C PROGRAMAZIO-LENGOAIA (V)

Kontrol-egiturak II

Iñaki Alegria & Montse Maritxalar

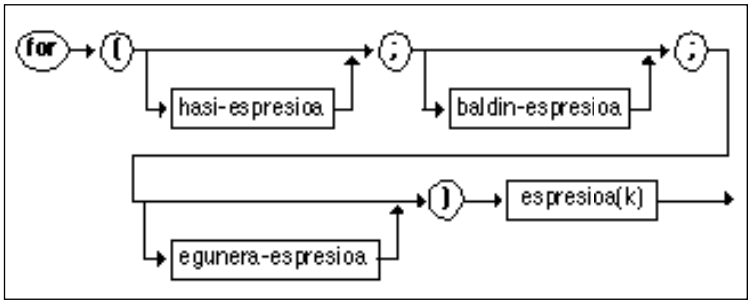
IF eta **switch** baldintzazko egiturak eta **while** eta **do-while** egitura errepikakorrak aurreko kapituluan aztertu eta gero, kapitulu honetan helburua ondoko azalpenetan datza:

- **for** egitura errepikakorra
- **break** eta **continue** sententzien erabilpena egitura errepikakorretan.
- Aztertutako egitura eta espresioen erabilpena zenbait adibide interesgarritan: **if** kabiatuak, bit-maneiua.
- Karaktere-katea eta taulen definizioa eta erabilpena egitura errepikakorretan.

FOR egitura

Egitura errepikakor konplexu honetan, bigiztaren gorputza eta errepikatzeko baldintzaren gain hasieraketarako eta ziklo bakoitzeko eguneratzeko espresioak onartzen dira, kasu batzuetan oso egitura erosoa gertatuz.

Hona hemen sintaxia:



Dagokion funtzionamendua ondokoa da:

1) hasi-espresioa ebaluatu (hasieraketa).
2) baldin-espresioa ebaluatu. Faltsua bada ondoko sententziara pasatu (amaiera).
3) baldin-espresioa egiazkoa bada, gorputzaren espresioa(k) burutzen d(ir)a.
4) egunera-espresioa ebaluatu (eguneratzea) eta 2) puntura.

Beraz, parentesien arteko lehen espresioa (hasieraketa) behin exekutatzen da, bigarrena ziklo bakoitzaren hasieran eta hirugarrena (eguneratzea) ziklo bakoitzaren bukaeran. Dena den **for** egituraz egiten dena **while** eta asignazioaren bidez egin daiteke ondoan ikus daitekeenez.

```
for (e1;e2;e3)
  {
     esp1;
     esp2;
  }
```

≡

```
e1;
while(e2)
  {  esp1;
     esp2;
     e3;
  }
```

1. programan aurreko kapituluan azaldutako faktorialaren kalkulua, **for** egitura erabiliz ikus daiteke.

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

1. programa. Faktoriala **for** erabiliz.

Egituraren espresioak aukerazkoak badira, baldin-espresioak beti agertu behar du; bestela bigizta infinitua gertatuko bait litzateke. Lehen eta hirugarren espresioak agertzen ez direnean **for** eta **while** guztiz baliokideak dira.

Gorputza espresio bakarrak edo **{** eta **}**-en bidez bereiztutako espresio-multzoak osa dezakete, gorputz hutsa ere onartzen delarik (kasu honetan gorputza **;** karaktereak osatzen du).

BREAK eta CONTINUE

Egitura errepikakorretan, batzuetan, bigizta kontrolatzen duen espresioa (baldin-espresioa) bete arren, bigiztatik ateratzea komeni izaten da salbuespenen bat gertatu bada. Horretarako erabiltzen da **break** sententzia egitura errepikakorretan.

2. programan ikus daiteke honen adibide bat. Bertan, 80 karaktere irakurtzen dira eta zenbat maiuskula dagoen kontatzen da. Dena den '.' karakterea irakurtzen bada, ez da karaktere gehiagorik irakurri behar.

```
main () /* maiuskulak kontrolatu */
{
int i, kont = Ø;
char c;
printf ("sakatu 8Ø karaktere \");
for (i = Ø; i < 8Ø; i++)
 {
    scanf ("%c", &c);
    if (c > = 'A' && c <= 'Z') kont ++;
    if (c =='.') break;
 }
}
```

2. programa. Break sententziaren erabilpena.

Beste kasuetan egitura errepikakorraren gorputza konplexua denean, gerta daiteke egoera batean bigiztan jarraitu nahi arren gorputzaren zati bat exekutatu behar ez izatea, horretarako **continue** sententzia oso egokia gertatuz.

3. programan 20 zenbaki irakurtzen dira eta beren arteko biderkadura kalkulatu behar da, baina zenbakia Ø bada ez da kontutan hartzen.

```
main () /* biderkaketa */
{
int i = Ø, zenb;
float em = 1.;
printf ("sakatu  Ø ez diren 2Ø zenbaki");
while (i < 2Ø)
  {
    scanf ("%d", & zenb);
    if (zenb ==Ø) continue;
    em = em * zenb;
    i++;
  }
}
```

3. programa. **Continue** sententziaren erabilpena.

ADIBIDEAK

1. enuntziatua: Irakurri hiru zenbaki oso eta handiena idatzi (4. programa).

```
main () /* 3 zenbakiren artean handiena */
{
int a, b, c;
scanf ("%d %d %d , &a, &b, &c);
if (a > b)
  if (a > c)  printf ("handiena: %d \n", a);
     else     printf ("handiena: %d \n", c);
  else if (b > c)  printf ("handiena: %d \n", b);
     else          printf ("handiena: %d \n", c);
{
```

4. programa. If kabiatuak.

Adibide honetan **if** baten adarretan **if** egiturak agertzen dira (if kabiatuak) eta egitura bakoitza adierazpen sinpletzat jotzen denez ez da giltzarik behar.

2. enuntziatua: Irakurri karaktere bat eta kalkulatu zero egoeran zenbat bit dagoen. (5. programa).

```
main ()
{
char datu, maskara = Ø x  Ø1;
int i, kont = Ø;
scanf ("%c", & datu);
for (i = Ø; i < 8; i++)
    {
      if ((datu & maskara) ==Ø) kont ++;
      maskara = maskara << 1;
    }
printf ("%c karakterean, %d bit Ø egoeran \n "datu, kont);
}
```

5. programa. Bit-maneiuaren adibidea.

Bit-maneiua oso zaila gertatzen da goi-mailako lengoaietan, baina Cz aldiz, izugarri erraza **&** (and), | (or), ^ (xor), << (ezker-desplazamendua) eta >> (eskuin-desplazamendua) eragileei esker. Adibidean, **and** eragiketaren bidez bit bat aztertzen da, zeren **and** eragiketan bit bakar bat 1 egoeran duen maskara batekin datu bat parekatzen dugunean, 1-ri dagokion datuaren bita zeroa bada emaitza zero izango bait da, eta bata bada, emaitza desberdin zero. Eragiketa 8 aldiz errepikatzen den bigizta batean dago (**for** egituraz) baina bit desberdinak aztertzeko, bigiztaren barruan maskarako bitak desplazatu behar dira (programan egin denaren legez), edo datu bera desplazatu.

3. enuntziatua: Irakurri 40 karaktere eta kontatu Ø egoeran zenbat bit duten.

Adibide honen aurrean 2 galdera sortzen dira:

– Karaktereak batera irakur al daitezke ala banan-banan? Posible da denak batera irakurtzea, baina horretarako karaktere-katea datu-mota erabili behar da; orain arte aztertu ez duguna.
– Posible al litzateke hau ebazteko 2. enuntziatuan erabilitako kodeaz baliatzea? Erantzuna baiezkoa da, baina horretarako 5. programa nagusia (main) izan beharrean ekintza ez-primitibo (funtzioa, prozedura edo errutina izenaz ere ezagutzen da) gisa definitu behar da eta hau hurrengo kapituluan aztertuko dugu.

Karaktere-kateak eta taulak

Orain arte ikusitako oinarrizko datuak karaktereak (**char**), zenbaki osoak (**short**, **int**, **long**) eta errealak (float, double) dira. Horien gain, datu-multzoak defini daitezke taulak edo bektoreak osatuz, ondoko adibidean ikus daitekeenez:

char string [8Ø]; /* 8Ø karaktereko katea */
int taula [2Ø], matrizea [M] [N];

Karaktere-katea (string ingelesez) karaktereez osatutako taula da, non azken karakterea '\Ø' konstantea den. Katearen bukaerako karaktere honen bidez katearen amaiera detektatzen da, katea luzera aldakorrekoa denean.

Taulak dimentsio bakarrekoak izan daitezke (**string** edo **taula** adibidean) edo anitzekoak (**matrizea** bikoa da adibidean). Erabiltzean, izena eta indizea (makoen artean) aipatzen dira, indizea Ø eta osagai-kopurua ken bat artean egongo delarik. Definizioan izena eta osagai-kopurua (makoen artean) jarri behar da hasieraketa aukerazkoa izanik. 6. programan adibide bat ikus daiteke.

```
/* lerro bat irakurri eta kontatu zenbat aldiz agertzen den bokal bakoitza */
main ()
{
int kont_bok [5], i;
char lerroa [8Ø], bokalak [6] = "aeiou";
scanf ("%s", lerroa); /*ez da & behar taula denean */
                      /*%s-string formatua */
for (i = Ø; lerroa [i]! = '\Ø'; i++)
   switch (lerroa [i]
   {  case 'a':  kont_bok [Ø] ++;
                 break;
      case 'e':  kont_bok [1] ++;
                 break;
      case 'i':  kont_bok [2] ++;
                 break;
      case 'o':  kont_bok [3] ++;
                 break;
      case 'u':  kont_bok [4] ++;
                 break;
   }
/* karaktereak bukatutakoan idatzi */
for (i = Ø; i < 5; i++)
   printf ("%c-en kopurua: %d\n", bokalak [i], kont_bok [i];
}
```

6. programa. Taulak erabiliz.