C PROGRAMAZIO–LENGOAIA (XIV)

Aplikazioak I

Iñaki Alegria & Montse Maritxalar

KAPITULU honetan aurreko kapituluetan aztertutako kontzeptu gehienak praktikan jarri nahi ditugu eta horretarako aplikazio bat programatuko dugu; liburutegi txiki baten kudeaketa hain zuzen.

Interesatzen zaigun aurreneko erabakia erregistroari buruzkoa da, hau da, liburu bakoitzari buruz mantenduko dugun informazioari buruzkoa. Pentsa dezagun ondoko informazioa gordeko dela: liburu-kodea, izenburua, idazlea, urtea eta liburu-motaren kodea. Erregistro hau funtzio gehienetan erabiliko denez, **liburu.h** fitxategian erregistroaren definizioa egingo dugu (ikus 1. programa).

```
/* liburu. h*/
struct liburu
     {
       int kod;
       char izen[40], idazle[40];
       int urte, mota;
     };
```

1. programa. Erregistroaren definizioa.

Erregistro honekin oinarrizko bi eragiketa egingo dira: informazioa teklatutik irakurtzea eta pantailan idaztea. Funtzio hauek aparte konpilatuko ditugu. Beraz fitxategi batean kokatuko dira; **lib_oinarri.c** izeneko fitxategian gure kasuan, 2. programan ikusten den bezala.

```
/* lib_oinarri.c — erregistro baten irakurketa eta idazketa */
# include "liburu.h"
void garbi_pantaila ( ); /* geroago definituta. Erazagupena */
void irakur (plib)
      struct liburu * plib;
      {
        int aux1;
        char aux2[4Ø];
        garbi_pantaila ( );
        printf ("liburuaren kodea: \n");
        scanf ("%d", &aux1);
        plib –> kod = aux1;
        printf ("\n liburuaren izenburua: \n");
        scanf ("%s", aux2);
        strcpy (plib  –> izen, aux2);
        printf ("in liburuaren idazlea: \n");
        scanf ("%s", aux2);
        strcpy (plib  –> idazle, aux2);
        printf ("\n liburuaren urtea: \n");
        scanf ("%d", & aux1);
        plib –> urte = aux1;
        printf ("\n liburu-motaren kodea: \n");
        scanf ("%d", & aux1);
        plib –> mota = aux1;
      }
void idatz (plib);
      struct liburu * plib;
      {  char c[2];
        garbi-pantaila ( );
        printf ("liburuaren kodea: % d\n" plib –> kod);
        printf ("liburuaren izenburua: % s\n", plib  –> izen);
        printf ("liburuaren idazlea: % s\n", plib  –> idazle);
        printf ("liburuaren urtea: % d\n", plib  –> urte);
        printf ("liburu-mota: % d\n", plib  –> mota);
        printf ("jarraitzeko sakatu edozein tekla!");
        scanf ("%c", &c [Ø];
      }
void garbi_pantaila ( )
      {  int i;
        for (i = Ø; i < 24; i++) printf ("\n");
      }
```

2. programa. Oinarrizko funtzioak.

Pantailaren maneiua hobe daiteke. Horretarako sistema berrietan leiho-sistema (windows) bat erabiltzen da datuak teklatu zein saguaren (mouse)

bidez sar daitezkeelarik. Zoritxarrez sistema hauek programatzeko liburutegiak ez dira estandarrak eta horrexegatik ez ditugu hemen azaldu.

Pentsa dezagun orain gure liburuei buruzko datu--basean zein eragiketa-mota egin daitezkeen. Hasiera batean gutxienez ondoko eragiketek egon beharko lukete: liburuen sarrera, liburuen listatua eta liburuen kontsulta. Liburuen kontsultari dagokionean ondoko erizpideak hartu beharko lirateke kontutan: kodea (horrela liburu bakarraren informazioa lortuz), idazlea, urtea (daten arteko tarteak onartuz) eta mota. Kontsulta hauek azkar egiteko bi aukera daude: batetik liburuen datu-basea txikia izan eta memoriara ekarri hasieran, horrela kontsultak azkarrago burutuz, eta bestetik, indizeak eraiki beste fitxategi batzuetan datu--basea eraiki ahala, eta kontsultak egitean indize hauek erabiliz datu-basea irakurri. Bigarren hau orokorragoa da, eta horrexegatik datu-base sistemetan erabiltzen da, baina artikulu-sail honen helburuetatik kanpo dago; oso programa-multzo konplexua gertatzen bait da. Beraz, lehenengoari ekingo diogu. Dena den, liburuaren kodea sekuentziala dela kontsideratuko dugu. Horrela kodearen araberako kontsulta berehalakoa izango da; irakurketa sekuentziala, memorian edo fitxategian, ez bait da egin beharko. Beraz, programa nagusiak egingo duena honako hau izango da: datu-basea duen fitxategia ireki, fitxategia memoriara ekarri, pantailan menu bat azaldu aukera desberdinak emanez, aukera bat irakurri eta aukeraren arabera funtzio berezitu bati deitu. 3. programan hori burutzen duen programa dugu.

```
/* nagusi.c — programa nagusia: menu */
# include <stdio.h>
# include "liburu.h"
# define MAX 1000
main (argc, argv)
        int argc;
        char * argv[ ]; /* argumentua datu-basearen izena */
        {  extern void sarrera ( ), listatu ( ), kon_kod ( ),
           kon_beste ( );
            extern void kon_beste ( );
           void irakur-osoa ( );
           FILE * fd;
           int auk;
           struct liburu lib[MAX];
        fd = fopen (argv[1], "r");
        irakur_osoa (fd, lib);
        while (1) /* etengabe jarraitu exit gertatu arte */
           garbi-pantaila ( );
           printf ("1- sarrera \n 2- listatua \n);
           printf ("3- kontsulta kodearen arabera \n);
           printf ("4- kontsulta idazlearen arabera \n);
           printf ("5- kontsulta dataren arabera \n);
           printf ("6- kontsulta motaren arabera \n);
           printf ("Ø- bukatu \n \n \n ZURE AUKERA:")
           scanf ("%d", &auk);
        switch (auk)
        {
           case Ø:  sarrera (lib, argv[1]);
                    break;
           case 2:  listatu (lib);
                    break;
           case 3:  kon_kod (lib);
                    break;
           case 4:
           case 5:
           case 6:  kon_beste (auk, lib);
                     /* hiru kasuetan */
                    break
           default: printf ("ERROR-aukera: 0-6");
                    scanf (" %c, &auk); /* edozer sakatu */
           }
        }
}
void irakur_osoa (fd, taula)
        FILE * fd;
        struct liburu taula [ ];
        {
           int i;
           while (fread (& taula [i],
              sizeof (struct liburu), 1, fd)! = EOF)
                 {
                    if (i == MAX) return;
                    i++;
                 }
           close (fd)
        }
```

3. programa. Programa nagusia.

Konturatu **sarrera**, **listatu, kon_kod** eta **kon_beste** funtzioak programa nagusian definitu gabe daudela, baina **extern** bezala erazagutu ditugula. Funtzio hauen definizioa 4 fitxategi desberdinetan egin daiteke, 4., 5., eta 6. programan azaltzen den bezala; listatu funtzio ez bait dugu azalduko. **garbi_pantaila** funtzioa ere **extern** bezala erazagutu da; **lib.oinarria** fitxategian bait dago. Beraz, ez da definitu behar. Ezta fitxategi horren **# include** egin behar ere; estekatzaileak (linker) lotura egingo bait du.

```
/* sarrera.c */
# include <stdio.h>
# include "liburu.h"
# define MAX 1000;
void sarrera (taula, izena)
        struct liburu taula[ ];
        char izena[ ];
        {
        FILE * fitx 2;
        struct liburu berri;
        extern void irakur ( );
```

```
void errore ( );
        irakur (&berri)
        if (berri.kod > = MAX)    errore (berri.kod);
        else {
        fitx2 = fopen (izena, "w")
        fseek (fitx2, berri.kod * sizeof (struct liburu), SEEK_SET);
        fwrite (& berri, sizeof (struct liburu), 1, fitx2);
        close (fitx2);
        taula [berri.kod] = berri; /* taulan sartu */
        }
}
void errore (kod)
    int kod;
    { char c[2];
      printf ("kodea 1000 baino txikiagoa, beraz ez %d \n", kod);
      scanf ("%c", &c[Ø]);
    }
```

4. programa. Liburuen sarrera.

```
/* kon_kod.c */
# include "liburu.h"
# define MAX 1000
void kon_kod (taula)
    struct liburu taula[ ];
    {
       int kod;
       extern void idatz ( ), errore ( );
       printf ("liburuaren kodea: \n");
       scanf ("%d", & kod);
       if (kod >= MAX) errore (kod);
       else
           if (tab [kod].kod == kod)
                       idatz (& tab[kod]);
    }
```

5. programa. Kontsulta kodearen arabera.

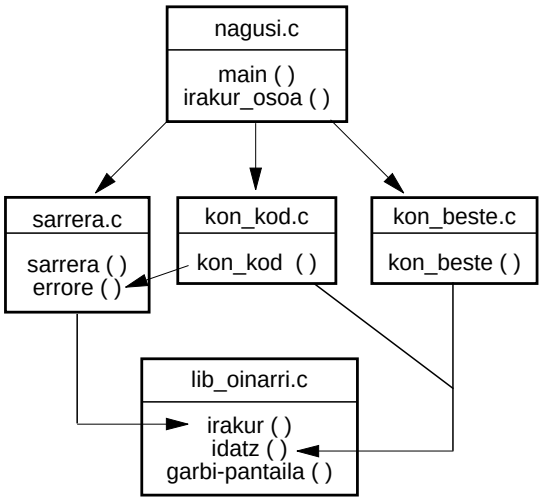
```
/* kon_beste.c */
# include "liburu.h"
# define MAX 1000
void kon_beste (taula, aukera)
    struct liburu taula[ ];
    int aukera;
    {
       int i, data1, data2, mota;
       char idazle[4Ø];
       extern void idatz ( )

       switch (aukera)
          {
             case 4:   printf ("idazlea: \n");
                       scanf ("%s", idazle);
                       break;
             case 6:   printf ("mota: \n");
                       scanf ("%d", &mota);
                       break;
             case 5:   printf ("epearen urte hasiera eta
                         bukaera: \n");
                       scanf ("%d %d, &data1, &data2);
                       break;
          }
       for (i = Ø; i < MAX; i++)
          if   (((aukera == 4) &&(strcmp (idazle,taula [i].
               idazle == Ø))
               ||((aukera == 6) &&(mota == taula[i]. idazle))
               ||((aukera)== 5) &&(data1>=taula [i]. urte) &
               &(data2<=taula[i].urte)))
               idatz (& taula[i]);
}
```

6. programa. Kontsultarako errutina orokorra.

Erroreen maneiua askoz ere konplexuago egin zitekeen, baina ahalik eta laburren egin dugu garrantzia ekintza nagusiei ematearren.

Beraz, 6 modulu ditugu: **liburu.h, lib.oinarri.c, nagusi.c, sarrera.c, kon_kod.c** eta **kon_beste.c. listatu.c** modulua ez da azaltzen idatz funtzioa antzekoa delako, baina inprimagailua maneiatuz eta **for** egitura batez kontrolaturik.

Modulu hauen dependentzia hierarkikoa lehen irudian azaltzen da. Bertan ez da agertzen **liburu.h** modulua izenburu-fitxategia delako eta beraz beste moduluetan # include sasiaginduaren bidez txertatzen delako.



1. irudia. Funtzioen arteko dependentziak.

Konpilatzeko, lehenbizi lau modulu eznagusiak banan-banan konpilatzen dira, dagozkien **.0** objektu--moduluak lortuz. Ondoren, programa nagusia konpilatu eta estekatu egiten da azken honetarako aurretik lortutako objektu-moduluen izenak aipatu egin behar delarik. Estekaketaren ondorioz moduluen arteko erreferentzi gurutzatuak ebaztu eta programa exekutagarri bakarra lortzen da.