# C PROGRAMAZIO–LENGOAIA (eta XV)

## Aplikazioak II

Iñaki Alegria & Montse Maritxalar

Kapitulu honetan erabilpen orokorreko funtzio--multzo bat definituko dugu, funtzio-multzo hau liburutegi batean gorde eta edozein aplikaziotatik erabil daitekeelarik.

Definituko dugun funtzio-multzoak listak maneiatzeko datu-mota berria osatuko du, eta aplikazio batek listak erabili nahi baditu, datu-mota berri horren bidez egin dezake C liburutegi estandarreko funtzioak balira bezala. Beraz, has gaitezen listak eta dagozkien funtzioak definitzen.

Lista bat bi erakuslek definituko dute, bata lehen osagaia adierazteko eta bestea azkena adierazteko izango direlarik. Listaren osagai bakoitzak badu erakusle bat listaren ondoko osagaia adierazteko, osagaiaren informazioaren erakuslea eta lehentasuna, zeren osagaiak lehentasunaren arabera txertatzeko aukera bait dago. 1. programan lista_def.h fitxategiaren edukina azaltzen da, bertan lista egin eta osagaia definitzen delarik.

```
struct elem {
        struct elem * ondoko;
        int lehentasuna;
        char * infor;
        }
struct lista {
     struct elem * lehena;
     struct elem * azkena;
     }
# define NIL (struct elem *) Ø /* lista hutserako */
# define BOOL int      /* funtzio boolearretarako */
# define TRUE 1
# define FALSE Ø
```

***1. programa.*** **lista_def.h** ***definizioak.***

Ondoren aplikazioek beharko dituzten listen gaineko funtzioak diseinatu eta definitu egin behar dira.

Diseinuaren aldetik, funtzioak zeintzuk izango diren zein motatakoak eta zein parametro izango dituzten da zehaztu behar dena. Hasteko, lista huts bat sortzeko funtzioa beharko dugu; **sortu** izenekoa parametro bakartzat listaren erakuslea izango duena. Lista hutsa den ala ez jakiteko, funtzio boolear bat izango dugu parametro bakarrarekin; **huts** funtzioa alegia. Listetan osagaiak erantsi eta kendu egiten direnez, hauexek izango dira funtzio nagusiak: bukaeran eranstea (lehentasuna kontutan hartu gabe) edo lehentasunaren arabera eranstea, eta lehen osagaia kentzea edo lehentasun jakin baten lehen osagaia kentzea. Lau funtzio hauei **lotu, txertatu, lehena** eta **atera** izenak dagozkie eta ondoko ezaugarriak egokituko dizkiegu:

- **lotu** eta **txertatu** eransteko funtzioak direnez, bi parametro izango dituzte, lista eta osagaia, (erakusleen bidez) eta ez dute baliorik itzuliko.

- **lehena** eta **atera** funtzioek osagai bat listatik kentzen dutenez, osagai horren erakuslea itzuliko dute, bietan parametro bakartzat listaren erakusle agertuz.

– **atera** funtzioan listaren gain beste parametro bat zehaztu behar da, lehentasuna hain zuzen; horretarako lehen osagaia kendu behar bait da listatik.

```
extern void sortu (struct lista *);
exter BOOL huts (struct lista * );
extern void lotu (struct lista *, struct elem *);
extern void txertatu (struct lista *, struct elem *);
extern struct elem * lehena (struct lista *);
extern struct elem * atera (struct lista *, int);
```

***2. programa. Funtzioen prototipoak.***

Diseinatutako funtzioak **lista_fun.h** fitxategian isladatuko dira; aplikazioetan fitxategi hau listen gaineko funtzioak erazagutzeko erabiliko bait da.

```
void sortu (plista)
     struct lista *plista;
{
     plista -> lehena = NIL;
     plista -> azkena = NIL;
}
```

***3. programa.* sortu *funtzioaren definizioa.***

```
BOOL huts (plista)
    struct lista *plista;
    {
    return (plista -> lehena = = NIL); /* balio boolearra
    }
```

***4. programa.* huts *funtzioaren definizioa.***

```
void lotu (plista, pelem)
      struct lista * plista;
      struct elem * pelem;
{     /* elementu bat listaren azkenari lotzen zaio */
      struct elem * paux;
      paux = plista -> azkena;
      if (paux = = NIL)  plista -> lehena = pelem;
                  else   paux -> ondoko = pelem;
      pelem -> ondoko = NIL;
      plista -> azkena = pelem;
}
```

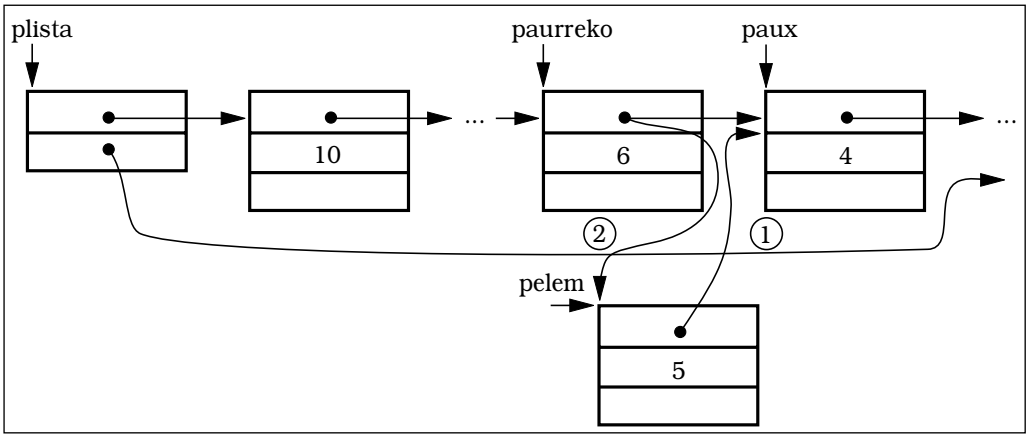
***5. programa.* lotu *funtzioaren definizioa.***

```
void txertatu (plista, pelem)
      struct lista * plista;
      struct elem * pelem;
{     /* elementu bat lehentasunaren arabera lotzen da */
      struct elem * paux, * paurreko;
      paurreko = NIL;
      for (paux = plista -> lehena; paux! = NIL;
                  paux = paux -> ondoko)
        {  /* lista korritzen da txertatu behar den tokia
           aurkitu arte */
           if (pelem -> lehentasuna <= paux -> lehentasuna)
                        paurreko = paux;
                  else break; /* hemen txertatu */
        }
      /* paurrekoa-n aurrekoa edo NIL lehena txertatzeko
         paux-n ondoko edo NIL azkena txertatzeko */
pelem -> ondoko = paux;
if (paurreko! = NIL)    paurreko -> ondoko = pelem;
          else          plista -> lehena = pelem;
if (paux = = NIL)  plista -> azkena = pelem;
```

***6. programa.* txertatu *funtzioaren definizioa.***

```
struct elem * lehena (plista)
      struct lista * plista;
   {
   struct elem * paux;
   paux = plista -> lehena;
   if (paux! = NIL)
      {
      plista -> lehena = paux -> ondoko;
      if (plista -> azkena = = paux) plista -> azkena = NIL;
      }
   return (paux);
}
```

***7. programa.* lehena *funtzioaren definizioa.***




***1. irudia.* txertatu *funtzioaren ondorioa.***

```
struct elem * atera (plista, lehent)
    struct lista * plista;
    int lehent;
{
    struct elem * paux, * paurreko;
    paurreko = NIL;
    for (paux = plista -> lehena; paux! = NIL;
                                        paux = paux -> ondoko)
      {
        if (paux -> lehentasuna > lehent)
             paurrerko = paux;
        if (paux -> lehentasuna = = lehent)
          { /* listatik kendu */
            if (aurreko! = NIL)  paurreko -> ondoko = paux -> ondoko;
                         else       plista -> lehena = paux -> ondoko;
            if (paux -> ondoko = = NIL) plista -> azkena = paurrekoa;
            return (paux);
          }
        if (paux -> lehentasuna < lehent) return (NIL);
      }
return (NIL)
```

***8. programa.* txertatu *funtzioaren definizioa.***

Ondoko programetan 3.etik 8.erarte funtzioen definizioari ekin diogu, definizioak denak fitxategi batean edo fitxategi bereiztuetan egon daitezkeelarik. Fitxategi bakoitzeko hasieran **#include lista_def.h** sententzia zehaztu beharko da.

Lehen irudian egoera arrunta isladatzen da, hau da, txertatzen den elementua lehena edo azkena ez denekoa.

**lotu** eta **txertatu** funtzioen alderantzizkoak **lehena** eta **atera** funtzioak dira; elementu bat lotu beharrean kendu egiten bait dute.

Funtzio hauek konpilatu ondoren, liburutegi batean gordetzea komeni da; horrela aplikazioetatik erreferentziatzen direnean estekatzeko prozedura erraztu egiten bait da. Aurreko kapituluko funtzioak ez ziren liburutegi batean kokatzen, aplikazio jakin baterako idatzita zeudelako, baina kapitulu honetako funtzioen helburua edozein aplikaziotarako datu-mota berria eskaintzea denez, liburutegi batean kokatzea da prozedura estandarra.

Objektu-moduluak liburutegietan gordetzeko sistema eragile bakoitzak komandoren bat dauka, 11. kapituluan esan genuenez.

Liburutegia osatuta edukiz gero, ikus dezagun liburutegian gordetako aipatutako funtzioak nola erabil dlaitezkeen.

Adibide batez azalduko dugu. Pentsa dezagun aurreko kapituluko datu-basean liburuak publikazio--dataren arabera listatu nahi ditugula. 9. programan azaltzen denaren arabera, lista bat definituko dugu eta listaren osagaiak memoria dinamikoan kokatuko dira. Ondokorako erakusleak, publikazio-datak lehentasun gisa eta dagokion libururako erakusleak osatzen dute listaren osagai edo elementu bakoitza. Liburu guztiak korrituz lista osatu ondoren, osagaiak ordenean atera eta inprimatuko dira.

Konturatu lista eta elementuak, orain arte memoria asignaturik ez zeukatenez, programa nagusian definitu direla, bata memoria estatikoan eta memoria dinamikoan besteak. Bestetik, listei dagozkien funtzioen erazagupena ez da egiten, **lista_fun.h** fitxategia txertaturik dago eta.

```
# include <stdio.h>
# include "lista_def.h"
# include "lista_fun.h"
# include "liburu.h"
# define MAX 1000
main (argc, argv)
      int argc;
      char * argv [ ]
    {
      extern void irakur_osoa ( );
      struct liburu lib [MAX];
      struct lista lis;
      struct elem *p;
      int i;
      FILE * fd;
      fd = fopen (argv [1], "r");
      irakur_osoa (fd, lib); /* memorian edukitzeko */
      sortu (& lis);       /* lista hutsa */
for (i = Ø; i < MAX; i++) /* listan txertatzeko */
    {
      if (lib [i] . kod! = Ø)
          {
            p = malloc (sizeof (struct elem));
            p -> infor = & lib [i];
            p -> lehentasuna = lib [i]. urte;
            txertatu (& lis, p);
          }
      }
p = lehena (& lis);
while (p! = NIL)    /* ordenean atzitzeko */
    {
      idatz (p -> infor);
      p = lehena (& lis);
    }
}
```

***9. programa. Listen gaineko funtzioen erabilpena liburuen sailkapenerako.***

Programa estekatzean listei dagokien liburutegia aipatu beharko da.