C PROGRAMAZIO–LENGOAIA (X)

Funtzioak II

Iñaki Alegria & Montse Maritxalar

Seigarren kapituluan ikusi genuen C-ren azpiprograma diren funtzioak nola definitu eta erabili. Bertan, funtzioen definizioa, deia, zein erreferentziak nola egin behar diren azaltzen zen, parametro nahiz emaitzen trukearen mekanismoa ere adierazten zelarik. Kapitulu honetan zehar funtzioei buruzko kontzeptu aurreratuagoak aztertuko ditugu: funtzioak eta erakusleak, **main** funtzio nagusia, erazagupen konplexuak eta funtzioen prototipoak.

FUNTZIOAK ETA ERAKUSLEAK

Funtzioak eta erakusleak modu naturalean C erabiliz nahas daitezke. Beraz, posible izango da funtzio baten parametroa erakuslea izatea, funtzioaren emaitza erakuslea izatea eta funtzioak erakusleen bidez erreferentziatzea.

Parametroak erakusleak izango dira erreferentziaren bidezko parametroak erabili behar direnean; adibidez, parametroan emaitza utzi behar denean (ikus 1. programa).

Funtzio baten emaitza, eta ondorioz funtzio-mota, erakuslea izan daiteke, horretarako funtzioaren erazagupenean izenaren ezkerraldean erakusten den datu-mota eta * karakterea zehaztu behar delarik (ikus 2. programa).

```
/* karaktere-katea baten kar karakterearen lehen
   agerpenaren helbidea itzultzen duen funtzioa */
char * bila_kar (str, kar)
char str [ ];
char kar;
{
  int i
  for (i=Ø; str [i]! = ' \Ø'; i++)
    if (str [i] = = kar) return (& str [i]);
  return ((char *) Ø);
  }
```

2. programa. Erakusle-motako funtzio bat.

1. programa. Hiru zenbakiren sailkapena.

```
    /* hiru zenbaki sailkatu */
void sailka (pa, pb, pc)
    int *pa; *pb; *pc;
  {
    extern void swap ( );
    if (*pa < *pb)  swap (pa, pb);
    if (*pa < *pc)  swap (pa, pc);
    if (*pb < *pc)  swap (pb, pc);
  }
  /* bi osoen arteko trukea */
  void swap (px, py)
    int *px, *py;
  {
    int tartean;
    tartean = *px;
    *px = *px;
    *py = tartean;
    }
```

Azkenik, funtzioetarako erakusleak oso interesgarriak gerta daitezke, datu edo kode baten arabera funtzio edo errutina desberdina exekutatzeko aukera ematen bait digute. Horren adibide argia eten-bektorea edo 3. programan azaltzen den errore-maneiurako modulua dugu. Bertan, funtzioen izenek, taulenek bezala, beraien erakuslea adierazten dute, beraz edozein funtzio erakusten duen aldagai bat definitzeko * ikurra erabili behar da.

Ondoko programan **f** funtzioaren helbidea **pf** aldagaian kokatzen dugu asignazio baten bidez:

```
extern int f ( );
int (* pf) ( );
pf = f;
```

ondo idatzitako testua da, eta bertan, **f** funtzioaren helbidea **pf** aldagaian kokatzen da.

```
void (* erroreak [2]) ( );/*   ezer itzultzen ez duten
                               funtzioetarako
                               erakusleez osatutako
                               taula da erroreak */
void erroreØ ( )
  {
     char c;
     printf ("errore arina, jarraitzeko tekla bat
              sakatu \n");
     scanf ("%c", & c);
  }
void errore1 ( )
  {
     prinf ("errore larria, ezin da jarraitu \n");
     exit ( ); /* programa bukaera UNIX-ean */
  }
void hasi-erroreak ( )
  {
     erroreak [0] = erroreØ;
     erroreak [1] = errore1;
  }
void errore-maneiua (kod)
int kod;
  { (* erroreak [kod]) ( );
  }
```

3. programa. Erroreak tratatzeko modulua.

FUNTZIO NAGUSIA

C programek **main** funtzio nagusia izan behar dute; exekutatzen den lehena bait da. Orain arte programa nagusi den **main** funtzio hau parametrorik gabe erabili badugu ere, batzuetan interesgarria gerta daiteke exekuzio-denboran sistema eragilearen exekuzio-komandotik daturen bat jasotzea. Fitxategiren bat maneiatzen duen programa bat egin nahi dugunean adibidez, bi aukera ditugu: fitxategi finko baterako programatu edo edozein fitxategitarako egin, fitxategiaren izena exekuzio-komandoan zehaztuko delakoan.

Bigarren hau askoz ere orokorragoa gertatzen da. Aukera hau erabili nahi dugunean **main**-aren parametro-gisa **argc** eta **argv** hitz-gakoak zehaztu behar dira argumentu gisa. Ondoren **argc** eta **argv** erazagutuko dira, lehena **int** motakoa, eta bigarrenerako **char * argv [ ]** espresioa erabiliko da; karaktere--kateetarako erakusle-taula bait da **argv**. Geroago programan zehar **argc**-ren balioa komandoan zehaztutako argumentu-kopurua gehi bat izango da, argv [1]-z aldiz, lehen argumentua erreferentziatuko da, argv [2]-z bigarrena, etab., argv [0]-z programaren izena lortuko dugularik. 4 programa oihartzun (echo) izeneko programa dugu; exekuzioan komando-lerroan idazten duguna errepikatzen duena (lehen hitza programa honen izena bada, noski).

```
main (argc, argv)
int argc;
char * argv [ ];
{
  int i
  for (i = Ø; i < argc; i++)
     printf ("%s", argv [i]);
  printf ("\n");
}
```

4. programa. Oihartzun-programa.

ERAZAGUPEN KONPLEXUAK

C-z idatzitako prometan batzuetan erazagupen konplexuak agertzen dira, deklaratutako datua zer den jakitea zaila gertatuz. Horregatik, hirugarren programan azaldutako **erroreak** aldagaiaren erazagupena ez da berehala ulertzen.

void (* erroreak [2]) ( )

Zailtasunaren arrazoi nagusia honetan datza: * eragilea aurretik idazten den bitartean **[ ]** eta **( )** atzetik, taula eta funtzioaren eragileek lehentasun handiagoa dute erakuslearenek baino, eta elkarketa ezkerretik eskuinera da **[ ]** eta **( )** eragileen kasuan, baina alderantziz erakusleenean.

Definizioa osatzeko ondoko erregelei jarraitzea da errazena:

– ezkerretatik hasi eta izenean bukatu.
– lehentasun erregelak alderantziz aplikatuz, ondoko itzulpena egin:

( ) ... itzultzen duen funtzio
[ ] ...-z osatutako taula bat
* ...-rako erakusle

Adibidez **char * x [ ]**

karaktere -rako erakusle -z osatutako taula bat da X

eta **struct S (* (* (* x) [ ]) ( )) [ ]**

S egitura -z osatutako taula -rako erakuslea itzultzen duen funtzio -rako erakusle -z osatutako taula bat –erako erakuslea da x

FUNTZIOEN PROTOTIPOAK

Seigarren kapituluan funtzioekin lotutako kontzeptuak aztertu ditugu; definizioa, erazagupena, eta deia hain zuzen. Funtzioen definizioan parametro formalak zehazten diren bitartean, dagozkien motak aipatuz, funtzio-deian parametroen momentuko balioa zehazten da, baina ikusten ez diren bihurketak gertatzen dira, zenbaki oso motzak **int** bihurtzen bait dira eta errealak **double.** Horrek eraginkortasun-galera dakar, eta gainera parametro-trukean ez da egiaztapenik gertatzen. Honen aurrean C konpiladore batzuk (ANSI C araua jarraituz) funtzio-erreferentzian funtzioaren izena eta emaitzaren mota aipatzeaz gain, parametroen datu-motak zehaztu behar dira. Adibidez:

extern void funtz1 (int, float, char *);

funtzio-prototipo bat da, non funtzioak dituen hiru parametroen mota (**int, float eta char ***) zehazten bait da. Parametro-trukean egiaztapenik ez egotea nahi bagenu, orduan **extern void funtz1 ( );** erreferentzia egingo genuke.

5. programan konpilazio-errore baten gertaera ikus daiteke parametroen arteko bateragarritasun eza dela eta.

```
...
{ extern void f (int *)
  ...
  float x;
  ...
  f (x); /*  ERROREA float eta int* ez dira
             bateragarriak */
  ...
```

5. programa. Prototipo baten erabilpena.