C PROGRAMAZIO-LENGOAIA (IV)

Kontrol-egiturak I

Iñaki Alegria & Montse Maritxalar

KONTROL-EGITUREN bidez espresioak aukeratu edo errepikatzeko aukera dago. Aurreko kapituluan azterturiko espresioak erabiliz programa guztiz linealak eta sekuentzialak idatz daitezke, baina programa gehienetan datuen arabera zenbait agindu exekutatu ala ez erabaki behar da, baldintzazko egituren bidez, eta agindu-multzoen egikaritzapena errepikatu egitura errepikakorrak erabiliz. Egitura hauek C lengoaiaz nola idatzi eta erabili da kapitulu honen helburua.

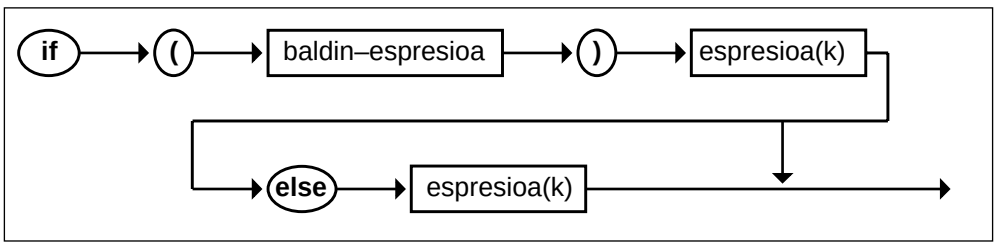
Baldintzazko egituren artean bi dira erabilienak eta C lengoaiak bi hauek dauzka: *baldin* egitura (if) eta *aukera* (switch). Lehenengoan espresio logiko baten balioaren arabera bi adarren artean aukeratzen da. Bigarrenean, aldiz, aukeran nahi adina adar jar daitezke

Egitura errepikakorretan C oso aberatsa da; hiru motakoak bait dauzka: *bitartean* (while), *errepika* (do/while) eta *aldatuz* (for). Beren ezaugarriak aurrerago aztertuko ditugu.

IF egitura

Egitura edo sententzia hau edozein lengoaiatan dago eta C-n ezaugarri orokorrak matendu egiten ditu, hau da, espresio logiko bat ebaluatzen da eta emaitzaren arabera ondokoari ala *bestela*ri (else) dagokion adarra egikarituko da.

Egiturari dagokion sintaxia honako hau da:



Ikus daitekeenez *bestela* adarra aukerakoa da, agerpenaren arabera egitura osoa edo sinplea bereiziz.

Adibidez, ondoko programa-zatian agertzen den **if** egitura sinplea da.

```
int n;
...
if (n < 0) n = –n; /* balio absolutua*)
...
```

Aldiz, honako zati honetan egitura osoa dugu:

```
int n, resul;
...
if (n < 0) printf ("errorea: negatiboa");
        else {  resul = n % 5;
                  printf ("%d modulu 5 = %d", n, resul);
              }
...
```

Ondoko ñabardura hauek hartu behar dira kontutan:

- Adarretako espresioak **;** karaktereaz bukatu behar dira.
- Agindu bat baino gehiago duten adarretan **{** hasieran eta **}** bukaeran zehaztu behar da.

- Baldintzazko espresioan berdintasuna adierazteko = = eragilea erabili eta ez =, zeren kasu honetan ez bait da errorerik sortzen, baina esleipen bat gertatzen da eta esleitzen den balioa Ø bada, emaitza faltsutzat hartuko da eta gainerakoetan egiazkotzat.

- If baten barruan beste if bat dagoenean (if kabiatua) arazoa dago *bestela* adarrekin, zeren **else** bat aurkitutakoan lehenengo edo bigarrenari ote dagokion zalantza egon bait daiteke. Honelakoetan dagoen araua honakoa da: **else** bat idatzitako azken **if**-ari dagokio beti, marjinak kontutan hartu gabe.

- If egitura **baldintza? espresio1: espresio2;** espresioaren bidez ordezka daiteke 1. programan ikus daitekeenez.

```
main ( ) /* bi zenbakien arteko handiena */
{ int a, b, hand;
printf (“sakatu bi zenbaki \n”);
scanf (“% d”. &a); scanf (“% d”, &b);

if (a > b) hand = a;
      else hand = b;
```

→

```
hand = ((a > b)? a:b);
```

```
print f (“% d eta %d artean handiena; %d”, a, b, hand);
```

1. programa. Baldintzazko egituraren adibideak.

SWITCH egitura

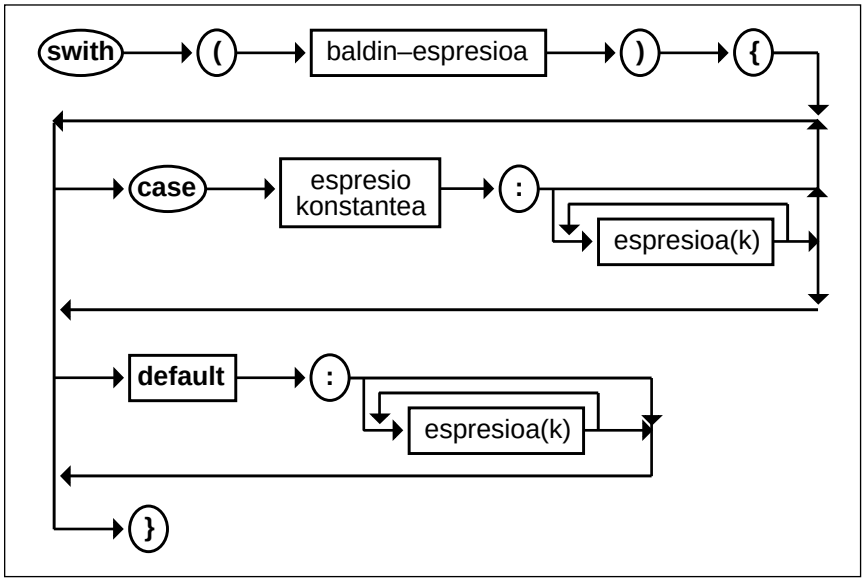
Egitura honen bidez espresio baten balioaren arabera nahi dugun adinako adar-kopurua sor daiteke, adar bakoitza **case** hitz gako batez hasten delarik (azkenekoa izan ezik; **default** hitz gakoaz has bait daiteke). Dagokion sintaxia honako hau da:

Ikus daitekeenez, adarretan espresioak aukerakoak dira **default** adarra bezala. Adar batean espresio bat baino gehiago jartzen bada, ez da **{** eta **}** jarri behar, baina **;** karaktereaz bukatu behar du bakoitzak.

Exekuzioan **swicth** bat aurkitzen denean, baldin--espresioa ebaluatzen da eta balioaren arabera bilatzen da zein adarretan espresio kontanteak balio berbera duen. Horrelako adarrik aurkitzen bada, adar horri eta gainerako hurrengoei dagozkien sententziak burutzen dira. Pascal-eko “case” egituran, adarrari dagozkion espresioak baino ez dira burutzen. Dena den, C-z horrelakorik nahi bada, adarretako azken espresioa **break** izango da.

```
main ( ) /* eragilea sakatuta eragiketa burutu */
{
char c
int a, b;
printf (“sakatu +, –, * edo /”);
scanf (“%c, &c);
a = 10;
b = 5;
switch (c)
    {  case ‘+’: printf (“%d\n”, a+b);
                 break;
       case ‘–’: printf (“%d\n”, a–b);
                 break;
       case ‘*’: printf (“%d\n”, a*b);
                 break;
       case ‘/’: printf (“%d\n”, a/b);
                 break;
       default:  printf (“gaizki sakatu dugu\”);
    }
}
```

2. programa. Switch eta break konbinatuak.



Espresioak duen balioa adarretan aurkitzen ez bada, **default** adarraren espresioak burutuko dira, adar hau idatzi bada noski.

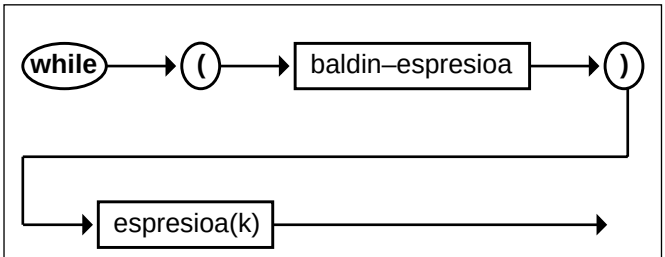
2. programan switch-aren erabilpen konbentzionala (break erabiltzen duena) azaltzen da. 3.ean aldiz, erabilpen bereziagoa.

```
main ( ) /* karakterea bokala den ala ez esan */
{
char c;
scanf (“%c, &c);
switch (c)
    {  case ‘a’:
       case ‘e’:
       case ‘i’:
       case ‘o’:
       case ‘u’:  printf (“bokala da \n”);
                  break;
       default: printf (“ez da bokala \n”);
    }
}
```

3. programa. Switch–aren erabilpen ez–konbentzionala.

WHILE egitura

Egitura errepikakor orokorrena da Pascal-eko “while-do” sententzia bezalakoa. Sintaxiaren aldetik:



Baldin-espresioa ebaluatzen da eta egiazkoa (ez Ø) bada, egiturari dagozkion espresioak burutzen dira eta espresioa ebaluatzera itzultzen da, bigizta osatuz.

Espresioaren ebaluazioa faltsua denean, ondoko sententziara pasatzen da.

Parentesien arteko espresioa da errepikatzeko baldintza eta beste espresioa edo espresio-multzoa errepikatzen den gorputza. Gorputzak sententzia bat baino gehiago badu **{** hasieran eta **}** bukaeran beharrezko dira. 4. programan erabilpen bat ikus daiteke.

DO-WHILE egitura

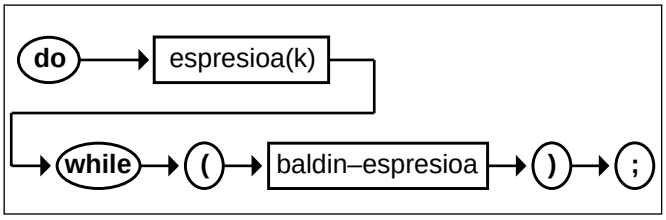
While egituraren aldaketa honetan, baldintzaren ebaluazioa bigiztaren bukaeran gertatzen da. Beraz,

```
main ( ) /* n faktoriala */
{
   int n, i;
   long fakt;
   scanf (“%d”, &n);
      if (n < Ø) printf (“errorea: zenbakia < Ø \n”);
          else {  fakt = 1;
                  i = 2;
                  while (i < = n)
                    {
                       fakt = fakt * i;
                       i ++;
                    }
                  printf (“%d faktoriala = %d \n”, n, fakt);
               }
}
```

4. programa. n faktoriala while–ren bidez.

do-while egitura honetan gorputza gutxienez behin exekutatzen da. Aldiz, **while** egituran behin ere ez burutzea gerta daiteke lehen ebaluazioa faltsua bada. Gainerakoan berdin-berdinak dira.

Egituraren sintaxia honako hau da:



5. programan do-while faktorialaren programan egindako aldaketak ikus daitezke.

if berria erabiltzen da Ø eta 1-en faktorialak ondo buru daitezen.

```
...
    else {
         fakt = 1;
         if (n > 1) { i = 2;
                      do {
                          fakt = fakt * i;
                          i ++;
                          }
                      while (i < = n );
                     }
         printf (“%d faktoriala = %d \n”, n, fakt);
         }
```

5. programa. n faktoriala **do–white**ren bidez.

Hurrengo kapituluan egiturekin segituko dugu eta adibideen laguntzaz kontrol-egituren erabilpena sakonduko dugu.