apunta a una llista amb el format correcte i que el punter a l'element actual apunta adequadament a un element.

LS_PREVIOUS Donat el punter a un element de la llista, retorna el punter al l'element que el precedeix dins la llista o bé $FFFFFFFF si l'element indicat és el primer de la llista.

Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista
(A7+4)	LONG	Punter a l'element actual

Post :

Posició	Tipus	Descripció
(A7+4)	LONG	Punter a l'element anterior a l'indicat o be $FFFFFFFF si l'element indicat és el primer

Post : Es suposa que el punter a la llista apunta a una llista amb el format correcte i que el punter a l'element actual apunta adequadament a un element.

LS_COUNT Retorna el número d'elements de la llista.

Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista

Post :

Posició	Tipus	Descripció
(A7)	WORD	Número d'elements de la llista

Observacions : Es suposa que el punter a la llista apunta a una llista amb el format correcte. Notau que el paràmetre de sortida només sobreescriu parcialment el paràmetre d'entrada.

LS_GET_ITEM Retorna el punter al n-èssim element de la llista, a on n és un paràmetre d'entrada de la subrutina, essent $n=0$ el primer element de la llista.

Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista
(A7+4)	LONG	Posició de l'element (n).

Post :

Posició	Tipus	Descripció
(A7+4)	LONG	Punter a l'element

Observacions : Es suposa que el punter a la llista apunta a una llista amb el format correcte.

4 Biblioteca d'interfície gràfica

A continuació es descriuen les subrutines de la biblioteca d'interfície gràfica que heu d'implementar. Per a cada una de les subrutines s'indica com han d'estar els paràmetres d'entrada dins la pila *abans* de cridar a la subrutina (Pre) i com han d'estar els paràmetres de sortida, si n'hi ha, *després de retornar* de la subrutina (Post). Recordau que dins la subrutina, la pila contindrà també l'adreça de retorn. *Aquest és el motiu de que les especificacions de l'enunciat i les que apareixen als comentaris del codi difereixin en 4 bytes.*

Sempre que necessiteu accedir a una llista dins d'aquesta biblioteca ho heu de fer cridant a les subrutines de la bibliteca de gestió de llistes. No heu d'accedir a la llista directament, únicament ho heu de fer amb les subrutines de l'esmentada biblioteca.

LP_INSTALL Instal·la la interfície gràfica. Això implica instal·lar adequadament la ISR del ratolí i inicialitzar el mode gràfic a 640x480, mode finestra i doble buffer. A més a més, s'ha d'inicialitzar a zero la variable (LP_ITEM_LAST) i a $FFFFFFFF la variable (LP_ITEM_SEL). Aquestes variables s'utilitzarà posteriorment.

Pre : Cap

Post : Cap.

Observacions : Addicionalment s'haurà de programar la ISR de gestió del mouse, que s'anomenarà LP_ISR_MOUSE_MOVE. La ISR s'haurà d'instal·lar de manera que s'executi quan el mouse es mogui o quan es faci click amb el botó esquerre. Aquesta ISR únicament haurà d'emmagatzemar les coordenades del mouse als WORDs LP_MOUSE_CX

i LP_MOUSE_CY, i també un $00 o un $01 al BYTE LP_MOUSE_BUT si el botó del mouse està amollat o pitjat respectívament.

LP_PLOT_UI Realitza el dibuixat de la interfície gràfica. No s'encarrega de cap interacció amb l'usuari, únicament dibuixa la interfície. En particular:

- Dibuixa els elements de la llista. És a dir, els de la sub-llista OCCUPIED. El dibuixat començarà a la coordenada X=0, Y=LP_INITIAL_Y. Cada element de la llista es dibuixarà com un rectangle blau de contorn blanc, d'amplada LP_ITEM_WIDTH i alçada LP_ITEM_HEIGHT. Cada element es dibuixarà a la dreta de l'anterior. Quan s'arribi a l'extrem dret de la pantalla, es començarà una nova fila. No us heu de preocupar si el número d'elements és tan gran que es sobrepassa la part d'abaix de la pantalla. Els valors esmentats (LP_INITIAL_Y, LP_ITEM_WIDTH i LP_ITEM_HEIGHT) són constants. El vostre codi ha de funcionar correctament per valors arbitraris d'aquestes constants. La única restricció que podeu considerar és que (LP_SCREEN_WIDTH / LP_ITEM_WIDTH) sigui un número sencer.
- Dibuixa un rectangle de contorn blanc i fons vermell a les coordenades LP_NEW_X i LP_NEW_Y amb amplada i alçada LP_NEW_WIDTH i LP_NEW_HEIGHT. A aquest rectangle l'anomenarem *Botó de Nou Element*. Els valors esmentats (LP_NEW_X, LP_NEW_Y, LP_NEW_WIDTH i LP_NEW_HEIGHT) són constants, però el vostre codi ha de funcionar correctament amb valors arbitraris per a aquestes constants.
- Imprimeix la cadena de text ITEM COUNT: X a l'extrem superior esquerre de la pantalla, a on X és el número, en base 10, d'elements de la llista (és a dir, de la sub-llista OCCUPIED).
- Imprimeix els valors continguts dins del camp de dades de l'element apuntat per (LP_ITEM_SEL). Els valors s'imprimeixen centrats horitzontalment, començant a la part superior de la pantalla amb un element per línia. Cada valor es mostrarà en base 10 ocupant exactament 6 caràcters. En el cas que (LP_ITEM_SEL) contengui $FFFFFFFF, no s'ha d'imprimir res. No vos heu de preocupar pel cas que els nodes contenguin tants elements que la impressió sobreescrigui altres elements de la interfície.
- Abans de començar a dibuixar res, aquesta subrutina ha de borrar la pantalla i, en acabar, ha d'actualitzar el doble buffer. És a dir, a excepció de la inicialització del doble buffer, que es du a terme a LP_INSTALL, la gestió del doble buffer es du a terme aquí.

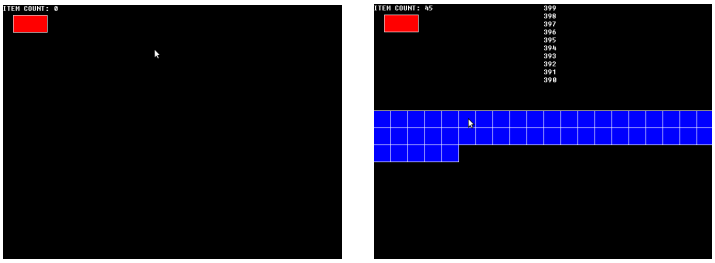
Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista a dibuixar

Post : Cap.

LP_UPDATE_UI Interactua amb l'usuari, però no dibuixa res per pantalla. En particular:

- Comprova si s'ha fet click al mouse. Un click al mouse és un canvi de no pitjat a pitjat. Per a fer això s'haurà d'accedir a la variable que actualitza la ISR del mouse. També es pot emprar la variable auxiliar LP_MOUSE_PRE.
- Si s'ha fet click dins del *Botó de Nou Element*, s'afegeix un nou element a la llista. Les dades de l'element que s'afegeixi seran valors numèrics consecutius, començant pel darrer valor introduït en insercions anteriors, o bé 0 si és la primera. Per a recordar el darrer valor introduït s'emprarà la variable (LP_ITEM_LAST).
- Si s'ha fet click sobre un dels elements de la llista, aquest es borrarà de la llista. Notau que per a aconseguir això, haureu d'esser capaços de determinar sobre quin element s'ha fet click.
- Tant si es fa click com si no, s'ha de posar el punter a l'element de la llista sobre el que es troba el mouse dins de (LP_ITEM_SEL). Si el mouse no es troba sobre cap element, es posarà $FFFFFFFF en l'esmentada posició de memòria.



(a) (b)

Figura 3: Exemples d'operativa de la interfície gràfica

Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista

Post : Cap.

La Figura 3 mostra dos exemples de funcionament de la interfície gràfica. A la Figura 3-a es mostra l'aspecte quan la llista és buida i a la Figura 3-b es mostra l'aspecte quan la llista conté alguns elements. Així mateix, podeu veure l'aplicació en funcionament al següent vídeo a YouTube:

`http://http://youtu.be/iABhMnjOBvY`

5 Tasques addicionals pels grups de 3 persones

Els grups de 3 persones heu d'implementar les següents funcionalitats addicionals. A la biblioteca de gestió de llistes:

LS_SAVE Guarda a disc els elements d'una llista. Podeu decidir el format de l'arxiu a disc. Consultau les funcions de File I/O del TRAP #15. Pel que fa al nom de l'arxiu o ruta completa de l'arxiu, recordau que Easy68K treballa sobre Windows i, per tant, s'empren les convencions pròpies de Windows. Ara bé, aquesta subrutina *no* ha de mostrar cap quadre de diàleg. Només s'encarrega de guardar.

Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista
(A7+4)	LONG	Punter a la cadena de text (null-terminated) amb el nom de l'arxiu o la ruta completa de l'arxiu. L'extensió de l'arxiu ha d'esser .EC2

Post : Llista guardada en un arxiu.

LS_LOAD Carrega els elements d'una llista des de disc. El format de l'arxiu a disc ha d'esser el mateix que hageu emprat a LS_SAVE. És a dir, la vostra rutina de carrega ha de poder obrir arxius generats amb la vostra rutina de guardat. Consultau les funcions de File I/O del TRAP #15. Pel que fa al nom de l'arxiu o ruta completa de l'arxiu, recordau que Easy68K treballa sobre Windows i, per tant, s'empren les convencions pròpies de Windows. Ara bé, aquesta subrutina *no* ha de mostrar cap quadre de diàleg. Només s'encarrega de carregar. A més a més, no és responsabilitat vostra garantir que la llista que es carregui capi a memòria. Si l'arxiu guardat conté una llista més extensa que la memòria reservada, és admissible que el programa es pengi.

Pre :

Posició	Tipus	Descripció
(A7)	LONG	Punter a la llista
(A7+4)	LONG	Punter a la cadena de text (null-terminated) amb el nom de l'arxiu o la ruta completa de l'arxiu. L'extensió de l'arxiu ha d'esser .EC2

Post : Llista carregada.

A la biblioteca d'interfície gràfica, dibuixau dos nous rectangles d'amplada i alçada idèntics a la del Botó de Nou Element, just a la dreta de l'esmentat botó. Els dos rectangles han d'esser verds amb contorn blanc. El primer ha de contenir el text LOAD i el segon SAVE. Quan es faci click sobre el primer botó, s'ha d'obrir el quadre de selecció d'arxiu i, quan es cliqui a OK, l'arxiu seleccionat s'ha de carregar com una llista. La llista que ja existís s'ha de borrar. El funcionament del segon botó és semblant al del primer, però ha de guardar. En qualsevol dels dos casos, si en comptes d'OK es clica CANCELAR al quadre de diàleg, no s'ha de fer res, i el programa ha de continuar amb l'operativa normal.

En aquest cas, no és responsabilitat vostra detectar si l'arxiu seleccionat té el format correcte. És a dir, si carregau un arxiu que no és una llista guardada per vosaltres mateixos, és admissible que el programa es pengi.

Per tal d'implementar aquestes funcionalitats *extra*, podeu cridar, si ho creis convenient, a les anteriors funcions de la biblioteca de llistes. Ara bé, les anteriors funcions de la biblioteca de llistes NO poden fer cridades a les *extra*.

6 Material subministrat

Se us proporcionen els següents quatre arxius:

screen.x68 : Conté algunes MACROs que vos poden facilitar l'accés als gràfics i l'escriptura de text. Podeu emprar-les lliurement, i podeu ampliar aquest fitxer tant com necessiteu, però no podeu llevar o modificar les MACROS que hi ha.

list.x68 : Conté algunes de les subrutines de biblioteca de gestió de llistes. Heu de completar les subrutines que falten, afegint-les a aquest fitxer. És a dir, *totes les subrutines de gestió de llistes* especificades a la Secció 3 han d'estar en aquest arxiu. També hi podeu posar subrutines auxiliars si les necessiteu, sempre i quan totes les etiquetes globals dins d'aquest arxiu comencin amb LS_. En cap cas, però, podeu modificar el codi existent.

Veureu que tot el codi està degudament comentat. S'espera el mateix del vostre codi. També veureu les capçaleres de les subrutines que falten, amb comentaris que les descriuen. Notau que a les capçaleres, el Pre i el Post defineixen l'estat de la pila al principi i al final de la subrutina respectivament, no abans i després de la cridada com s'ha fet a l'enunciat. Això suposa una diferència de 4 bytes, però notau que *són equivalents*.

list_plot.x68 : Dins aquest arxiu heu d'implementar les subrutines de la biblioteca de gestió de la interfície gràfica. És a dir, *totes les subrutines de gestió de la interfície gràfica* especificades a la Secció 4 han d'estar en aquest arxiu. També hi podeu posar subrutines auxiliars si les necessiteu, sempre i quan totes les etiquetes globals dins d'aquest arxiu comencin amb LP_. En cap cas, però, podeu modificar el codi existent.

Com abans, el codi està degudament comentat i l'especificació del Pre i el Post a les capçaleres difereixen de les donades en aquest enunciat tot i que *són equivalents*.

list_test.x68 : És el codi del programa principal. Aquest codi defineix algunes constants, inclou els arxius anteriors i executa una prova de funcionament. Quan les llibreries de biblioteca estiguin finalitzades, l'execució de list_test.x68 hauria de mostrar per pantalla una imatge semblant a la Figura 3-a i hauria de permetre interactuar amb la llista de la manera descrita.

Tot i que aquest programa configura el tamany de la llista i el número de WORDs de dades a uns determinats valors, les vostres biblioteques han de funcionar correctament amb llistes d'un tamany arbitrari amb un número arbitrari de WORDs per element. En particular, canviant els valors de les constants TST_LIST_ITEM_SIZE o TST_LIST_LIST_SIZE el programa hauria de funcionar perfectament.

A la versió que entregueu de la pràctica, el fitxer list_test.x68 *no ha d'estar modificat*, ha d'esser el mateix que se us subministra.

Abans de començar a programar heu d'entendre el funcionament de les subrutines proporcionades. Vos ajudarà a comprendre millor l'estructura de la llista, el que han de fer la resta de subrutines i vos donaran alguna idea sobre com programar la pràctica.

7 Material a entregar

L'entrega es farà exclussívament en format electrònic a través de Campus Extens. L'entrega es farà en un únic ZIP o RAR amb tots els arxius a l'arrel, sense carpetes. És imprescindible que el nom de l'arxiu ZIP que entregueu sigui *pr1_grupXX.ZIP* o *pr1_grupXX.RAR*, a on XX és el número del vostre grup de pràctiques. Només s'entregarà un ZIP per grup.

L'entrega es farà dins del termini establert. Cada dia de retràs en l'entrega suposarà una penalització de 0.5 punts sobre la nota de la pràctica.

L'arxiu ZIP o RAR contindrà *única i exclussívament* els següents arxius:

screen.x68 : L'arxiu que se us ha subministrat. El podeu haver ampliat, però no podeu haver llevat MACROs ni modificat les existents.

list.x68 : L'arxiu que se us ha subministrat, amb les funcions que hi falten i que haureu programat voltros. També pot haver-hi funcions auxiliars si les necessiteu, però no podeu haver modificat el codi proporcionat.

list_plot.x68 : L'arxiu que se us ha subministrat amb el vostre codi de les funcions de gestió de la interfície gràfica. També pot haver-hi funcions auxiliars si les necessiteu, però no podeu haver modificat les definicions de constants i variables proporcionades.

list_test.x68 : L'arxiu que se us ha subministrat, sense cap modificació.

document.pdf : La documentació de la pràctica. Ha de tenir una extensió màxima de 10 pàgines amb uns marges i tamanys de lletra raonables, més portada i/o índex. Com a mínim s'hi ha d'explicar el funcionament de les funcions que hageu implementat i s'ha de mostrar algun exemple de funcionament.

8 Avaluació

La pràctica es puntua de 0 a 10. En la puntuació es tindran en compte molt especialment els següents aspectes:

- Compliment de les especificacions. Entre d'altres, es faran les següents proves:
  1. Execució del codi que heu entregat (s'executarà list_test.x68). Si no funciona correctament, s'aplicarà una penalització d'entre 6 i 10 punts depenent de l'error.
  2. Execució de la vostra interfície gràfica amb una biblioteca de gestió de llistes correcta i distinta a la vostra. Si no funciona correctament significarà que la vostra interfície gràfica no compleix les especificacions i s'aplicarà una penalització d'entre 3 i 7 punts depenent de l'error.
  3. Prova de la vostra biblioteca de gestió de llistes amb una interfície gràfica correcta i distinta a la vostra. Si no funciona correctament significarà que la vostra biblioteca de gestió de llistes no funciona correctament i s'aplicarà una penalització d'entre 3 i 7 punts depenent de l'error.
  4. En tots els casos anteriors, les proves es faran amb distints tamanys de llista i de número de WORDs de dades. És a dir, es provaran distints valors de TST_LIST_ITEM_SIZE i TST_LIST_LIST_SIZE.
- Qualitat del codi. S'avaluarà:

1. Correcció. Entre d'altres, es comprovarà que els registres es restaurin adequadament i que tot registre o posició de memòria que s'empri estigui adequadament inicialitzat. També es comprovarà que les subrutines de gestió de la llista només modifiquin l'espai reservat per la llista.
2. Estructura. El codi ha d'estar ben estructurat, emprant subrutines sempre que sigui precís.
3. Claredat. El codi ha d'estar ben comentat. En particular, les subrutines auxiliars que programeu han d'incloure comentaris respecte als seus paràmetres d'entrada, de sortida i registres o posicions de memòria que modifiquin. Es penalitzarà tant la falta com l'excés de comentaris. Emprau els comentaris que apareixen al material subministrat com a exemple.
4. Velocitat d'execució. Aquest aspecte es valorarà en menor mesura, i únicament es penalitzaran aquells programes que presentin temps d'execució exageradament elevats.

- Documentació. S'avaluarà:

  1. Extensió. L'extensió màxima és de 10 pàgines (més portada i/o índex). Cada pàgina addicional suposarà una penalització de 0.5 punts. Si s'ha emprat una tipografia o uns marges massa petits per tal de no superar les 10 pàgines, també es penalitzarà. Es tracta de que demostreu la vostra capacitat de síntesi i que sapigueu centrar-vos en els conceptes clau, no de que demostreu que el vostre ordinador suporta Times New Roman de 3 punts.
  2. Correcció. Les explicacions del document han d'esser correctes i han de coincidir amb el codi entregat.
  3. Completitud. Les explicacions del document han de permetre entendre *tot* el que heu fet.
  4. Claredat i redacció. Les explicacions han d'esser clares. El document ha d'estar ben redactat i sense faltes d'ortografia o errors gramaticals. Un document excessívament mal redactat o amb un excés d'errors d'escriptura *pot suposar el suspens de la pràctica*.

- Còpia. No es tolerarà la còpia total o parcial de la pràctica. Qualsevol còpia, total o parcial, de la pràctica suposarà el *suspens* de *tota* l'assignatura i la impossibilitat de presentar-se a cap examen de l'assignatura ni en el període ordinari ni en el de recuperació d'aquest any acadèmic.