Exemple disseny descendent

Enunciat

1. Escriu un programa en C que gestioni un menú d'opcions. Es mostrarà la llista d'opcions disponibles, identificades amb un valor numèric i es demanarà a l'usuari que introdueixi el valor numèric de la opció que desitgi. Tingues en compte els següents requeriments:
   a) La primera opció té l'identificador 1.
   b) La última opció permet sortir de l'aplicació.
   c) En seleccionar una opció diferent a la de sortir, es mostrarà un missatge indicant que s'ha seleccionat l'acció corresponent i es tornarà a llistar la llista d'opcions.
   d) En seleccionar una opció incorrecta, es mostrarà l'error corresponent, i es demanarà un altre cop que es triï una opció, sense tornar a mostrar la llista d'opcions.
   e) Les opcions disponibles són: "Afegir fitxer", "Eliminar fitxer" i "Llistar fitxers". (i la de sortir)

A mesura que anem agafant pràctica programant, podem inferir molts aspectes de la nostra solució a partir de l'enunciat. A continuació anirem construint la solució pas a pas, ressaltant aquelles coses que ens puguin ajudar:

- *Es demana un valor numèric a l'usuari amb la opció desitjada. Això ja ens ha d'indicar que tindrem una variable entera numèrica, la qual està dins d'un domini acotat [1,..,N], on N és el nombre d'opcions del menú.*
- *L'aplicació té un comportament seqüencial. Anirem demanant opcions fins que l'usuari decideixi sortir de l'aplicació. Per tant, sabem que tindrem una composició iterativa. A més a més, si utilitzem els coneixements sobre composicions iteratives, ho podem veure com una seqüència de valors acabada amb la opció de sortir, això és una estructura de tipus recorregut, i la podem implementar com a tal.*

Exemple disseny descendent

Solució:

- *Es demana un valor numèric a l'usuari amb la opció desitjada. Això ja ens ha d'indicar que tindrem una variable entera numèrica, la qual està dins d'un domini acotat [1,..,N], on N és el nombre d'opcions del menú.*
  - *Quan tenim una variable dins d'un domini acotat, s'ha de verificar sempre que el valor que se li assigna és correcte i dins d'aquest domini.*
    - *Cal verificar que s'ha entrat un valor numèric.*
    - *Cal verificar que el valor numèric entrat és major o igual a 1 i menor o igual a N.*
  - *Si el domini és discret i estàtic, o sigui, que tenim un conjunt d'etiquetes que no canvia amb el temps, podem pensar en definir constants o un tipus de dades Enumerat, que representi els diferents valors.*
    - *En aquest cas tenim quatre opcions definides en el propi enunciat, per tant podem definir l'Enumerat, assignant a cada valor numèric una constant.*

```
typedef   enum {
        OPT_ADD_FILE=1,
        OPT_DEL_FILE=2,
        OPT_LIST_FILES=3,
        OPT_EXIT=4
} opcionsMenu;
```

Exemple disseny descendent

Solució:

- *L'aplicació té un comportament seqüencial. Anirem demanant opcions fins que l'usuari decideixi sortir de l'aplicació. Per tant, sabem que tindrem una composició iterativa. A més a més, si utilitzem els coneixements sobre composicions iteratives, ho podem veure com una seqüència de valors acabada amb la opció de sortir, això és un esquema de tipus recorregut aplicat a la entrada, i ho podem implementar com a tal.*

```
algorisme recorregutEntrada
        var elem: T fvar
        inici tractament
        elem:=llegir();
        mentre no (elem = marca) fer
                tractar element (elem)
                elem := llegir();
        fmentre
        tractament final
falgorisme
```

T serà un tipus numèric sense signe.

llegir() serà la funció de lectura d'elements . Podem utilitzar la estàndard per llegir enters, *llegirEnter()*, o definir-ne una de nostra que permeti controlar el domini dels valors llegits, per exemple *llegirOpcio()*.

En l'enunciat no es detalla cap tractament final. El tractament inicial pot ser mostrar les opcions i el tractament de cada element és mostrar l'acció seleccionada.

Exemple disseny descendent

Solució:

- *Una manera ràpida de començar a resoldre un problema de forma modular, aplicant el concepte de disseny descendent de forma intuïtiva, és escriure el "main" utilitzant funcions/accions per a tots els passos que necessitem, utilitzant comentaris per explicar què faran i després anar fent el mateix per cada acció/funció que ens hagi aparegut, fins a que arribem a accions/funcions trivials. Per exemple:*

```
int main(void) {
        /*Definim la variable on es guardarà la opció. El tipus dependrà de les opcions que tinguem. Pot ser un
        enumerat o un enter.*/
        opcionsMenu opcio;

        /* Mostra el menú d'opcions i demana a l'usuari que en triï una. En cas que la entrada sigui incorrecta, continua
        demanant una opció fins que el valor entrat sigui correcte.*/
        opcio=llegirOpcio();

        /* Anem analitzant les opcions triades fins que es demani sortir. Definirem la marca de sortida amb la constant
        OPT_EXIT. */
        while (opcio!= OPT_EXIT) {
                /* Fem les accions definides per a cada opció*/
                ferAccions(opcio);
                /* Demanem una altra opció. */
                opcio=llegirOpcio();
        }
        /* Finalitzem la execució del programa*/
        return EXIT_SUCCESS;
}
```

Exercici1.c

Exemple disseny descendent

Solució:

- *Els tipus de dades i les accions/funcions que anem utilitzant, les anirem declarant en el fitxer de capçaleres (*.h). Junt amb la declaració de cada acció/funció, podem mantenir la descripció del què fa. Per exemple:*

```
/*Definim el tipus de dades utilitzat per a guardar les opcions del menú. Per exemple, amb l'enumerat, comentant cada opció.*/
typedef enum {
        OPT_ADD_FILE=1, /*Opció per afegir un fitxer.*/
        OPT_DEL_FILE=2, /*Opció per eliminar un fitxer.*/
        OPT_LIST_FILES=3, /* Opció per llistar els fitxers.*/
        OPT_EXIT=4 /* Opció de sortir de l'aplicació. */
} opcionsMenu;

/* Mètode principal. S'encarrega de la lògica de l'aplicació. */
int main(void);

/* Mostra el menú d'opcions i demana a l'usuari que en triï una. En cas que la entrada sigui incorrecta, continua demanant una opció fins que el valor entrat sigui correcte.*/
opcionsMenu llegirOpcio(void);

/* Executa les accions definides per a cada opció*/
void ferAccions(opcionsMenu opcio);
```

Exercici1.h

Exemple disseny descendent

Solució:

- *Importem els fitxers de capçalera necessaris per a que funcioni tot. Aquesta tasca es fa amb la directiva "include", a la qual se li passa el fitxer corresponent. En el nostre cas, importarem dos fitxers estàndard i el que hem creat amb les nostres definicions. Fixeu-vos que es fa de forma lleugerament diferent. La diferència és que en utilitzar "cometes", li diem que comenci a buscar pel directori/carpeta on es troba el fitxer de codi, i no pels directoris/carpetes estàndard.*

```
#include <stdio.h>
#include <stdlib.h>
#include "Exercici1.h"

int main(void) {
  ....
        return EXIT_SUCCESS;
}
```

Exercici1.c

stdio.h conté les declaracions dels mètodes d"entrada/sortida", que són els que permeten escriure i llegir dels dispositius estàndards (pantalla i teclat o fitxers).

stdlib.h conté les declaracions dels mètodes bàsics, així com declaracions de constants de sistema, com per exemple la que indica que s'ha acabat correctament (EXIT_SUCCESS).

Exercici1.h és el fitxer que hem creat nosaltres. Fixeu-vos que aquest l'incloem utilitzant cometes en comptes dels signes de desigualtat < i >.

Exemple disseny descendent

Solució:

- *A més a més, ara afegirem al fitxer de codi les implementacions de les accions i funcions que hem declarat, afegint un missatge que indiqui que encara no estan fetes, però retornant algun valor per defecte que permeti compilar i executar l'aplicació. També és útil deixar la descripció del que ha de fer, amb una indicació TODO (a fer en l'anglès), indicant que encara s'ha d'implementar.*

```
int main(void) {
   ....
}

opcionsMenu llegirOpcio(void) {
        /* Mostra el menú d'opcions i demana a l'usuari que en triï una. En cas que la entrada sigui
         incorrecta, continua demanant una opció fins que el valor entrat sigui correcte.*/
        printf("TODO: Funció llegirOpcio\n");

        return OPT_EXIT;
}

void ferAccions(opcionsMenu opcio) {
       /* Executa les accions definides per a cada opció*/
       printf("TODO: Acció ferAccions\n");
}
```

Exercici1.c

Exemple disseny descendent

Solució:

- *En aquest moment el nostre programa ja es pot compilar i executar, assegurant que la estructura/esquelet de la aplicació és correcte. Com és normal, no funcionarà, ja que no hem implementat res, però ens permetrà comprovar que hem declarat correctament tot el què necessitem. La sortida que hauríem de tenir és la següent:*

```
>> Exercici1
        TODO: Funció llegirOpcio
>>
```

- *Donat que la opció per defecte que hem posat és la de sortir, mai es cridarà a l'acció per tractar les opcions. Anem a implementar la funció per llegir la opció. Al igual que hem fet en el cas del main, indicarem els passos que hem d'anar fent, però ara només utilitzarem comentaris, per decidir si declarem una nova acció/funció per resoldre'ls o ho fem dins aquesta mateixa.*

```
opcionsMenu llegirOpcio(void) {

        /* Mostrem la llista d'opcions*/
        /* Anar demanant opcions a l'usuari fins que n'entri una de correcta.*/
        /* Convertir l'entrada per teclat al valor de retorn desitjat. */
}
```

Exercici1.c

Exemple disseny descendent

Solució:

- *Tot i que mostrar la llista de les opcions ho podem fer directament, ja que simplement és mostrar cadenes de text per pantalla, donat que potser ens interessarà posar-hi un títol o potser fer alguna altra acció prèvia, com ara netejar la pantalla abans de mostrar el menú, o el que sigui, ho posarem en una acció a part (mostrarOpcions).*
- *Demanar una opció a l'usuari i validar que sigui correcta, inclou conversions de tipus i una composició iterativa, ja que hem d'analitzar totes les opcions entrades fins que trobem un valor correcte. Al igual que en el cas anterior, podríem posar-ho en el mateix cos de la funció, però crearem una funció auxiliar, que s'encarregui de llegir un enter en el rang [a,b]. En cas que el valor entrat sigui incorrecte, retornarà un valor (a-1), que com està fora del domini, ens servirà de valor d'error. (getDomainValue).*
- *La conversió de tipus la farem en el propi mètode, sabent que hem assignat a les constants de l'enumerat els valors que els hi toca en el menú. En cas contrari hauríem de fer la conversió de forma manual (varis blocs if-else o un bloc switch).*

Exemple disseny descendent

Solució:

- *Amb els canvis proposats, el mètode llegirOpcio quedaria de la següent manera:*

```
opcionsMenu llegirOpcio(void) {
        int opcio=-1;
        opcionsMenu opcioRet;

        /* Mostrem la llista d'opcions*/
        mostrarOpcions();

        /* Anar demanant opcions a l'usuari fins que n'entri una de correcta.*/
        printf("Entra la opció desitjada: ");
        opcio=getDomainValue(1,4);
        while(opcio<1) {
                printf("Entra la opció desitjada: ");
                opcio=getDomainValue(1,4);
        }

        /* Convertir l'entrada per teclat al valor de retorn desitjat. */
        opcioRet=(opcionsMenu)opcio;

        return opcioRet;
}
```

Exercici1.c

Exemple disseny descendent

Solució:

- *També caldrà afegir les declaracions i implementacions dels mètodes auxiliars:*

```
/* Mostrem la llista d'opcions*/
void mostrarOpcions(void);

/* Llegeix un valor enter per teclat, i valida que estigui en el rang [valMin,valMax].
En cas contrari retorna valMin-1*/
int getDomainValue(int valMin,int valMax);
```

Exercici1.h

```
void mostrarOpcions(void) {
    /* Mostrem la llista d'opcions*/
    printf("TODO: Acció mostrarOpcions\n");
}

int getDomainValue(int valMin,int valMax) {
    /* Llegeix un valor enter per teclat, i valida que estigui en el rang [valMin,valMax].
    En cas contrari retorna valMin-1*/
    printf("TODO: Funció getDomainValue\n");

    return valMin-1;
}
```

Exercici1.c

Exemple disseny descendent

Solució:

- *Definim el mètode mostrarOpcions per tal que mostri un títol i la llista d'opcions descrita a l'enunciat:*

```
void mostrarOpcions(void) {
    /* Mostrem la llista d'opcions*/
    printf("============================\n");
    printf("====          MENU          ====\n");
    printf("============================\n");
    printf("\n");
    printf("%d.- %s\n", 1 , "Afegir Fitxer");
    printf("%d.- %s\n", 2 , "Eliminar Fitxer");
    printf("%d.- %s\n", 3 , "Llistar Fitxers");
    printf("%d.- %s\n", 4 , "Sortir");
}
```

Exercici1.c

Exemple disseny descendent

Solució:

- *Finalment, en la següent i última iteració d'aquest procés de desenvolupament, implementaríem el mètode getDomainValue, on s'ha decidit no afegir cap mètode auxiliar:*

```
int getDomainValue(int valMin,int valMax) {
    int retVal;

   /* Llegeix un valor enter per teclat, i valida que estigui en el rang [valMin,valMax].
   En cas contrari retorna valMin-1*/

   /* Demanem la opció */
   printf("Entra  una opcio: ");
   scanf(" %d",&retVal);

   /* Comprovem el valor entrat */
   if(retVal<valMin || retVal>valMax) {
        retVal=valMin-1;
   }

   return retVal;
}
```

Exercici1.c