# 4. GAIA

# ITERAZIOA

## 1 MOTIBAZIOA

Esan bezala, programatzeak ataza bat ordenagailuaren bitartez ebaztean datza eta ordenagailuak ataza oso konplexuak ebatzi ditzake oinarrizko eragiketa konbinazio desberdinak erabiliz (ordenagailuak oinarrizko eragiketa konkretu batzuk baino ez oinarrizko eragiketa multzo barruan kokatzen dira. Iterazioen bitartez zehaztutako **baldintza bat betetzen den bitartean** ordenagailuak **agindu multzo bera behin eta berriz** egikaritzea lortuko dugu.

### 1.1 Zertarako eta zergatik erabili iterazioa?

**Arrazoi bat** izan **daiteke batzutan agindu sinpleen iterazioa** (errepikapena) **eginez eragiketa konplexuagoak ebatzi daitezkeelako**. Oso ezaguna dugun adibide bat jarrita ikusiko dugu hau. Jakin badakigu kontzeptualki gehiketak zatiketak baino eragiketa sinpleagoak direla (horregatik eskolan lehenago ikasten ditugu); zatiketa bat egiteak gehiketa bat egitea baino abstrakzio maila altuagoa eskatzen duelako. Horrela, gehiketak egiten baino ez dakien ume bati zatiketak egiten irakatsi diezaiokegu, biderketa taulak ikasi gabe, bakarrik gehiketak erabiliz; zatiketa bat azken finean gehiketen iterazio bat (errepikapena) baino ez delako.

### 1.2 Adibideak

#### 1.2.1 1. adibidea

Demagun 347/111 zatiketa osoa egin behar dugula:

0+111=111 (**gehiketa bat** egin dugu) +111=222(**bigarren gehiketa** bat)+111=333(**hirugarren gehiketa**)+111=444 -> azkenengo gehiketa honekin eman diguten zenbakia gainditzen dugu jadanik orduan badakigu jarraitzeak ez daukala zentsurik; orain esan dezakegu **3 gehiketa egin daitezkela, hau da, zatiketaren emaitza 3 dela, edo beste era batera esanda 3 gehiketen iterazioa**. **Gehiketak behin eta berriro eginez** gehiketa horien emaitza **347 baino txikiagoa den bitartean** zatiketaren emaitza kalkulatzen dugu.

**Adibide honen helburua** eragiketa sinple baten errepikapenarekin baldintza bat betetzen ez den bitartean eragiketa konplexuago bat eraiki dezakegula ikustean datza.

#### 1.2.2 2. adibidea

**Iterazioa justifikatzeko beste adibide bat**. Ataza bat planteatuko dugu (kasu honetan bataz besteko bat kalkulatzea izango da) eta lehenengoz iteraziorik erabili gabe ebazten saiatuko gara, aztertuko ditugu sortuko zaizkigun arazoak, eta 4.2 puntuan problema berdina planteatuko dugu iterazioa erabiliz soluzioa askoz sinpleagoa dela ikusiz.

Demagun esaten digutela programatxo bat egin behar dugula eskola batentzat, edozein gelan dauden ikasleen bataz besteko nota kalkulatu ahal izateko. **Ez**

**dakigu zenbat ikasle egongo diren**, gure programa erabiliko duen irakasleak noten zerrenda bat izango du. Badakigu bataz bestekoa kalkulatzeko nota horiek gehitu behar ditugula akumulatuz eta zatituz ikasle kopuruagatik. Orduan *noten akumulazioa* eta *ikasleen kopurua* kalkulatu beharko dugu (hauek izango dira erabili beharko ditugun aldagai batzuk) bukaeran zatiketa egin ahal izateko. Notak banan banan eskatuko ditugu, nota bakoitzeko aurreko noten akumulazioarekin gehitzen joateko, (hau da akumulatzen) eta aldi berean ikasleen kopurua kontatzen joateko. Horrela behar ditugun balioak kalkulatuko ditugu.

**1. bertsioa bataz bestekoa kalkulatu ITERAZIORIK erabili gabe**

**Arazoa:** Sortzen zaigun lehenengo arazoa aldagaien deklarazioa da.

Zenbat aldagai beharko ditugu? Ikasle kopurua guretzat ezezaguna izango da. Norbaitek pentsatu dezake, beno ba estimazio bat egingo dugu, 100 daudela suposatuko dugu. Orduan ikasle bakoitzeko nota gorde ahal izateko aldagai bat beharko genuke.

```
float nota_1, nota_2, nota_3, nota_4, nota_5, nota_6,    eta horrela 100 arte, nota_100.
/*Ordenagailuan DENAK idatzi beharko genituzke*/
```

Orain falta zaizkigu 3 aldagai, bat noten akumulazioa gordetzeko, bigarrena benetako ikasle kopurua gordetzeko (eta ez estimazio bat) eta hirugarren bat bukaeran behin bataz bestekoa kalkulatu eta gero, bere balioa gorde ahal izateko.
float noten_akumulazioa, benetako_ikasle_kopuru, bataz_bestekoa;

Ematen du lehenengo arazoa gainditu dugula, baina proposatu dugun irtenbidea oso astuna eta neketsua da 103 aldagai sortu behar izan ditugulako. (Hau iterazioa erabiliz guztiz sinplifikatuko dugu. begiratu gero 4.2.1.3).

Orain gure programa hasi zitekeen aldagaiak hasieratuz

```
benetako_ikasle_kopurua = 0; /*hasieran eta daturen bat sartu arte ikasle kopurua 0 da*/
noten_akumulazioa = 0; /*Hau ere hasieran eta daturen bat sartu arte 0 izango da*/

printf ("Sartu nota bat mesedez: ");
scanf("%f",&nota_1);
benetako_ikasle_kopurua = benetako_ikasle_kopurua +1; /*nota bat sartu dugunez ikasle kopurua
                                                        eguneratu dugu, ez da 0 izango baizik eta 1*/
noten_akumulazioa = noten_akumulazioa+ nota_1; /*noten_akumulazioa ere eguneratu behar dugu ez da
                                                 0 izan behar*/

printf ("Sartu nota bat mesedez: ");
scanf("%f",&nota_2);
benetako_ikasle_kopurua = benetako_ikasle_kopurua +1; /*nota bat sartu dugunez ikasle kopurua
                                                        eguneratu dugu, lehenago zeukan balioari
                                                            1gehituz*/
noten_akumulazioa = noten_akumulazioa+ nota_2; /*noten_akumulazioa ere eguneratu behar dugu eta
                                                 bere balio berria izango da lehenago zeukan balioa
                                                 gehi nota_2 aldagaiak duen balioa*/

printf ("Sartu nota bat mesedez: ");
scanf("%f",&nota_3);
benetako_ikasle_kopurua = benetako_ikasle_kopurua +1; /*nota bat sartu dugunez ikasle kopurua
                                                        eguneratu dugu, lehenago zeukan balioari
                                                            1gehituz*/
noten_akumulazioa = noten_akumulazioa+ nota_3; /*noten_akumulazioa ere eguneratu behar dugu eta
                                                 bere balio berria izango da lehenago zeukan balioa
                                                 gehi nota_3 aldagaiak duen balioa*/
```

```
printf (“Sartu nota bat mesedez: ”);
scanf(“%f”,&nota_4);
benetako_ikasle_kopurua = benetako_ikasle_kopurua +1; /*nota bat sartu dugunez ikasle kopurua
                                                        eguneratu dugu, lehenago zeukan balioari
                                                        1gehituz*/
noten_akumulazioa = noten_akumulazioa+ nota_4; /*noten_akumulazioa ere eguneratu behar dugu eta
                                                 bere balio berria izango da lehenago zeukan balioa
                                                 gehi nota_4 aldagaiak duen balioa*/
```

1. Arazoa

Zenbat nota eskatu behar ditugu? Ez dakigu benetan zenbat ikasle dauden….. Ez dugu 100 sartu nahi, ziur aski ez dira hainbeste izango eta lehenago moztu nahiko dugu. Nola egin dezakegu? Beno beste irtenbide bat topatu beharko dugu, adibidez hasieran irakasleari mezu bat aurkeztea esanez, "Notak bukatzen zaizkizunean sartu nota bezala –1 bat ", hau da, *kode bat ezartzen dugu* irakaslearekin (erabiltzailearekin) gure programak bere notak noiz bukatzen diren jakin dezan, orduan –1 bat irakurtzen dugunean badakigu programak ez duela notak eskatzen jarraitu behar….. Irtenbide hau inplementatzeko idatzi dugun kodea aldatu beharko genuke eta hasi hasieratik eman aukera programa bukatzeko, hau da, nota berri bat irakurtzen dugun bakoitzean galdetu beharko genuke ea –1 den.

```
printf (“Sartu nota bat mesedez: ”);
scanf(“%f”,&nota_1);
if (nota_1 != -1){
        benetako_ikasle_kopurua = benetako_ikasle_kopurua +1;
        noten_akumulazioa = noten_akumulazioa+ nota_1;

        printf (“Sartu nota bat mesedez: ”);
        scanf(“%f”,&nota_2);
        if (nota_2 != -1){
                benetako_ikasle_kopurua = benetako_ikasle_kopurua +1;
                noten_akumulazioa = noten_akumulazioa+ nota_2;
                printf (“Sartu nota bat mesedez: ”);
                scanf(“%f”,&nota_3);
                if (nota_3 != -1){
                        benetako_ikasle_kopurua = benetako_ikasle_kopurua +1;
                        noten_akumulazioa = noten_akumulazioa+ nota_3;
                        printf (“Sartu nota bat mesedez: ”);
                        scanf(“%f”,&nota_4);
                        if (nota_4 != -1){
                                benetako_ikasle_kopurua = benetako_ikasle_kopurua +1;
                                noten_akumulazioa = noten_akumulazioa+ nota_4;
                                printf (“Sartu nota bat mesedez: ”);
                                scanf(“%f”,&nota_5);

                        /*…… eta horrela printzipioz 95 bider gehiago.
                        Ordenagailuan ezin izango genuke horrela utzi,
                         falta zaizkigun 95 eragiketa multzoak idatzi beharko genituzke*/
                        }
                }
        }
}

if (benetako_ikasle_kopurua !=0){
        bataz_bestekoa= noten_akumulazioa/benetako_ikasle_kopurua;
  }
```

```
else{
    bataz_bestekoa= 0;
    }
```

Berriro, ematen du planteatu zaigun arazoa gainditu dugula, baina berriro proposatu dugun irtenbidea oso astuna eta neketsua da, kode gehiegi agertzen delako (berriro, hau iterazioa erabiliz asko sinplifikatu daiteke kodea murriztuz, begiratu gero 4.2.1.3). Eta are gehiago, beti gaude eragiketa berdinak errepikatzen.

```
benetako_ikasle_kopurua = benetako_ikasle_kopurua +1;
noten_akumulazioa = noten_akumulazioa+ nota_?????;
printf ("Sartu nota bat mesedez: ");
scanf("%f",&nota_????);
```

Iteraziozko aginduen sintaxia eta esanahia ikusi eta gero, ariketa hau agindu iteratiboak erabiliz birplanteatuko dugu, gure programa nola sinplifikatzen eta nola murrizten den ikusiz.

Orain arte ikasi ditugun oinarrizko eragiketekin (2. gaia: +,-,esleipena e.a.), baldintza eragiketekin (3. gaia) eta orain ikusiko ditugun iteraziozko eragiketekin batera planteatzen diguten edozein ataza ebazteko nahiko tresna izango ditugu. Nahiz eta ataza oso konplexua izan eragiketa sinple hauen konbinazioren batek ebatzi dezake. Programatzea izango da konbinazio egoki bat topatzea.

## 2 ITERAZIOZKO AGINDU DESBERDINAK C-N

Iteraziozko aginduetako baldintzak baldintzazko aginduetarako (if, if-else, ...) azaldutako baldintzen bezalakoak izango dira. Beraz eragile erlazionalak (==, <, >, <=, >=) eta eragile logikoak (!, &&, ||) erabiliz eratutako espresio logikoak izango dira. C lengoaian hiru agindu errepikakor daude:

1. **While** (baldintza){
   Errepikatu behar den eragiketa multzoa
   }.

2. **Do**{
   Errepikatu behar den eragiketa multzoa
   }**while** (baldintza).

3. **For** (hasieraketa;baldintza;eguneraketa){
   Errepikatu behar den eragiketa multzoa
   }

Hirurak oso antzekoak dira horregatik, nahiz eta hirurak azaldu, batez ere while agindua landuko dugu.

## 3 WHILE AGINDUA

- while aginduaren itxura:

```
while (baldintza)
 {
  Errepikatu behar diren
aginduak
 }
```

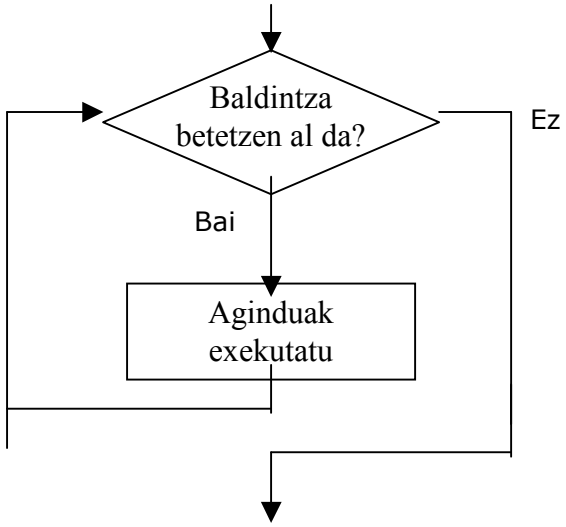
- while aginduaren esanahia:

  Baldintza betetzen den **bitartean**, aginduak behin eta berriz burutu. While agindua baldintza faltsua izatera pasatzen denean bukatuko da.

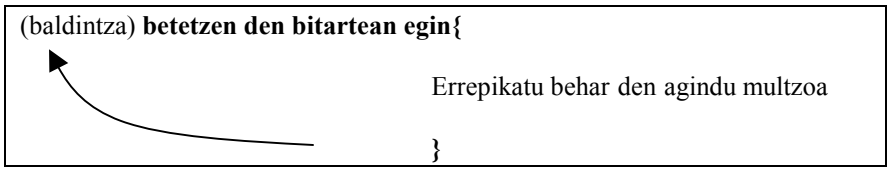
  While agindura iristen garenean, hasieratik baldintza faltsua bada, aginduak behin ere exekutatu gabe aurrera joko da.

  While agindua ulertzeko beste era bat honako hau da: **baldintza faltsua izan arte** agindu-multzoa exekutatu behar da.

- while aginduaren esanahia grafikoki:




- while agindua euskeraz:




Beharrezkoa da buelta batetik bestera aldagairen baten balio aldatzea, hau egoera aldatzea zerbait, bestela ITERAZIO BUKAEZIN baten barruan egongo gara. Hau da, inoiz bukatzen ez den iterazio bat.

Adibidez,

```
a=0;
while(a<10){
        printf (“%d”,a);
}
```

Hau iterazio bukaezin bat izango litzateke.
Programak a aldagaiari 0 balioa esleitzen dio. Gero whilearen baldintza konprobatzera doa eta nola a-ren balioa < 10 den giltzen artean dagoena egikaritzen du, a-ren balioa pantailan inprimatuz. Behin bukatu duela,badoa berriro whilearen baldintza konprobatzera, oraindik egia denez, sartzen da berriro giltzen artean dagoena egikaritzera. Eta berriro inprimatzen du a-ren balioa. Eta horrela behin eta beriro behin eta berriro bukaerarik gabe, a-ren balioa ez delako inoiz aldatzen eta beraz baldintza ez da inoiz gezurra izango..

Aldiz konparatu kode honekin

```
a=0;
while(a<10){
        printf (“%d”,a);
        a= a+1;
}
```

hasieran a-ren balioa 0 da eta baldintza betetzen denez giltza artean dagoena egikaritzen da, hau da a-ren balioa inprimatzen da eta gero a-ren balioa aldatzen da a-ri bat gehituz (A_REN BALIOA ALDATZEN DA). Berriro konprobatzen da baldintza eta a-ren balio 1 izanik, oraindik < 10 denez, berriro egikarituko da giltzen artean dagoena, berriro inprimatuz a-ren balio eta berriro gehituz bat (A_REN BALIOA ALDATUKO DA!!!).Orain ikusten duzuenez a-ren balioa goraka doa eta orain bai baldintza 9 buelta eman ondoren ez da berriro egia izango (a-ren balioa 10 izango bait delako). Iterazio honek badauka BUKAERA.

## 3.1 Adibide sinple bat: Mezu bat bost aldiz aurkezten duen programa

Ariketa honetan idatzi beharreko programak "Kaixo! Egunon!" mezua 5 aldiz aurkeztu behar du pantailan.

Aukera bat bost printf ipintzea izango litzateke, baina mezua 5 aldiz aurkezteko eskatu beharrean 1000 aldiz aurkezteko eskatuko baligute, 1000 printf ipintzeak ez luke zentzurik izango. Era honetako gauzak egiteko while agindua erabiltzea askoz hobea da. While aginduarekin nahikoa da mezua 5 aldiz aurkeztea nahi dugula esatearekin. Guk printf bakar bat ipiniko dugu, eta printf hori 5 aldiz errepikatzeko esango diogu ordenagailuari while-aren bidez.

```
#include <stdio.h>

/* Datuak: Programa honek ez dio daturik eskatuko erabiltzaileari. */

/* Emaitzak: Programak "Kaixo! Egunon!" mezua 5 aldiz aurkeztuko du
             pantailan. */

/* Aldagaiak:
          - zenbat: une bakoitzean une horretara arte mezua zenbat aldiz
                    aurkeztu den jakiteko. */

. */

void main()
 {   int zenbat;


   zenbat = 0; /* zenbat aldagaia 0 balioarekin hasieratu behar da, orain arte
                  mezua 0 aldiz aurkeztu baita. */

   while (zenbat < 5){
```

```
      printf("\nKaixo! Egunon!");
      zenbat = zenbat + 1;
    }

}
```

Ariketa honetako while-a noiz bukatuko da?

(zenbat < 5) betetzen ez denean.

While-eko buelta bakoitzean mezua behin aurkeztuko da. Beraz mezua 5 aldiz aurkezteko 5 buelta eman beharko dira.

Jarraian aurreko ariketarako idatzi dugun programa **nola exekutatuko litzatekeen** azalduko da urratsez urrats, *zenbat* aldagaiaren balioa eta pantaila denboran zehar nola aldatzen diren erakutsiz:

- int zenbat;

  zenbat | ? |

  Pantaila

- zenbat = 0;

  zenbat | 0 |

  Pantaila

- zenbat < 5 betetzen al da? BAI

  zenbat | 0 |

  Pantaila

- printf("Kaixo! Egunon!");

  zenbat | 0 |

  Pantaila | Kaixo! Egunon! |

- zenbat = zenbat + 1;

zenbat 1

Pantaila
```
Kaixo! Egunon!
```

- zenbat < 5 betetzen al da? BAI

zenbat 1

Pantaila
```
Kaixo! Egunon!
```

- printf("Kaixo! Egunon!");

zenbat 1

Pantaila
```
Kaixo! Egunon!
Kaixo! Egunon!
```

- zenbat = zenbat + 1;

zenbat 2

Pantaila
```
Kaixo! Egunon!
Kaixo! Egunon!
```

- zenbat < 5 betetzen al da? BAI

zenbat 2

Pantaila
```
Kaixo! Egunon!
Kaixo! Egunon!
```

- printf("Kaixo! Egunon!");

zenbat 1

Pantaila
```
Kaixo! Egunon!
Kaixo! Egunon!
Kaixo! Egunon!
```

- zenbat = zenbat + 1;

zenbat | 3 |

Pantaila

```
Kaixo! Egunon!
Kaixo! Egunon!
Kaixo! Egunon!
```

- zenbat < 5 betetzen al da? BAI

zenbat | 3 |

Pantaila

```
Kaixo! Egunon!
Kaixo! Egunon!
Kaixo! Egunon!
```

- printf("Kaixo! Egunon!");

zenbat | 3 |

Pantaila

```
Kaixo! Egunon!
Kaixo! Egunon!
Kaixo! Egunon!
Kaixo! Egunon!
```

- zenbat = zenbat + 1;

zenbat | 4 |

Pantaila

```
Kaixo! Egunon!
Kaixo! Egunon!
Kaixo! Egunon!
Kaixo! Egunon!
```

- zenbat < 5 betetzen al da? BAI

zenbat | 4 |

Pantaila

```
Kaixo! Egunon!
Kaixo! Egunon!
Kaixo! Egunon!
Kaixo! Egunon!
```

- printf("Kaixo! Egunon!");

zenbat | 4 |

Pantaila

```
Kaixo! Egunon!
Kaixo! Egunon!
Kaixo! Egunon!
Kaixo! Egunon!
```

| |
|---|
| Kaixo! Egunon! |

- zenbat = zenbat + 1;

zenbat | 5 |

Pantaila

| |
|---|
| Kaixo! Egunon!<br>Kaixo! Egunon!<br>Kaixo! Egunon!<br>Kaixo! Egunon!<br>Kaixo! Egunon! |

→

- zenbat < 5 betetzen al da? EZ

zenbat | 5 |

Pantaila

| |
|---|
| Kaixo! Egunon!<br>Kaixo! Egunon!<br>Kaixo! Egunon!<br>Kaixo! Egunon!<br>Kaixo! Egunon! |

zenbat | 5 |

Pantaila

| |
|---|
| Kaixo! Egunon!<br>Kaixo! Egunon!<br>Kaixo! Egunon!<br>Kaixo! Egunon!<br>Kaixo! Egunon!<br>Sakatu tekla bat amaitzeko. |

### 3.1.1 Lehenengo n zenbaki naturalak pantailan aurkeztuko dituen programa

Ariketa honetan idatzi beharreko programak erabiltzaileari osoa eta positiboa ($\geq 1$) den n zenbaki bat eskatu (konprobatu n <=10 izatea) eta lehenengo n zenbaki natural bikoitiak pantailan aurkeztu behar .

```
#include <stdio.h>


/* Datuak: Programa honek erabiltzaileari [1..10] tarte barruko
           zenbaki bat eskatuko dio. */

/* Emaitzak: Erabiltzaileak [1..10] kanpoko zenbaki bat tekleatzen badu,
.............zenbakia egokia ez dela esanez
              mezu bat aurkeztuko du, beste zenbaki bat tekleatzeko aukerarik
.............eman  gabe.  Aldiz  erabiltzaileak  tekleatutako  zenbakia  egokia
.............balitz 1etik hasita eta erabiltzaileak
              tekleatutako zenbaki arte dauden zenbaki guztiak
              pantailaratu behar dira . */

/* Aldagaiak:
```

```
          - n: erabiltzaileak tekleatuko duen zenbakia jasotzeko erabiliko da.

          - zenb: zenbakiak 1etik hasi eta banan-banan pasatzeko erabiliko da.
                  Une bakoitzean while-eko hurrengo bueltan aztertuko den
                  zenbakia edukiko dugu aldagai honetan.
*/

void main()
 {
   int n, zenb;


 printf("\nOsoa eta positiboa (1<= zenbakia <=10) den zenbaki bat tekleatu: ");
 scanf("%d", &n);

   if ((n < 1)||(n > 10))
    {
      printf("\nTekleatutako zenbakia ez da egokia.");
    }
   else
    {

      zenb = 1; /* while-eko lehenengo bueltan 1 zenbakia aztertu nahi delako.*/
      while (zenb < n)
       {
         printf("\n%d", zenb);
         zenb = zenb + 1; /* hurrengo bueltan hurrengo zenbakia aztertzeko */
       }/*whilearen bukaera*/
    }/*elsearen bukaera*/

 }
```

### 3.1.2 Lehenengo n zenbaki naturalen batura

Ariketa honetan idatzi beharreko programak erabiltzaileari osoa eta positiboa (≥ 1) den n zenbaki bat eskatu eta lehenengo n zenbaki naturalen batura kalkulatu eta pantailan aurkeztu behar du.

Erabiltzaileak tekleatutako n zenbakia 1 baino txikiagoa baldin bada, mezu bat aurkeztu beharko du zenbakia egokia ez dela esanez. Baina ez du beste zenbakirik eskatu behar:

```
#include <stdio.h>

/* Datuak: Programa honek erabiltzaileari osoa eta positiboa (>= 1) den
           zenbaki bat eskatuko dio. */

/* Emaitzak: Erabiltzaileak tekleatutako zenbakia 1 edo handiagoa baldin bada,
             1etik zenbaki horretarainoko zenbakien arteko batura kalkulatu
             eta aurkeztuko du pantailan. Baina erabiltzaileak 1 baino
             txikiagoa den zenbaki bat tekleatzen badu, zenbakia egokia ez
             dela esanez mezu bat aurkeztuko du. */

/* Aldagaiak:
          - n: erabiltzaileak tekleatuko duen zenbakia jasotzeko erabiliko da.

          - zenb: zenbakiak 1etik n-ra banan-banan pasatzeko erabiliko da. Une
                  bakoitzean zenb aldagaian while-eko hurrengo bueltan batuko
                  den zenbakia edukiko dugu.

          - batura: 1etik n-ra arteko zenbakien batura hemen gordeko da.
                    Zenbakiak banan-banan batuko dira. While-eko buelta
                    bakoitzean ordura arteko baturari hurrengo zenbakia batuko
                    zaio. 1etik n-ra arteko zenbaki denak pasatutakoan aldagai
                    honetan zenbaki horien arteko batura edukiko dugu. */
```

```
void main()
 {
   int n, zenb, batura;

   system ("cls"); /* Pantaila garbitzeko. */

   printf("\nOsoa eta positiboa (>= 1) den zenbaki bat tekleatu: ");
   scanf("%d", &n);

   if (n < 1)
    {
      printf("\nTekleatutako zenbakia ez da egokia.");
    }
   else
    {
      batura = 0; /* while-eko buelta bakoitzean ordura arte kalkulatutako
                     baturari zenbaki berri bat gehituko zaio. Horregatik
                     lehenengo bueltan batura aldagaiaren balioa 0 izatea
                     komeni da. */
      zenb = 1; /* while-eko lehenengo bueltan 1 zenbakia batu nahi delako. */
      while (zenb <= n)
       {
         batura = batura + zenb; /* orain arte batuta geneuzkan zenbakiei
                                    zenbaki berri bat batzeko */
         zenb = zenb + 1; /* hurrengo bueltan hurrengo zenbakia batzeko */
       }
      printf("\nLehenengo %d zenbaki naturalen batura %d da.", n, batura);
    }

   printf("\nSakatu tekla bat amaitzeko.");
   getch(); /* Guk tekla bat sakatu zain egongo da ordenagailua. */
 }
```

### 3.1.3 Gure hasierako adibidea: Bataz bestekoa kalkulatzeko programa

Behin dakigula nola kalkulatu n zenbaki bat arte dauden zenbakien batura kalkulatzen, orain bueltatu gaitezke gure hasierako adibidera. Gogoratu eskatzen zigutela klase bateko ikasleen batazbesteko nota kalkulatzea.

4.1 puntuko 2. adibidean aurkeztu dugun programan ariketa hau ebazten saiatu gara iteraziorik erabili gabe. Ikusi dugun bezala arazo batzuk agertu zaizkigu. Konpontzea lortu dugu, baina kodea oso itsusi, astuna idazteko eta luzea gelditu zaigu.

Orain iterazioak sortzeko while eragiketa ezagutzen dugula ariketa berdina ebazten saiatuko gara.
Hona hemen soluzioa:

```
#include <stdio.h>

/* Datuak: Programa honek notak eskatuko dizkio erabiltzaileari. */

/* Emaitzak: Programak nota jorien bataz bestekoa kalkulatuko du*/

/* Aldagaiak:

    nota -> erabiltzaileak banan banan sartzen doan notak gordetzeko aldagaia,
... noten_akumulazioa -> erabiltzaileak sartzen doan notak akumulatzeko
... benetako_ikasle_kopurua -> ikasle kopuru gordetzen joateko aldagaia
....bataz_bestekoa -> Bukaeran noten bataz bestekoa gordetzeko aldagaia


void main()
```

```
 {
   float nota,noten_akumulazioa, benetako_ikasle_kopurua, bataz_bestekoa;


   noten_akumulazioa = 0; /* noten_akumulazioa 0 balioarekin hasieratu behar
                          da, orain arte ez dugulako notarik akumulatu*/
   benetako_ikasle_kopurua=  0;  /*  benetako_ikasle_kopurua  0  balioarekin
.............................hasieratu behar da, orain arte ez dugulako
                               notarik sartu*/
    printf ("Sartu nota bat mesedez: ");
    scanf("%f",&nota);
     while (nota != -1){
        noten_akumulazioa = noten_akumulazioa + nota;
        benetako_ikasle_kopurua = benetako_ikasle_kopurua +1;
        printf ("Sartu nota bat mesedez: ");
        scanf("%f",&nota);
     }
   if (benetako_ikasle_kopurua !=0){
              bataz_bestekoa= noten_akumulazioa/benetako_ikasle_kopurua;
   }
   else{
        bataz_bestekoa= 0;
        }
 }
```

Ikusten den bezala kodea askoz motzago eta argiago gelditu zaigu. Eta beste abantaila batzuk dauzka 4.1 kodearekin konparatuz. buelta bakoitzean *nota* aldagaia berrerabiltzen dugun ez ditugu hainbeste aldagai behar (4 aldagai baino ez ditu erabiltzen) eta ez dugu zertan estimatu behar zenbat aldagai beharko ditugun, 4 aldagai horiekin edozein ikasle kopuruarentzat kalkulatu dezakegu bataz bestekoa ez gara zertan estimatutako ikasle kopuru batera murriztu behar.

### 3.1.4 Lehenengo n (non n <=10) zenbaki natural bikotiak pantailan aurkeztu

Ariketa honetan idatzi beharreko programak erabiltzaileari osoa eta positiboa ($\geq 1$) den n zenbaki bat eskatu (konprobatu n <=10 izatea) eta lehenengo n zenbaki natural bikoitiak pantailan aurkeztu behar ditu.

Erabiltzaileak tekleatutako n zenbakia 0 baino handiagoa eta 10 baino txikiagoa izan behar da. Ez balitz horrela izango mezu bat aurkeztu beharko genuke zenbakia egokia ez dela esanez. Baina ez du beste zenbakirik eskatu behar.

Lehenengo n zenbaki natural bikoitiak aurkezteko[1], zenbaki naturalen multzoan baldintza jakin bat (bikoitiak izatea) betetzen duten zenbakiak bilatu behar direla kontsideratuko dugu. Horretarako 1etik hasi eta zenbakiak banan-banan pasatuz joango gara eta zenbaki bakoitza bikoitia al den ala ez aztertu, eta bikoitia bada, aurkeztu egingo da.

Guztira n zenbaki bikoiti aurkitu behar direnez, kontagailu bat ere beharko dugu, une bakoitzean une horretara arte zenbat zenbaki bikoiti aurkeztu ditugun jakiteko.

Zenbaki bikoitiak jarraian aurkeztuko dira.

```
#include <stdio.h>
```

[1] Hemen aurkezten dugun soluzioa ez da bakarra, adibidez zenb 2rekin hasieratuz gero eta buelta bakoitzean zenb eguneratzen badugu zenb=zenb+2 eginez, orduan zenbaki bikoitiak zuzenean lortuko genituzke, soluzioa zuzenagoa izango litzateke, baina guk soluzio hau hautatu dugu metodologikoki eta kontadoreak azaltzeko egokiena iruditzen zaigulako.

```
/*Datuak:Programa honek [1..10] tarte barruan dagoen zenbaki bat eskatuko du*/

/* Emaitzak: zenbaki horrek adierazten duen adina zenbaki bikoiti
             aurkeztuko ditu pantailan. Baina erabiltzaileak 1 baino txikiagoa
.............den zenbaki bat tekleatzen badu, zenbakia egokia ez dela esanez
             mezu bat aurkeztuko du, baina ez dio beste zenbaki bat
             tekleatzeko aukerarik emango. */

/* Aldagaiak:
          - n: erabiltzaileak tekleatuko duen zenbakia jasotzeko erabiliko da.

          - zenb: zenbakiak 1etik hasi eta banan-banan pasatzeko erabiliko da.
                  Une bakoitzean while-eko hurrengo bueltan aztertuko den
                  zenbakia edukiko dugu aldagai honetan. Zenbakia aztertzea
                  bikoitia al den ala ez erabakitzea da.

          - kop: aldagai hau kontagailu bezala erabiliko da. Une bakoitzean
                 une horretara arte zenbat zenbaki bikoiti aurkeztu diren
                 adieraziko digu aldagai honek. */
void main()
 {
   int n, zenb, kop;

   printf("\nOsoa eta positiboa (1<= zenbakia <=10) den zenbaki bat tekleatu:
");
   scanf("%d", &n);

   if ((n < 1)||(n > 10))
    {
      printf("\nTekleatutako zenbakia ez da egokia.");
    }
   else
    {
     kop = 0; /* hasieran oraindik ez delako zenbaki bikoitirik aurkeztu. */
     zenb = 1; /*while-eko lehenengo bueltan 1 zenbakia aztertu nahi delako.*/
      while (kop < n)
       {
         if (zenb % 2 == 0)
          {
            printf("\n%d", zenb);
            kop  = kop + 1;/*kop bezalako aldagaiak kontadore deituko ditugu,
.........................bilatzen ari garen egoera zenbat bider eman den
.........................jakiteko kontadore bat erabiltzen da*/
          }
         zenb = zenb + 1; /* hurrengo bueltan hurrengo zenbakia aztertzeko */
       }
    }
 }
```

### 3.1.5 n zenbaki bat 2 zenbakiaz zenbat aldiz zati daitekeen hondarrik sortzeke

Ariketa honetan osoa eta positiboa (≥ 1) den n zenbaki bat 2 zenbakiaz zenbat aldiz zati daitekeen (hondarrik sortu gabe) erabakitzen duen programa idatzi behar da.

Datu-eskatze prozesua datu egoki bat lortu arte errepikatu behar da:

```
#include <stdio.h>


/* Datuak: Programa honek erabiltzaileari positiboa eta osoa (>= 1) den
           zenbaki bat eskatuko dio. */

/* Emaitzak: Erabiltzaileak zenbaki egokia tekleatzen duenean, zenbaki hori 2
             zenbakiaz zenbat aldiz zati daitekeen kalkulatu eta emaitza hori
```

```
             aurkeztuko du. Datu-eskatze prozesua datu agokia lortu arte
             errepikatuko da. */

/* Aldagaiak:
          - n: erabiltzaileak tekleatuko duen zenbakia jasotzeko erabiliko da.

          - nlag: n-ren kopia gordetzeko erabiliko da.

          - kont: n zenbakia 2 zenbakiaz zenbat aldiz zati daitekeen
                  kontatzeko erabiliko da. */

void main()
 {
   int n, nlag, kont;


   printf("\nOsoa eta positiboa (>= 1) den  zenbaki bat tekleatu: ");
   scanf("%d", &n);

   while (n < 1)
    {
      printf("\nTekleatutako zenbakia ez da egokia.");
      printf("\nOsoa eta positiboa (>= 1) den  zenbaki bat tekleatu: ");
      scanf("%d", &n);
    }

   nlag = n; /* n-ren kopia bat sortuko dugu, gero eragiketak n-ren gainean
                egiterakoan n-ren hasierako balioa galdu egingo baita. */

   kont = 0; /* Une bakoitzean une horretara arte hasierako n balioa 2
                zenbakiaz zenbat aldiz zatitu dugun adieraziko digu. Hasieran
                oraindik ez dugunez n zenbakia 2 zenbakiaz zatitu, 0 balioa
                esleitu behar zaio. */

   while (n % 2 == 0)
    {
      kont = kont + 1;
      n = n / 2;
    }

   printf("\n%d zenbakia 2 zenbakiaz %d aldiz zati daiteke.", nlag, kont);

 }
```

### 3.1.6 Positiboa den n zenbakiaren zatitzaile guztiak aurkeztu baita zatitzaile hauen kopurua ere

Ariketa honetan osoa eta positiboa (n > 0) den n  zenbaki bat eskatu eta zenbaki horren zatitzaileak aurkeztu eta zatitzaile denak aurkeztu ondoren guztira zenbat zatitzaile aurkitu diren esaten duen programa idatzi behar da.

Datu-eskatze prozesua datu egoki bat lortu arte errepikatu behar da:

→

```
#include <stdio.h>


/* Datuak: Programa honek erabiltzaileari osoa eta positiboa (> 0) den zenbaki
           bat eskatuko dio. */

/* Emaitzak: Erabiltzaileak zenbaki egokia tekleatzen duenean, zenbaki horren
             zatitzaile denak aurkeztuko ditu eta guztira zenbat zatitzaile
             diren esango du. */

/* Aldagaiak:
          - n: erabiltzaileak tekleatuko duen zenbakia jasotzeko erabiliko da.
```

```
          - zenb: 1etik n-ra arteko zenbakiak banan-banan pasatzeko erabiliko
                  da. Une bakoitzean while-eko hurrengo bueltan aztertuko den
                  zenbakia edukiko dugu aldagai honetan. Zenbaki bat aztertzea
                  zenbaki horrek n zatitzen al duen begiratzea esan nahi du.

          - kont: une bakoitzean une horretara arte n-ren zenbat zatitzaile
                  aurkitu ditugun jakiteko erabiliko da. */

void main()
 {
   int n, zenb, kont;

   printf("\nOsoa eta positiboa (>= 1) den zenbaki bat tekleatu: ");
   scanf("%d", &n);

   while (n < 1)
    {
      printf("\nTekleatutako zenbakiak ez dira egokiak.");
      printf("\nOsoa eta positiboa (>= 1) den zenbaki bat tekleatu: ");
      scanf("%d", &n);
    }

   zenb = 1; /* 1etik n-ra arteko zenbakiak banan-banan pasatzeko erabiliko da
                eta 1etik hasi behar dugunez hasieran 1 balioa esleituko
                diogu. */

   kont = 0; /* Une bakoitzean une horretara arte n-ren zenbat zatitzaile
                aurkitu ditugun adieraziko digu. Hasieran oraindik ez dugunez
                n-ren zatitzailerik aurkitu, 0 balioa esleitu behar zaio. */

   printf("\n%d(r)en zatitzaileak honako hauek dira: ", n);

   while (zenb <= n)
    {
      if (n % zenb == 0)
       {
         printf("\n%d", zenb);
         kont = kont + 1;
       }
      zenb = zenb + 1;
    }

   printf("\n%d(e)k guztira %d zatitzaile ditu.", n, kont);
 }
```

### 3.1.7 Batukari baten kalkulua

Ariketa honetan osoa eta positiboa (> 0) den n zenbaki bat eskatu eta honako formula hau kalkulatzen duen programa idatzi behar da:

$$\sum_{i=1}^{n}((8*i)+2)$$

Datu-eskatze prozesua datu egoki bat lortu arte errepikatu behar da:

```
#include <stdio.h>


/* Datuak: Programa honek erabiltzaileari osoa eta positiboa (> 0) den zenbaki
           bat eskatuko dio. */

/* Emaitzak: Erabiltzaileak zenbaki egokia tekleatzen duenean, ((8 * 1) + 2) +
             ((8 * 2) + 2) + ((8 * 3) + 2) + ... + ((8 *n) + 2) batura
             kalkulatu eta aurkeztuko du. Datu-eskatze prozesua datu egokia
```

```
            lortu arte errepikatuko da. */

/* Aldagaiak:
          - n: erabiltzaileak tekleatuko duen zenbakia jasotzeko erabiliko da.

          - i: 1etik n-ra arteko zenbakiak banan-banan pasatzeko erabiliko da.
               Une bakoitzean while-eko hurrengo bueltan batu nahi den
               batugaia kalkulatzeko behar dugun formulako i balioa edukiko
               dugu aldagai honetan.

          - batura: batura kalkulatuz eta gordez joateko erabiliko da. */

void main()
 {
   int n, i, batura;

   printf("\nOsoa eta positiboa (>= 1) den zenbaki bat tekleatu: ");
   scanf("%d", &n);

   while (n < 1)
    {
      printf("\nTekleatutako zenbakiak ez dira egokiak.");
      printf("\nOsoa eta positiboa (>= 1) den zenbaki bat tekleatu: ");
      scanf("%d", &n);
    }

   i = 1; /* 1etik n-ra arteko zenbakiak banan-banan pasatzeko erabiliko da
             eta 1etik hasi behar dugunez hasieran 1 balioa esleituko diogu.
          */
   batura = 0; /* while-eko buelta bakoitzean batugai berri bat batuko diogu
                  aldagai honi, eta hasieran 0 balioa eduki behar du. */

   while (i <= n)
    {
      batura = batura + ((8 * i) + 2);
      i = i + 1;
    }

   printf("\nEmaitza: %d", batura);
 }
```

→

Jarraian programa honen exekuzio-adibidea azalduko dugu agindu bakoitzak aldagaiengan eta pantailan duen eragina adieraziz. Erabiltzaileak lehenengo -7 eta gero 3 balioak tekleatu dituela suposatuko dugu:

- int n, i, batura;

| ? | ? | ? |
|---|---|---|
| n | i | batura |

Pantaila

- printf("\nOsoa eta positiboa (>= 1) den zenbaki bat tekleatu: ");

| ? | ? | ? |
|---|---|---|
| n | i | batura |

Pantaila | Osoa eta positiboa (>= 1) den zenbaki bat tekleatu:

| |
|---|
| |

- scanf("%d", &n);

| -7 | ? | ? |
|---|---|---|
| n | i | batura |

Pantaila

| |
|---|
| Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7 |

- n < 1 al da? BAI

| -7 | ? | ? |
|---|---|---|
| n | i | batura |

Pantaila

| |
|---|
| Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7 |

- printf("\nTekleatutako zenbakia ez da egokia.");

| -7 | ? | ? |
|---|---|---|
| n | i | batura |

Pantaila

| |
|---|
| Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7<br>Tekleatutako zenbakia ez da egokia. |

- printf("\nOsoa eta positiboa (>= 1) den zenbaki bat tekleatu: ");

| -7 | ? | ? |
|---|---|---|
| n | i | batura |

Pantaila

| |
|---|
| Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7<br>Tekleatutako zenbakia ez da egokia.<br>Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: |

- scanf("%d", &n);

| 3 | ? | ? |
|---|---|---|
| n | i | batura |

Pantaila

| |
|---|
| Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7<br>Tekleatutako zenbakia ez da egokia.<br>Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: 3 |

- n < 1 betetzen al da? EZ, beraz while-a bukatu

| 3 | ? | ? |
|---|---|---|
| n | i | batura |

Pantaila

```
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7
Tekleatutako zenbakia ez da egokia.
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: 3
```

- i = 1;
  batura = 0;

| 3 | 1 | 0 |
|---|---|---|
| n | i | batura |

Pantaila

```
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7
Tekleatutako zenbakia ez da egokia.
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: 3
```

- i <= n betetzen al da? BAI

| 3 | 1 | 0 |
|---|---|---|
| n | i | batura |

Pantaila

```
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7
Tekleatutako zenbakia ez da egokia.
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: 3
```

- batura = batura + ((8 * i) + 2); (esleipen honetan i-ren balioa 1 da)
  i = i + 1;

| 3 | 2 | 10 |
|---|---|---|
| n | i | batura |

Pantaila

```
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7
Tekleatutako zenbakia ez da egokia.
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: 3
```

- i <= n betetzen al da? BAI

| 3 | 2 | 10 |
|---|---|---|
| n | i | batura |

Pantaila

```
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7
Tekleatutako zenbakia ez da egokia.
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: 3
```

- batura = batura + ((8 * i) + 2) ); (esleipen honetan i-ren balioa 2 da)
  i = i + 1;

| 3 | 3 | 28 |
|---|---|---|
| n | i | batura |

Pantaila

```
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7
Tekleatutako zenbakia ez da egokia.
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: 3
```

- i <= n betetzen al da? BAI

| 3 | 3 | 28 |
|---|---|---|
| n | i | batura |

Pantaila

```
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7
Tekleatutako zenbakia ez da egokia.
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: 3
```

- batura = batura + ((8 * i) + 2) ); (esleipen honetan i-ren balioa 3 da)
  i = i + 1;

| 3 | 4 | 54 |
|---|---|---|
| n | i | batura |

Pantaila

```
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7
Tekleatutako zenbakia ez da egokia.
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: 3
```

- i <= n betetzen al da? EZ, beraz while-a bukatu.

| 3 | 4 | 54 |
|---|---|---|
| n | i | batura |

Pantaila

```
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7
Tekleatutako zenbakia ez da egokia.
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: 3
```

- printf("\nEmaitza: %d", batura);

| 3 | 4 | 54 |
|---|---|---|
| n | i | batura |

Pantaila

```
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: -7
Tekleatutako zenbakia ez da egokia.
Osoa eta positiboa (>= 1) den zenbaki bat tekleatu: 3
Emaitza: 54
Sakatu tekla bat amaitzeko.
```

## 3.2 While baten barruan

While baten barruan edozein eragiketa mota izan dezakegu. Aurreko adibideetan esleipenak, baldintza aginteak edota eragiketa aritmetikoak agertu zaizkigu. Orain arte ez dugu ikusi adibide bat ere non while baten barruan beste while bat agertu daiteken. Hori ere posiblea da hurrengo adibideetan ikusiko dugun bezala.

### 3.2.1 While baten barruan beste while bat egon daiteke

Ariketa honetan ondoren agertzen dena pantailan aurkezten duen programa idatzi behar da:

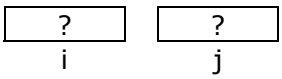
```
1 2 3 4
1 2 3 4
1 2 3 4
```

Lerroak batetik hirura pasatzeko while bat behar da, baina lerro bakoitzean gaudela lerroko elementuak, hau da, 1etik 4ra arteko zenbakiak pasatzeko beste while bat behar da. Beraz while baten barruan beste while bat ipini beharko da:

```
#include <stdio.h>


/* Datuak: Programa honek ez dio daturik eskatuko erabiltzaileari. */

/* Emaitzak: Pantailan hiru lerro berdin aurkeztuko ditu eta lerro bakoitzean
             honako hau agertuko da: 1  2  3  4. */

/* Aldagaiak:
          - i: 1etik 3ra arteko zenbakiak banan-banan pasatzeko eta une
               bakoitzean zein lerrotan gauden jakiteko erabiliko da.

          - j: 1etik 4ra arteko zenbakiak banan-banan pasatzeko eta une
               bakoitzean gauden lerroaren barruan zein zenbaki aurkeztea
               tokatzen zaigun jakiteko erabiliko da. */

void main()
 {
   int i, j;

   i = 1; /* Lehenengo lerroan gaudelako. */

   while (i <= 3)
    {
      j = 1; /* i lerroko lehenengo balioa */
      while (j <= 4)
       {
         printf("%d  ", j);
         j = j + 1;   /* i lerroan hurrengo balioa hartzeko */
       }
      i = i + 1; /* Lerroz aldatzeko. */
      printf("\n"); /* Pantailako hurrengo lerrora pasatzeko. */
    }

 }
```

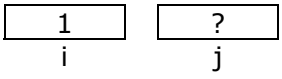
Jarraian programa honen exekuzio adibidea azalduko dugu agindu bakoitzak aldagaiengan eta pantailan duen eragina adieraziz:

- int i, j;



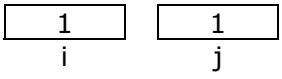

Pantaila

❖ i = 1;
i <= 3 al da? BAI



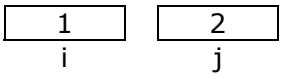

Pantaila

❖ j = 1;
j <= 4 al da? BAI




Pantaila

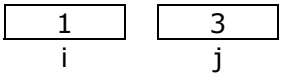
❖ printf("%d ", j); (une honetan j-ren balioa 1 da)
j = j + 1;




Pantaila

```
1
```

❖ j <= 4 al da? BAI (une honetan j-ren balioa 2 da)
printf("%d ", j);
j = j + 1;




Pantaila

```
1 2
```

→

- j <= 4 al da? BAI (une honetan j-ren balioa 3 da)
  printf("%d ", j);
  j = j + 1;

| 1 | 4 |
|---|---|
| i | j |

Pantaila

```
1  2  3
```

- j <= 4 al da? BAI (une honetan j-ren balioa 4 da)
  printf("%d ", j);
  j = j + 1;

| 1 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
```

- j <= 4 al da? EZ (barruko while-a bukatu da baina kanpokoak aurrera darrai)

| 1 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
```

- i = i + 1; (lerroz aldatu behar dugu)
  printf("\n"); (pantailan ere lerroz aldatu behar dugu)

| 2 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
```

- i <= 3 al da? BAI (kanpoko while-aren beste buelta bat hastera doa)

| 2 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
```

❖ j = 1;
j <= 4 al da? BAI

| 2 | 1 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
```

❖ printf("%d  ", j);  (une honetan j-ren balioa 1 da)
j = j + 1;

| 2 | 2 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1
```

❖ j <= 4 al da? BAI  (une honetan j-ren balioa 2 da)
printf("%d  ", j);
j = j + 1;

| 2 | 3 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2
```

❖ j <= 4 al da? BAI  (une honetan j-ren balioa 3 da)
printf("%d  ", j);
j = j + 1;

| 2 | 4 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3
```

❖ j <= 4 al da? BAI  (une honetan j-ren balioa 4 da)
printf("%d  ", j);
j = j + 1;

| 2 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
```

```
1  2  3  4
```

- j <= 4 al da? EZ (barruko while-a bukatu da baina kanpokoak aurrera darrai)

| 2 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3  4
```

- i = i + 1; (lerroz aldatu behar dugu)
  printf("\n"); (pantailan ere lerroz aldatu behar dugu)

| 3 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3  4
```

- i <= 3 al da? BAI (kanpoko while-aren beste buelta bat hastera doa)

| 3 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3  4
```

- j = 1;
  j <= 4 al da? BAI

| 3 | 1 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3  4
```

- printf("%d  ", j); (une honetan j-ren balioa 1 da)
  j = j + 1;

| 3 | 2 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3  4
```

```
1
```

- j <= 4 al da? BAI (puntu honetan j-ren balioa 2 da)
  printf("%d ", j);
  j = j + 1;

| 3 | 3 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3  4
1  2
```

- j <= 4 al da? BAI (une honetan j-ren balioa 3 da)
  printf("%d ", j);
  j = j + 1;

| 3 | 4 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3  4
1  2  3
```

- j <= 4 al da? BAI (une honetan j-ren balioa 4 da)
  printf("%d ", j);
  j = j + 1;

| 3 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3  4
1  2  3  4
```

- j <= 4 al da? EZ (barruko while-a bukatu da baina kanpokoak aurrera darrai)

| 3 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3  4
1  2  3  4
```

- i = i + 1; (lerroz aldatu behar dugu)
  printf("\n"); (pantailan ere lerroz aldatu behar dugu)

| 4 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3  4
1  2  3  4
```

- i <= 3 al da? EZ (kanpoko while-a bukatu da)
  printf("\nSakatu tekla bat amaitzeko.");
  getch();

| 4 | 5 |
|---|---|
| i | j |

Pantaila

```
1  2  3  4
1  2  3  4
1  2  3  4
Sakatu tekla bat amaitzeko.
```

Ariketa honetako exekuzio-adibidea jarraitzerakoan oso garrantzitsua da i aldagaiak hartzen duen balio bakoitzeko j aldagaiak 1etik 4ra arteko balio denak hartzen dituela konturatzea. Beste era batera esanda, i-ren balio bakoitzeko, i berdin mantetzen den bitartean j aldatuz joaten da (eta 5eraino iristen da). Ikusi ditzagun i eta j-ren balioak denboran zehar:

| i | j |
|---|---|
| ? | ? |
| 1 | ? |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |
| 1 | 4 |
| 1 | 5 |
| 2 | 5 |
| 2 | 1 |
| 2 | 2 |
| 2 | 3 |
| 2 | 4 |
| 2 | 5 |
| 3 | 5 |
| 3 | 1 |
| 3 | 2 |
| 3 | 3 |
| 3 | 4 |
| 3 | 5 |
| 4 | 5 |

### 3.2.2 While baten barruan beste bi while independente ere egon daitezke

Ariketa honetan erabiltzaileari positiboa eta osoa (≥ 1) den n zenbaki bat eskatu eta pantailan honako hau aurkeztuko duen programa idatzi behar da:

```
1
1  2  1
1  2  3  2  1
...
1  2  3  4  5  6  7  ......  n  ......  7  6  5  4  3  2  1
```

←gorazko ordena → ←beheranzko ordena →

Lerroak 1etik n-ra pasatzeko while bat behar da, baina lerro bakoitzean gaudela lerroko elementuak, hau da, 1etik lerro-zenbakira arteko zenbakiak pasatzeko beste while bat behar da eta lerro-zenbakitik 1erainoko zenbakiak pasatzeko beste while bat behar da. Beraz while baten barruan beste bi while independente ipini beharko dira.

Datu-eskatze prozesua datu egoki bat lortu arte errepikatu behar da:

```
#include <stdio.h>


/* Datuak: Programa honek erabiltzaileari osoa eta positiboa (>= 1) den
           zenbaki bat eskatuko dio, datu-eskatze prozesua zenbaki egoki bat
           lortu arte errepikatuz. */

/* Emaitzak: Datu egokia emandakoan pantailan n lerro aurkeztuko ditu, eta
             lerro bakoitzean 1etik lerroaren zenbakiak adierazten duen
             zenbakira arte eta zenbaki horretatik 1erainokoak aurkeztuko
             dira:
                  1
                  1  2  1
                  ...
                  1  2  3  ...  n  ...  3  2  1 */

/* Aldagaiak:
          - n: erabiltzaileak tekleatutako zenbakia gordetzeko erabiliko da.

          - i: 1etik n-ra arteko zenbakiak banan-banan pasatzeko eta une
               bakoitzean zein lerrotan gauden jakiteko erabiliko da.

          - j: 1etik i-ra arteko zenbakiak eta i-tik 1era arteko zenbakiak
               banan-banan pasatzeko eta une bakoitzean gauden lerroaren
               barruan zein zenbaki aurkeztea tokatzen zaigun jakiteko
               erabiliko da. */
void main()
 {
   int n, i, j;

   printf("\nOsoa eta positiboa (>= 1) den zenbaki bat tekleatu: ");
   scanf("%d", &n);

   while (n <= 0)
    {
      printf("\nTekleatutako zenbakia ez da egokia.");
      printf("\nOsoa eta positiboa (>= 1) den zenbaki bat tekleatu: ");
      scanf("%d", &n);
    }

   i = 1; /* i aldagaiak une bakoitzean zein lerrotan gauden adieraziko digu.
             Hasieran lehenengo lerroan gaude. */

   while (i <= n)
    {
   /*zenbaki (edo i) bakoitzeko 1etik hasita gorazko ordena sortzeko while-a*/
      j = 1; /* i lerroko lehenengo balioa */
      while (j <= i)
       {
         printf("%d  ", j);
         j = j + 1; /* i lerroan hurrengo balioa hartzeko */
       }

      /* Lerroaren erdia aurkeztu dugu eta orain beste erdia aurkeztu behar
         da. */
```

```
 .... /*Orain zenbakitik (i-1ren baliotik)hasita eta 1arte bererazko ordena
......sortzeko while-a*/

      j = j - 2; /* (i - 1)-etik 1eraino joateko. */

      while (j >= 1)
       {
         printf("%d  ", j);
         j = j - 1; /* i lerroan hurrengo balioa hartzeko */
       }

      i = i + 1; /* Lerroz aldatzeko. */
      printf("\n"); /* Pantailako hurrengo lerrora pasatzeko. */
    }
 }
```

Soluzio honetan while baten barruan bi while daude, baina barruko bi while horiek elkarren artean independenteak dira. Barruko whilen egikariketa bukatu arte while handiaren hurrengo bueltan ez gara sartzen.

While baten barruan beste while baten barruko beste adibide bat

### 3.2.3 Batukari bikoitza

Ariketa honetan erabiltzaileari positiboak eta osoak (≥ 0) diren m eta n bi zenbaki eskatu eta honako formula hau kalkulatu eta pantailan aurkeztuko duen programa idatzi behar da:

$$\sum_{i=0}^{n}\sum_{j=0}^{m}(i*j)$$

Datu-eskatze prozesua datu egokiak lortu arte errepikatu behar da:

```
#include <stdio.h>

/* Programaren izena: P6.64 */

/* Datuak: Programa honek erabiltzaileari osoak eta positiboak (>= 0) diren bi
           zenbaki eskatuko dizkio, datu-eskatze prozesua zenbaki egokiak
           lortu arte errepikatuz. */

/* Emaitzak: Pantailan honako batuketa honen emaitza aurkeztuko da:

              (0 * 0) + (0 * 1) + (0 * 2) + ... + (0 * m) +
              + (1 * 0) + (1 * 1) + (1 * 2) + ... + (1 * m) +
              + (2 * 0) + (2 * 1) + (2 * 2) + ... + (2 * m) +
              + ... +
              + (n * 0) + (n * 1) + (n * 2) + ... + (n * m) +
*/

/* Aldagaiak:
          - m, n: erabiltzaileak tekleatuko dituen bi zenbakiak gordetzeko
                  erabiliko dira.

          - batura: programak kalkulatuz joan behar duen batura gordetzeko
                    erabiliko da.

          - i: 1etik n-ra arteko zenbakiak banan-banan pasatzeko erabiliko da.

          - j: 1etik m-ra arteko zenbakiak banan-banan pasatzeko erabiliko da.
```

```
*/

void main()
 {
   int m, n, batura, i, j;

   printf("\nOsoak eta positiboak (>= 0) diren bi zenbaki (n eta m) "
          "tekleatu komaz bereiztuta: ");
   scanf("%d, %d", &n, &m);

   while ((n < 0) || (m < 0))
    {
      printf("\nTekleatutako zenbakiak ez dira egokiak.");
      printf("\nOsoak eta positiboak (>= 0) diren bi zenbaki (n eta m) "
             "tekleatu komaz bereiztuta: ");
      scanf("%d, %d", &n, &m);
    }

   batura = 0; /* batura urratsez urrats kalkulatuko da, eta urrats bakoitzean
                  aurreko urratsera arte kalkulatutako baturari elementu berri
                  bat batuko zaio. Horregatik lehenengo elementua batzerakoan
                  aurretik 0 edukitzea komeni zaigu. */

   i = 0; /* i aldagaiak 0tik n-ra arteko balioak banan-banan hartuz joan
             behar du. */

   while (??????????????????????????)
    {
      j = ????; /* i ren balio bakoitzeko j aldagaiak 0tik m-ra arteko balioak
                   pasatu behar ditu. */
      while (????????????????????)
       {
         batura = batura + (i * j);
         j = ????????; /* i mantenduz eta j aldatuz joan behar da */
       }
      i =?????????????;
      printf("\nn = %d eta m = %d balioentzat formularen balioa honako "
             "hau da: %d", n, m, batura);
    }

 }
```

# 4 DO-WHILE AGINDUA

- do-while aginduaren **itxura**:

```
do
  {
    aginduak
  }
while
(baldintza);
```

- do-while aginduaren **esanahia**:

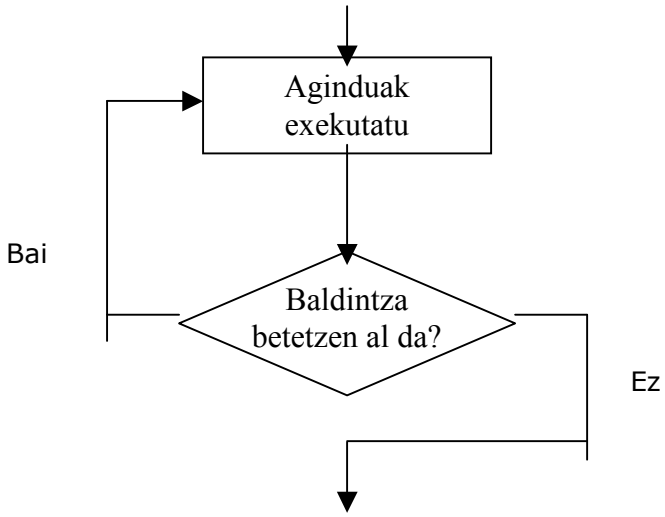
  Aginduak behin eta berriz burutu **baldintza egiazkoa den bitartean**. Do-while agindua baldintza faltsua izatera pasatzen denean bukatuko da.

  Do-while agindu batera iristen garenean, hasteko giltza arteko aginduak exekutatuko dira, eta gero baldintza betetzen al den ala ez aztertuko da. Betetzen bada, giltza arteko agindumultzoa berriro exekutatuko da eta berriro baldintza aztertuko

da. Baldintza berriro aztertutakoan egiazkoa bada, aginduak berriz exekutatu eta baldintza aztertzera joan beharko da berriz ere. Prozesu hori behin eta berriz errepikatuko da baldintza egiazkoa den bitartean. Beraz, baldintza betetzen bada, aginduak exekutatu eta baldintzara itzuli prozesua errepikatzeko. Baldintza ez bada betetzen, aginduak exekutatu gabe do-while-etik irten.

Do-while agindua ulertzeko beste era bat honako hau da: agindu-multzoa exekutatu behar da **baldintza faltsua izan arte**.

- do-while aginduaren **esanahia grafikoki**:




- do-while euskeraz:

**egin** {
Errepikatu behar den eragiketa multzoa
}(baldintza) **betetzen den bitartean;**

- do-while aginduaren erabilera:

Do-while errepikatu behar dena **gutxienez behin** burutu behar dela dakigun kasuetarako da egokia. Adibidez erabiltzaileari datuak eskatzeko eta datu-eskatze prozesua datu egokiak lortu arte errepikatzeko. Baita menu bat aurkeztu, erabiltzaileari menuko aukera bat hautatzeko eskatu, hautatutako aukerari dagokiona burutu eta berriro menua aurkeztu behar denerako ere erabil daiteke.

## 4.1 do-while eta while:

Konparatu dezagun oraintxe bertan aipatu dugun kasuaren adibide konkretu bat, hau da, erabiltzaileari zenbaki bat eskatu eta zenbaki-eskatze prozesua zenbakia [1..10] tartean egon arte errepikatzea bai do-while batekin, eta baita while batekin inplementatuta.

While batekin

```
#include <stdio.h>
void main(){
int zenb;

printf("Sartu [1..10] tarte barruko zenbaki bat: ");
scanf("%d",&zenb);
while((zenb <1)||(zenb>10)){
       printf("Sartu [1..10] tarte barruko zenbaki bat: ");
       scanf("%d",&zenb);
}
```

Do-while batekin

```
#include <stdio.h>
void main(){
int zenb;

do{
       printf("Sartu [1..10] tarte barruko zenbaki bat: ");
       scanf("%d",&zenb);
   }while((zenb <1)||(zenb>10));
}
```

Do-while-ekin adieraz daitekeen edozer while-ekin ere adieraz daiteke. While-ean hasteko baldintza aztertzen da eta gero aginduak burutzen dira baldintza egiazkoa bada. Do-while-ean hasteko aginduak behin burutzen dira eta gero baldintza aztertzen da. Baldintza egiazkoa bada, aginduak berriro exekutatuko dira.

While-ean gerta daiteke aginduak behin ere ez exekutatzea, izan ere, while-era iristen garenean eta baldintza lehenengo aldiz aztertzen dugunean baldintza ez bada betetzen, aginduak ez dira baten ere exekutatuko.
Do-while-ean aginduak gutxienez behin exekutatzen dira, izan ere, do-while-era iristen garenean baldintza lehenengo aldiz aztertu baino lehen agindu-multzoa behin exekutatzen da.

Desberdintasuna hobeto ulertzeko beste adibide bat erabiliko dugu:

Programa bat idatzi 2tik hasita zenbaki bikoiti guztiak idazteko erabiltzaileak zenbaki bikoiti gehiago ez dituela kalkulatu nahi adierazi arte.

While

```
#include <stdio.h>
void main(){
int zenb;
char erantzuna;

zenb=2;
printf("%d bikoitia da\n", zenb);
printf ("Hurrengo zenbaki bikoitia ikusi nahi duzu?(b/e) ");
fflush(stdin);
scanf ("%c",&erantzuna);

while(erantzuna=='b'){
       zenb=zenb+2;
       printf("%d bikoitia da\n", zenb);
       printf ("Hurrengo zenbaki bikoitia ikusi nahi duzu?(b/e) ");
       fflush(stdin);
       scanf ("%c",&erantzuna);
}
```

Do-while batekin

```
#include <stdio.h>
void main(){
int zenb;
char erantzuna;

zenb=2;
do{
       printf("%d bikoitia da\n", zenb);
       zenb=zenb+2;
       printf ("Hurrengo zenbaki bikoitia ikusi nahi duzu?(b/e) ");
       fflush(stdin);
       scanf ("%c",&erantzuna);
}while(erantzuna=='b');
```

# 5 FOR AGINDUA

- for aginduaren itxura:

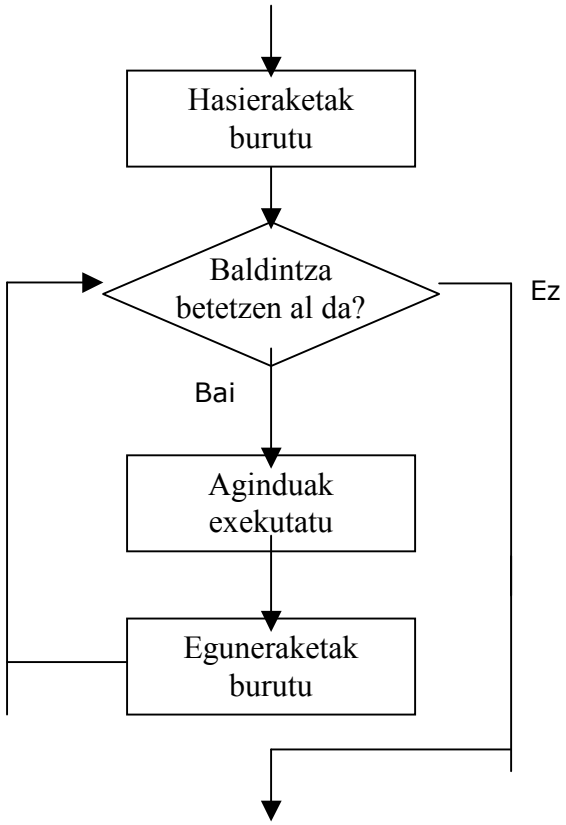
```
for (hasieraketak; baldintza;
eguneraketak)
  {
    aginduak
  }
```

- for aginduaren esanahia:

  Hasteko hasieraketak burutu behar dira. Hasieraketak behin bakarrik exekutatuko dira, ez dira errepikatuko. Jarraian baldintza egiazkoa den **bitartean** aginduak eta eguneraketak exekutatuko dira behin eta berriz. Baldintza faltsua izatera pasatzen denean bukatuko da for agindua.

For agindua beste era honetara ere uler daiteke: hasieraketak behin exekutatu eta agindu-multzoa eta eguneraketak baldintza faltsua izan arte errepikatu.

- for aginduaren esanahia grafikoki:




## 5.1 for eta while:

C lengoaian for-ekin adieraz daitekeen edozer while-ekin ere adieraz daiteke.

For-etik while-erako itzulpena zuzenean buru daiteke ondoko eskema jarraituz:

```
for (hasieraketak; baldintza; eguneraketak)
  {
    aginduak
  }
```

```
hasieraketak
while (baldintza)
  {
    aginduak
    eguneraketak
  }
```

BAINA KONTUZ!!! **Guk** for[2] bat bakarrik erabiliko dugu iterazio kopurua ezaguna denean. Nahiz eta Cz while eta for baliokideak izan beste programazio lengoai batzuetan ez da horrela, hori dela eta esan bezala **Guk** for bat bakarrik erabiliko dugu iterazio kopurua ezaguna denean

Adibidez esaten badigute erabiltzaileari zenbakiak eskatzeko erabiltzaileak 0 bat tekleatu arte eta bakarrik inprimatzea erabiltzaileak sartzen duen zenbakia bikoitia balitz. Ba al dakigu zenbat zenbaki sartuko dituen erabiltzaileak? Guk ezin dugu jakin erabiltzaileak noiz tekleatuko duen 0a. Kasu hauetan ez genuke for bat erabiliko baizik eta while edo do-while bat.

Aldiz esango baligute erabiltzaileari 50 zenbaki eskatzeko eta bakarrik inprimatzea erabiltzaileak sartzen duen zenbakia bikoitiak. Kasu honetan badakigu zein den iterazio kopurua (50 iterazio).Kasu hauetan for bat erabili dezakegu.

Orain eta whilearentzat planteatutako adibide bat for-aren bitartez ebatziko dugu. Ariketa hau ebazteko while bat beste while baten barruan erabili dugu. Orain ebatziko dugu for bat beste for baten barruan erabilita.

Ariketa honetan ondoren agertzen dena pantailan aurkezten duen programa idatzi behar da:

```
1 2 3 4
1 2 3 4
1 2 3 4
```

Gogoratu, lerroak batetik hirura pasatzeko while bat erabili da, eta lerro bakoitzean gaudela lerroko elementuak, hau da, 1etik 4ra arteko zenbakiak pasatzeko beste while bat erabili da. Beraz while baten barruan beste while bat ipini dugu lehenago:

```
#include <stdio.h>

/* Datuak: Programa honek ez dio daturik eskatuko erabiltzaileari. */

/* Emaitzak: Pantailan hiru lerro berdin aurkeztuko ditu eta lerro bakoitzean
             honako hau agertuko da: 1  2  3  4. */

/* Aldagaiak:
          - i: 1etik 3ra arteko zenbakiak banan-banan pasatzeko eta une
               bakoitzean zein lerrotan gauden jakiteko erabiliko da.

          - j: 1etik 4ra arteko zenbakiak banan-banan pasatzeko eta une
               bakoitzean gauden lerroaren barruan zein zenbaki aurkeztea
               tokatzen zaigun jakiteko erabiliko da. */
void main()
 {
   int i, j;

   i = 1; /* Lehenengo lerroan gaudelako. */

   while (i <= 3)
    {
       j = 1; /* i lerroko lehenengo balioa */
```

[2] for aginduaren ezaugarri bitxi batzuk dauzka Cz:

Hasieraketarik, baldintzarik edo eguneraketarik ez duten for-ak idatz daitezke. Baldintzarik ez duen for-a infitu aldiz, hau da, amaierarik gabe exekutatuko da. Hasieraketa eta eguneraketari dagozkien ataletan esleipenez gain printf eta scanf aginduak ere ipin daitezke. Gainera bai hasieraketa eta bai eguneraketa ataletan agindu bat baino gehiago ipin daitezke.

```
            while (j <= 4)
            {
              printf("%d  ", j);
              j = j + 1;   /* i lerroan hurrengo balioa hartzeko */
            }
      i = i + 1; /* Lerroz aldatzeko. */
        printf("\n"); /* Pantailako hurrengo lerrora pasatzeko. */
     }

 }
```

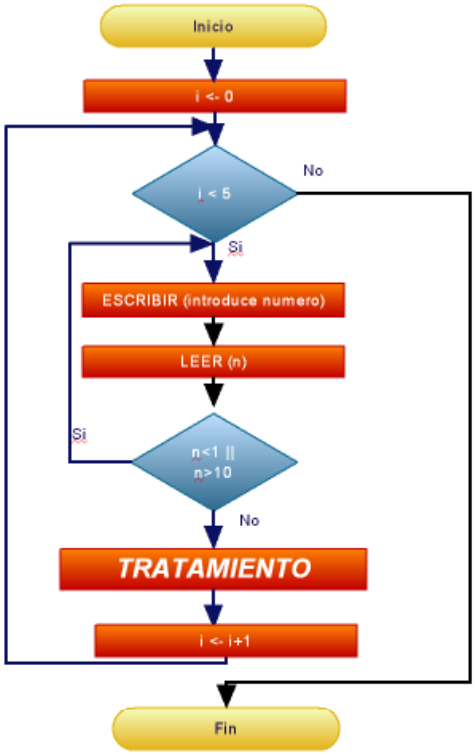
Forrekin berdina egiteko kodea hau izango litzateke

```
#include <stdio.h>

/* Datuak: Programa honek ez dio daturik eskatuko erabiltzaileari. */

/* Emaitzak: Pantailan hiru lerro berdin aurkeztuko ditu eta lerro bakoitzean
             honako hau agertuko da: 1  2  3  4. */

/* Aldagaiak:
          - i: 1etik 3ra arteko zenbakiak banan-banan pasatzeko.

       -   j: lerro bakoitzean 1etik 4ra arteko zenbakiak banan-banan
       -   pasatzeko*/
void main()
 {
   int i, j;

   for (i = 1; i <= 3; i=i+1)
    {
       for (j = 1; j <= 4; j=j+1)
       {
         printf("%d  ", j);
       }
      printf("\n"); /* Pantailako hurrengo lerrora pasatzeko. */
    }
 }
```

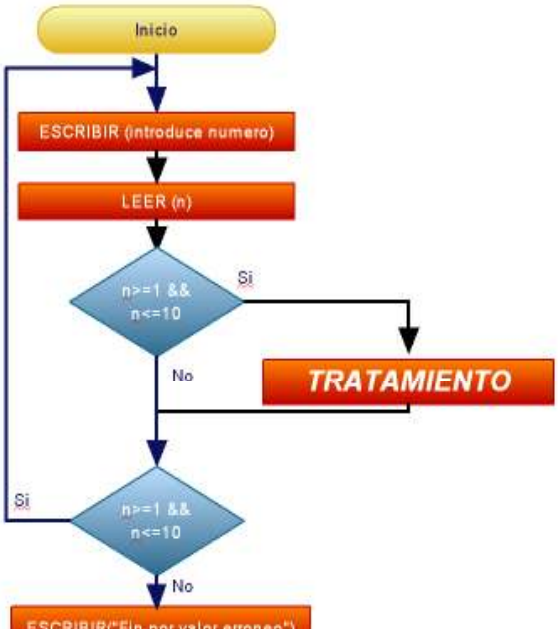
# 6 1. GEHIAGARRIA: FLUXU DIAGRAMA BATZUK

**1-Pedir un número y validar que esté entre 1 y 10 y luego tratar.**

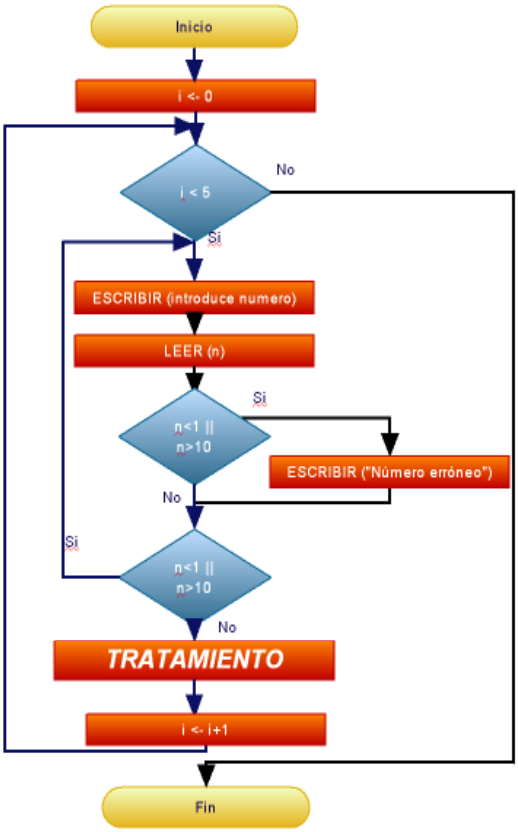

**2.Pedir 5 números validos que estén entre 1 y 10 tratando cada uno**



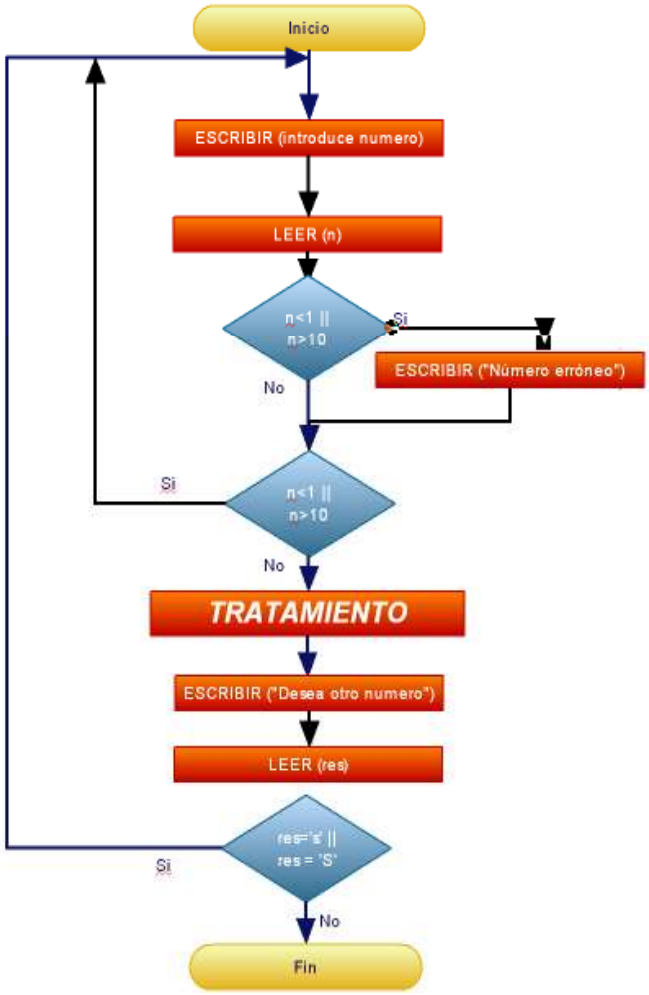
## 3.- Pedir números validos que estén entre 1 y 10 tratando cada uno y finalizando cuando el valor sea no válido



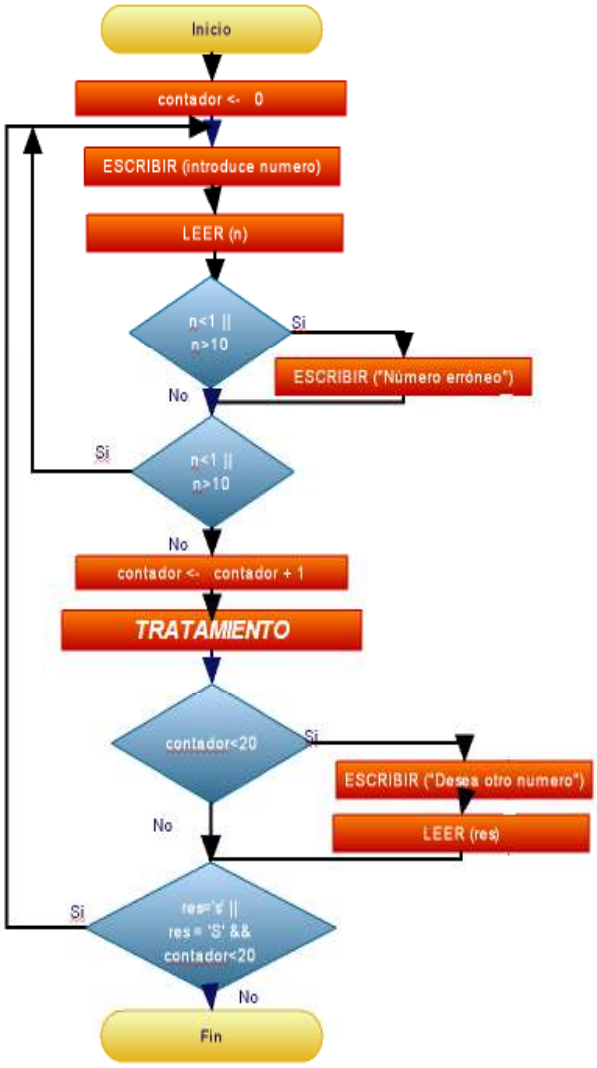

## 4.-Pedir 5 números validos que estén entre 1 y 10 tratando cada uno e incluyendo mensaje error cuando el valor no es válido



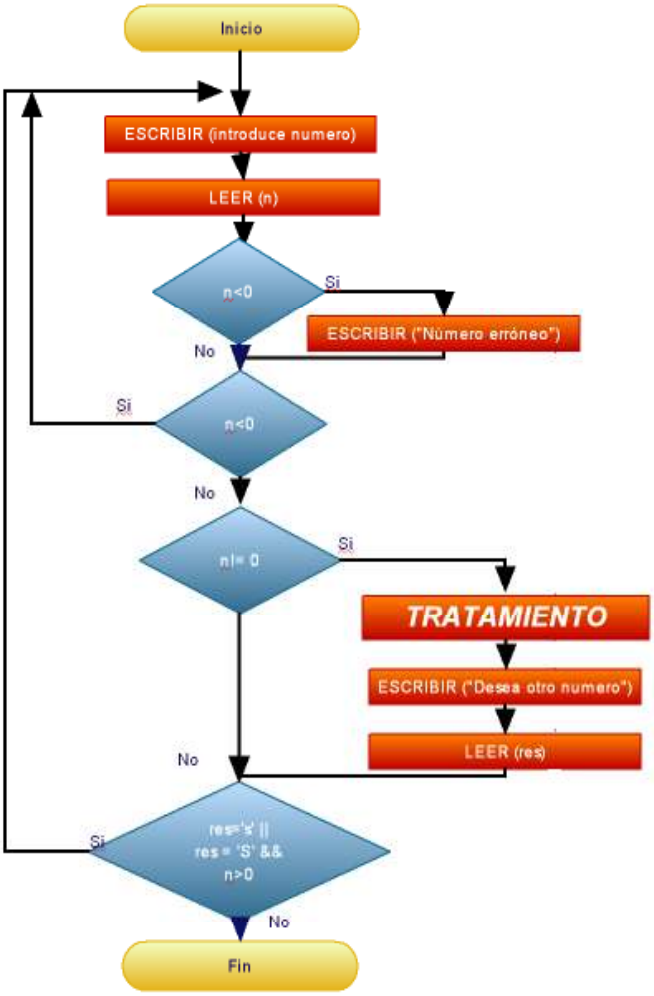
**5.-Pedir números validos que estén entre 1 y 10 tratando cada uno e incluyendo mensaje error cuando el valor no es válido. Repetir hasta que el usuario decida no introducir más números con pregunta (s/n).**



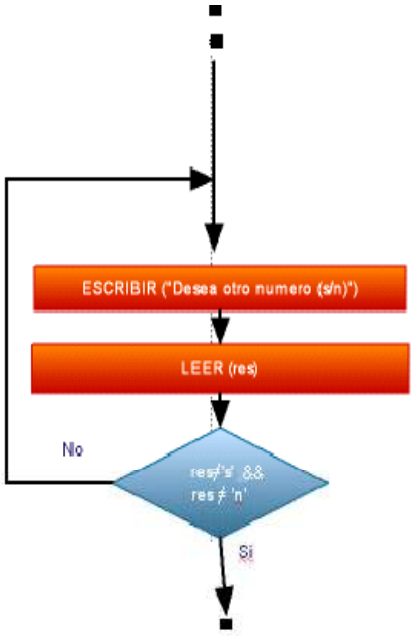
**6.-Pedir números validos que estén entre 1 y 10 tratando cada uno e incluyendo mensaje error cuando el valor no es válido. Repetir hasta que el usuario decida no introducir más números con pregunta (s/n) o hasta que haya introducido 20 válidos.**



**7.- Pedir números validos que sean positivos tratando cada uno e incluyendo mensaje error cuando el valor no es válido. El 0 indica fin de datos y no se trata. Repetir hasta que el usuario decida no introducir más números con pregunta (s/n) o que haya introducido un 0.**



## 8.- Solicitar respuesta del usuario hasta que sea ‘s’ o ‘n’ . (A Añadir en cualquiera de las anteriores)




9.- Programan bitartez definitu constante bat eta joan eskatzen erabiltzaileari zenbakiak, hauek euren artean batuz. Programa bukatuko da batuketa hau ezarritako balio kontantea gainditzen duen momentuan, batuketaren balio pantailatik idatziz.

10.-1etik hasita eta erabiltzaileak sartutako muga baterarte pantailaratzen joan zenbakia eta bere balio karratua bi zutabeetan.

11.-Pantailan atera letra minuskula eta maiuskulei dagokien ASCII kodeak.

12.-Lehenengo 5 gramu 20 euro eta hortik aurrera hurrengo 5 gramu 12 euro gehiago balio dutela jakinda, lortu 5 gramutik hasita (5naka gehituz) eta 60 gramurarte pisu bakoitzaren balioa.

| | |
|---|---|
| 5 gramu | 20 euro |
| 10 gramu | 32 euro |
| .... | |

13.-Joan sartzen karakterak puntu bat sartu arte kontrolatuz eta kontatuz zenbat zurigune sartu diren.

14.-Puntu batez bukatutako karaktere sekuentzia bat emanda kalkulatu zenbat bider agertu den 'A' karakterea.

15.-Puntu batez bukatutako karaktere sekuentzia bat emanda kalkulatu zenbat bokal agertu diren.

16.-Puntu batez bukatutako karaktere sekuentzia bat emanda kalkulatu zenbat bokal, zenbat ez-bokal eta guztira zenbat karaktere agertu diren.