C PROGRAMAZIO-LENGOAIA (VI)

Funtzioak eta Makroak

Iñaki Alegria & Montse Maritxalar

FUNTZIOAK

Goi-mailako beste lengoaietan bezala C-k azpiprogramak idazteko aukera ematen digu. Lengoaiaren arabera azpiprogramei izen desberdinak ematen zaizkie, prozedura, funtzioa, errutina, azpirrutina arruntenak direlarik. Izendapen desberdinak garrantzi txikiko ezaugarri diferenteek desberdintzen dituzte. Horregatik funtzioa diogunean, C-ren kasuan adibidez, azpiprogramak inplizituki emaitza bat itzuliko duela suposatzen da. Prozedurak aldiz, emaitza inpliziturik ez du itzultzen.

Dena den, C-ren funtzioen helburua, azpiprograma guztiena bezala, programan diseinua eta idazketa erraztea da; beraien bidez posible bait da problema konplexu bat beste errazagoetan banantzea, bakoitza programa errazagoaz ebatziko delarik. Teknika honi beheranzko programazioa edo programazio modularra deitzen zaio eta gaur egun oso hedapen handia du programatzaileen eraginkortasuna (garapen zein zuzenketaren garaian) hobetzen duelako.

Ikus dezagun adibide bat; faktorialaren kasuarena esaterako. Aurreko bi kapituluetan faktoriala kalkulatzen duten programa desberdinak egin ditugu, baina beti programa bakar batean. Hori ez da aukera bakarra; zeren faktorialaren kalkulua azpiprograma edo funtzio batean programa bait daiteke behin-betirako eta programa nagusi desberdinetatik erabili 1. programan azaltzen den legez.

2. programan funtzio beraren erabilpena ikus daiteke,

```
/* funtzioaren definizioa*/
long faktoriala (n)
int n;
  {  long em = 1;
     int i;
     for (i = 1; i ≤n; i++)
          em = em * i;
     return (em);
  }
```

```
/* faktorialaren 1. erabilpena
main ( )
{
int zenb;
long fakt;
scanf (“%d”, & zenb);
if (zenb < Ø) printf (“errorea\n”);
   else {  fakt = faktoriala (zenb);
           printf (“%d\n”, fakt);
        }
}
```

1. programa. Faktoriala funtzio baten bidez.

$\frac{m!}{(m-n)!\,n!}$ formulaz $\binom{m}{n}$ espresioa kalkulatzeko.

Bigarren programan ez dugu faktorialaren kodea idatzi, eta funtzioa aparte konpilatuko dela adierazi beharko zaio konpiladoreari. Horregatik faktoriala zein motatakoa den adierazi behar da, extern bereizgarria zehaztuz. Eskema honekin, faktoriala funtzioaren definizioa konpilatzean sortzen den objektu-modulua programa nagusiak sortutakoarekin batera estekatuko da programa exekutagarri bakarra lortuz.

```
main ()
{
extern long faktoriala ();
int m, n;
float em;
scanf (“%d %d”, &m, &n;
em = faktoriala (m)/(faktoriala (m–n) * faktoriala (n))
printf (“%d\n”, em);
```

2. programa. Faktorial funtzioaren 2. erabilpena.

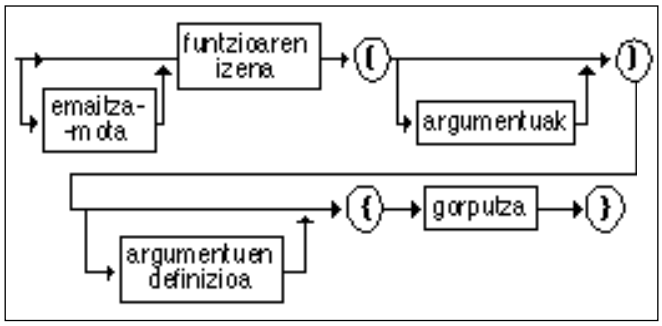
Adibideetan ikusitakoaren arabera, C prometan funtzioak erabiltzen baditugu hiru moduko aipamenak egin daitezke: definizioa, deia eta erazagupena.

Funtzioaren definizioa

Azpiprogramak egiten duena adierazteko balio du, bertan funtzioaren emaitza-mota, izena, argumentuak, argumentuen definizioa, eta gorputza zehazten direlarik.

Argumentu bat baino gehiago badago **,** karaktereaz bereizten dira. 1. adibidean funtzioaren izena **faktoriala** da, emaitza-mota **long**, argumentua **n**, argumentuaren definizioa **int n**; eta gorputza ondoan dagoen guztia programa nagusiraino. Kontutan hartu beharrekoa da programa nagusia argumenturik gabeko main izeneko funtzioa dela.

Definizioari dagokion formatua ondokoa da:



Funtzioak emaitzarik itzultzen badu, return sententziaren bidez egingo da, gehienetan sententzia hau gorputzaren azkena delarik. Funtzioak emaitzarik itzultzen ez badu (beste lengoaietan honi prozedura deitzen zaio), emaitza-mota gisa **void** idatziko da, zeren emaitza-motarik aipatzen ez bada zenbaki osoa itzuliko dela suposatuko bait da.

Argumentuen definizioa datuen definizioa bezala egiten da. Diferentzia bakarra datu hauek beste funtzioetatik (edo programa nagusitik) bidaltzea da. Horrexegatik argumentu hauei *formalak* deitzen zaie eta funtzioari deitzean datu zehatzekin beteko dira.

Funtzioaren gorputzean azpiprogramaren kodea doa, bertako datuen definizioa zein aginduak bertan adierazten direlarik.

Goazen bi zenbaki osoetan handiena itzultzen duen definizioa idaztera: izena asmatuko dugu, **handien** adibidez, parametroak bi dira eta gainera biak zenbaki osoak (**a** eta **b** izenekoak hain zuzen) eta emaitza-mota bi zenbakietako bat denez osoa izango da gainera. Beraz, funtzioaren burua honako hau izango da:

```
int handien (a, b)
int a, b;          /*funtzioaren parametroak */
```

Gorputza oso sinplea da eta hemen egindakoan **emaitza** izeneko bertako aldagaia erabiliko da.

```
{
int emaitza        /*bertako aldagaia*/
if (a > b) emaitza = a;
     else emaitza = b;
return (emaitza)
}
```

Funtzioaren deia

Funtzio bat exekutatu nahi denean dei bat egiten zaio, horrela deian aipatutako datuekin (parametroekin) funtzioaren definizioari dagokion kodea exekutatu eta emaitza itzuliko delarik.

Funtzioak emaitzarik itzultzen badu, deia normalean asignazio baten eskuin aldean agertuko da. Hala ere, espresio baten erdian edo beste funtzio baten parametro gisa ere ager daiteke.

Pentsa dezagun bi zenbaki irakurri ditugula z1 eta z2 aldagaietan eta handiena idatzi nahi dugula. Ondoko bi aukera hauek dauzkagu:

a)
```
int em;
...
em = handien (z1, z2)
printf ("%d", em);
...
```

b)
```
printf ("%d", handien (z1, z2);
```

Hasieran azaldutako 2. programan, faktoriala funtzioari programa berean hiru aldiz deitzen zaiola ikus daiteke, baina parametro desberdinak erabiliz.

Azpiprograma batean emaitza bat baino gehiago lortu behar denean, ezin da return sententziaz egin (honen bidez emaitza bakarra itzul bait daiteke). Hau erreferentziaz definitzen diren parametroen bidez egingo da (ikus 10. kapitulua).

Funtzioaren erazagupena

Programa batean beste modulu batean definituta dagoen funtzio bati deitzen diogunean funtzioa erazagutu egin behar da, definizioaren faltaz konpiladoreak akatsik eman ez dezan. Hori 2. adibidean egin dugu eta erazagupenean zehaztu behar dena ondokoa da: **extern** hitz gakoa kanpoan definitzen dela adierazteko, emaitza-mota, funtzioaren izena eta **( )** funtzioa dela bereizteko. Ari garen adibidean erazagupena honako hau litzateke.

```
extern int handien ( );
```

Funtzioa eta deia modulu berean daudenean ez da funtzioaren erazagupenik behar, funtzioaren definizioa deia baino lehenago idazten bada, bestela erazagupena beharrezkoa da baina **extern** hitz-gakoa jarri gabe.

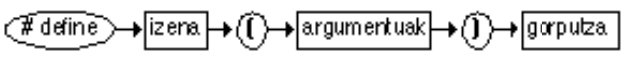
MAKROAK

Mihiztadura-lengoaietan hain arrunt den programazio-tresna hau goi-mailako lengoaia gutxi batzuetan agertzen da. C-n adibidez.

Makroak azpiprograma sinpleak definitzeko eta erreferentziatzeko erabiltzen dira, baina funtzioekiko ondoko desberdintasunak aurkezten dituzte:

- deiek ez dute azpiprogramaren exekuzioa eragiten. Aldiz konpiladoreak dei bat ikustean definizioaren gorputzaz ordezkatzen du. Beraz, konpilazio-garaian deiak desagertu eta kode berri bat konpilatu egiten da.
- programa exekutagarriak luzeagoak, baina azkarragoak, gertatzen dira.
- parametroekin parentesiak erabiltzen ez badira, arazoak egon daitezke.
- azpiprograma konplexuetarako ez dira egokiak.

Makroak definitu eta erreferentziatu (deitu) egiten dira. Definizioaren formatua honako hau da:



eta handienaren kasua honelaxe egingo genuke:

#define handien (x, y) ((x) > (y)? (x) : (y))

if egitura erabili beharrean, **?:** espresioa erabili izan da; bestela emaitza itzultzerik ez bait dago. Erreferentziak funtzioaren deiak bezala egiten dira, eragina lortzeko bidea oso diferentea izan arren.

Horrelako erreferentzia jarrita,

em = handien (zenb1, zenb2);

konpiladoreak makina-lengoaian bihurtzean egiten duen ordezkapenak ondoko emaitza lortuko duelarik:

em = ((zenb1) > (zenb2))? (zenb1) : (zenb2))

Horrela da, zeren eta konpiladoreak makro izen bat ikustean makroaren gorputzaz ordezkatzen bait du, argumentu formalen ordez erreferentzi parametroak idazten dituelarik.

Ikusten denez mekanismoa funtzioena baino murritzagoa da, eta horregatik gutxitan erabiltzen da.

Bukatzeko, zirkuluaren azalera kalkulatzeko makroa ipiniko dugu:

#define PI 3.1416

 # define azalera (r) PI ∗(r)∗(r)

ELHUYAR

KULTUR ELKARTEAREN

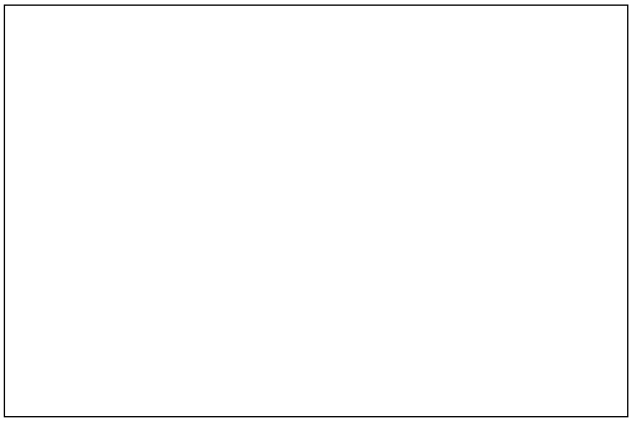
ORDENADORE-SOFTWAREAK

I.X.I.

Zubieta

* Aditzarekin jolastuz
* Euskal aditz laguntzailea
* Euskal deklinabidea
* Euskal Herria ezagutuz: Gipuzkoa
* Euskal Herria ezagutuz: Bizkaia
* Geografia fisikoa (ibaiak)
* Geografia politikoa (herriak)

E.A.E.



Argiñetako nekropolia (Elorrio)

* Elementu kimikoen taula periodikoa
* Fisika begibistan
* Fisika BBB 2 (ariketak)
* Fisika. Oinarrizko kontzeptuak
* Matematika BBB 1 (ariketak)
* Matematika BBB 2 (ariketak)
* Flip/Flop asinkronoak eta sinkronoak