# C PROGRAMAZIO–LENGOAIA (VIII)

# Datu-egiturak (II)

Iñaki Alegria & Montse Maritxalar

## EGITURAK, BILDURAK ETA ENUMERATUAK

Aurreko kapituluan datu egituratuak aztertzen hasi ginen, bertan taulak eta erakusleak azaldu genituelarik. Oraingo honetan hiru datu-mota berri aztertuko ditugu: egiturak edo erregistroak, bildurak eta enumeratuak.

### Egiturak

Taulan datu guztiak mota berekoak diren bitartean, egitura edo erregistroan datu-multzo bat dago, baina ez dute mota berekoak izan behar. Egitura osatzen duten datuak unitate edo elementu bati dagozkio, diskoan gordetzen den datu-mota arruntena izanik. Liburutegiaren fitxategian adibidez, liburu bakoitzeko erregistro (edo egitura) bat dago, bertan kodea, izenburua, idazlea, orri-kopurua, etab. (mota desberdineko datuak) kokatzen direlarik.

Taulekin konparatuz datu bat egituran zehazteko indize bat erabiltzea ezinezkoa denez, datu edo eremu bakoitzari izen bat egokitzen zaio eta horrela datu bat identifikatzeko egitura eta eremuaren izenak zehaztu beharko dira.

C lengoaiaz egiturak maneiatzeko **struct** hitz gakoa eta • eragilea erabiliko dira.

```
struct liburu
  { int kod;
      char izenburua [30], idazlea [30];
      int orri-kop; salneurria;
      char arg-etxea [30];
  }; /* egituraren definizioa*/
/* prezio-tarte batean dauden  liburuen zerrenda lortzen duen funtzioa */

print_lib (min, max)
int min, max; /* prezio-tarterako */
{
struct liburu lib; /*liburu-motako aldagaia*/
int kod-buk;
kod-buk = irakur_lib (& lib);
while (kod-buk > = Ø)
  {
      if ((lib.salneurria <= max) &&
          (lib.salneurria >=min))
          printf ("%d %s %s %d \n", lib.kod, lib.izenburua,
                  lib.idazlea, lib.salneurria);
      kod-buk = irakur_lib (&lib);
  }
```

1. programa.
Egituren definizioa eta erabilpena.

Beste datu-motetan bezala egituretan definizioa eta erabilpena bereiztuko ditugu, baina oraingo honetan definizioa bi urratsetan egingo da, zeren lehenengoan egitura abstraktua dagokien eremuekin definitzen den bitartean (egituraren definizioa), bigarrenean mota honetako aldagaia (k) definituko bait da.

1. programan liburutegiko fitxategiaren erregistroa definitu eta erabiltzen dugu.

Beraz, ikus daitekeenez, egituraren definizioaren eremuen izenak eta motak agertzen dira { eta } artean, **struct** eta izena aurretik eta **;** atzetik azaldu behar delarik. Erabilpenean aldiz, hasieran mota horretako aldagaia definitutakoan eremuak adierazteko aldagaia, • eragilea eta eremuaren izena zehazten dira.

Egitura-mota bereko bi aldagai badaude haien arteko esleipena konpiladore batzuekin asignazio bakarrez posible den bitartean, beste batzuekin eremuz eremu burutu behar da. Dena den, taulekin gertatzen zenaren kontra, egitura motako aldagai baten izenak bere balioa adierazten du eta ez erreferentzia. Beraz, egitura bat parametro gisa zehaztean **&** zehaztuko zaio aurretik eta trukatuko dena bere erakuslea izango da.

## Egiturak eta erakusleak

Egiturak eta erakusleak konbina daitezke. Batetik, egitura baten osagaia erakusle bat izan daiteke, eta bestetik, aurretik esan dugunez, egituran erakusleak erabiltzen dira; funtzio-deietan batez ere.

Adibide gisa pentsa dezagun aurreko egitura idazteko, liburutegiarena hain zuzen, funtzio bat dugula eta parametro gisa **liburu** egituraren erreferentzia. Funtzio hau 2. programan definitzen da, eta • eragilea erabili beharrean → eragilea erabiliko da, zeren **(* erakusle) . eremua** eta **erakusle -> eremua** baliokide bait dira C-z.

Lehenengo programan **printf** aginduaren ordez ondoko deia idatziko litzateke kasu honetan:

**print_lib_bat (&lib)**

## Egitura kabiatuak

Egituren erakusleak erabiliz lor daiteke egitura batean egitura bereko beste aldagai bat erreferentziatzea. Hau oso erabilia da zerrendak, iladak pilak, eta antzeko datu-motak definitu eta erabiltzeko.

3. programan stringen arbola bitarra definitzeko erabiltzen den egitura eta funtzio bat azaltzen dira. Funtzioak arbola ezkerretik eskuinera idazten du algoritmo errekurtsiboa erabiliz.

```
struct nodo {
  char izen [MAX];
  struct nodo * ezker, eskuin;
  };
struct nodo arbola [NMAX]; /* egituren taula*/

void print_arbola (pndo)
struct nodo * pnodo;
{
if (pnodo! = NULL)
  {
  print_arbola (pnodo -> ezker);
  printf ("%s\n", pnodo -> izen);
  print_arbola (pnodo -> eskuin);
  }
}
```

3. programa. Egitura konplexu baten adibidea.

## Bildurak

Bildurak definitzeko eta erabiltzeko egituretan aztertutako eragile berberak (. ->) erabili arren, datu-mota honen helburua zeharo desberdina da, zeren eta bietan deskribatutako eremuak mota desberdinekoak izan badaitezke ere, bilduretan eremu horiek bait dira eremu bakar bati dagokion motaren interpretazio desberdinak, egituretan multzo baten osagaiak diren bitartean. Beraz, bildura motako datu bat irakurtzean bilduran definitutako eremuen artean bat baino ezingo da erabili. Mota hau gutxi erabiltzen bada ere, fitxategi batetik irakurritakoa datu-mota desberdin gisa tratatzeko memoria gutxiago gastatuz da duen erabilpen hedatuena. Adibidez, 4. programan 80 byteko memori zati bat inprimatzen da, baina kode baten arabera 80 karaktere dira edo 20 zenbaki oso luze.

```
void print_lib_bat (plib)
  struct liburu * plib;
  {
  printf ("%d %s %s %d \n", plib -> kod,
              plib -> izenburua, plib -> idazlea,
              plib -> salneurria);
  }
```

2. programa. Liburu egitura inprimatzen duen funtzioa.

```
union kar-zen{
  char kate [80];
  long zenb [20];
  };
print_bildu (p, kod)
union kar-zen *p; /*bilduraren erakuslea*/
int kod;
{
  int i;
  if (kod ==Ø)
       for (i = Ø; i < 80; i++)
            printf ("%c", p → katea [i]);
       else for (i = Ø; i < 20; i++)
            printf ("%d\t", p -> zenb [i]);
}
```

4. programa.
Bildura baten adibidea.

osoez egin daiteke (balio bakoitzari kode bat emanez), baina programa irakurterraza izan dadin, oso datu-mota aproposa gertatzen da. Zerrendaren definizioa **enum** hitz gakoak, motaren izenak eta balio posibleek, komen bidez bereiztuak eta giltzen artean osatzen dute.

5. programan bere aplikazio bat ikus daiteke, asteko egunaren arabera tarifa desberdina aplikatzen duen funtzio batekin:

```
enum aste-eg {astele, astear, asteaz, ostegu, ostira, larunb, igande};
long tarifa (eguna, tar1, tar2)
  enum aste-eg eguna;
  long tar1, tar2;
  {
  switch (eguna)
       {
       case astele:
       case astear:
       case asteaz:
       case ostegu:
       case ostira: return (tar1);
       default: return (tar2);
       }
  }
```

5. programa.
Enumeratuak erabiltzen dituen funtzioa.

Programan ikusten denez, bildurak erakusle eta taulekin konbina daitezke. Horrela bildura baten erakuslea bidali egin da funtziora eta bilduraren eremuak taulak dira. Bildurak egiturekin ere konbina daitezke; posible bait da egitura baten barruan bilduraren bat egotea, eta alderantziz, bilduraren barruan egiturak egotea.

## Enumeratuak

Datu-mota hau, aldagai baten balio posiblea multzo labur baten osagaietako bat denean erabiltzen da. Mota honetan egiten dena, karaktere edo zenbaki

## SINONIMOAK

Aurrekoan esan genuenez, C-k ez du datu-mota berriak definitzeko ahalmenik, baina hala ere antzeko zerbait egin daiteke **typedef** eragilearen bidez sinonimoak definitzeko aukeraz.

Metodo honen bidez programek irakurgarritasuna hobetzen dute, baina ez da benetako datu-mota berria sortzen, ondorio bakarra konpilazio-garaian string-en itzulpen soila bait da (makroen kasuan bezala).

Horrela datu-mota boolearra defini daiteke 6. programan ikus daitekeenez:

```
#define TRUE 1
#define FALSE Ø
typedef short bool; /* bool short-en sinonimoa da */

bool eta (a,b)

  bool a, b;
  {
  return (a&b);
  }
```

6. programa.
typedef erabilpena mota boolearra simulatzeko.