6. GAIA
TAULA NUMERIKOAK / BEKTOREAK

1 BEKTOREAK: DEFINIZIOA ETA ERABILERA

1.1 Definizioa eta erabilera

Taula bat mota bereko aldagai edo osagai multzo batez osatuta egoten da. Dimentsio bakarreko taula bat definitzeko osagaien mota, taularen izena eta osagai-kopurua zehaztu behar dira. Taulek mota bereko aldagai asko definitzeko eta maneiatzeko bide egokia eta erraza eskaintzen digute.

Char motako osagaiak dituzten taulek ezaugarri bereziak dituztenez, gai honetan zenbakizko taulak bakarrik aztertuko ditugu. Beraz int eta float motako osagaiak dituzten taulak bakarrik azalduko dira gai honetan.

Dimentsio bakarreko zenbakizko taulei **bektore** ere deitzen zaie.

Bektore bateko osagai bakoitza **indizea** bezala ezagutzen den zenbaki baten bidez identifikatzen da. Bektore batek N osagai baldin baditu, lehenengo osagaiaren indizea 0 izango da eta azkenarena N – 1.

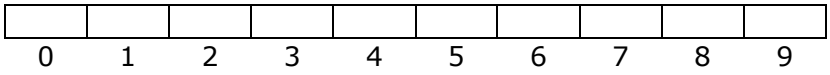
1. adibidea:

Demagun int motako 10 osagai dituen eta t izena duen bektore bat definitu nahi dugula. Honako hau idatzi beharko genuke:

```
int t[10];
```

Definizio horren bidez honako taula hau sortuko litzateke:

t



0 1 2 3 4 5 6 7 8 9

Taula bat definitzen denean, balio ezezagunak izango ditu.

t taulako osagai bakoitza int motako aldagai normal bat bezala erabil daiteke.

2. adibidea:

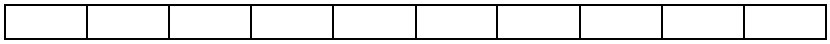
Adibide honetan bektoreak nola erabiltzen diren azalduko da.

Demagun int motako a aldagaia eta int motako 10 osagai dituen t taula definitu ditugula honako hau idatziz:

```
int a;
int t[10];
```

a

t



Hasieran bai a aldagaiak eta bai t taularen osagaiek, balio ezezaguna izango dute.

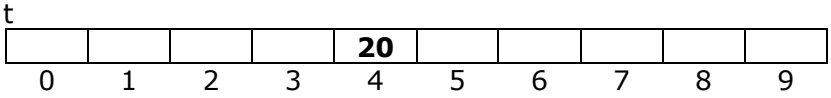
Demagun orain t taularen 4 osagaiari 20 balioa esleitu nahi diogula. Hori honela egingo genuke:

```
t[4] = 20;
```

Ondorio bezala honako hau edukiko genuke:

a

t



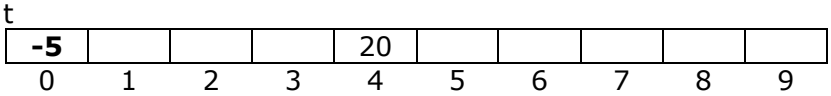
Orain 0 posizioko osagaiari -5 balioa esleitzeko honako hau idatzi beharko genuke:

```
t[0] = -5;
```

Ondorio bezala honako hau edukiko genuke:

a

t



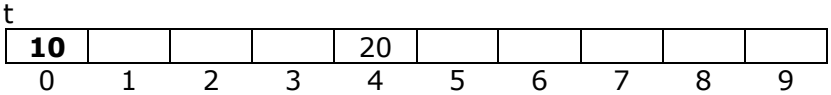
Ondoren 0 posizioko osagaian 4 posizioan dagoena gehi 0 posizioan dagoena bider 2 balioa gordeko dugu:

```
t[0] = t[4] + (t[0] * 2);
```

Ondorio bezala honako hau edukiko genuke:

a

t



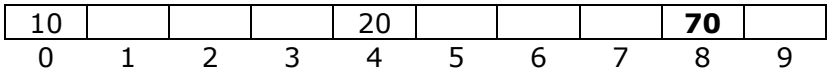
Jarraian a aldagaiari 7 balioa esleituko diogu eta gero t taulako a + 1 posizioan, 0 posizioan dagoena bider a gordeko dugu:

```
a = 7;
t[a + 1] = t[0] * a;
```

Ondorio bezala honako hau edukiko genuke:

a

t



1.2 Bektoreak eta funtzioak

Bektore bat funtzio baten argumentu bezala ipintzen denean, ez da zehaztu behar bektorearen osagai-kopurua.

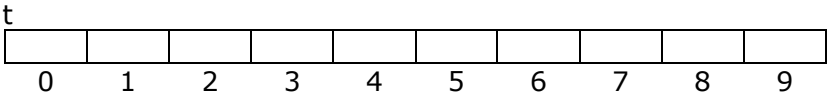
Funtzio batek bere argumentua diren bektoreen osagaien balioak alda ditzake.

2 ADIBIDEAK

2.1 t taula nola bete erabiltzaileari zenbakiak eskatuz

Demagun 10 osagaiko taula bat definitu dugula eta orain taula hori zenbakiz bete nahi dugula erabiltzaileari zenbakiak tekleatzeko eskatuz. Bi era desberdin azalduko dira.

int t[10];



a) Lehenengo bertsioan osagaiak banan-banan eskatuko zaizkio erabiltzaileari.

```
int i, zenb;

i = 0; /* i aldagaia t taularen indize bezala erabiliko da. Une
          bakoitzean t taulan libre dagoen lehenengo posizioa zein den
          adieraziko du i aldagaiak. Gainera une bakoitzean une
          horretara arte zenbat zenbaki eskatu diren jakiteko ere
          erabiliko da. */

while (i < 10)
 {
   printf("\nSartu %d posiziorako zenbaki oso bat: ", i);
   scanf("%d", &zenb);
   t[i] = zenb;
   i = i + 1;
 }
```

Bertsio honetan osagai bakoitza eskatzeko mezu bat aurkeztuko zaio erabiltzaileari. Osagai bakoitza tekleatzen duenean return sakatu beharko du. Ondoren hurrengo osagaia eskatuko zaio eta abar.

Erabiltzaileari zenbakiak eskatzerakoan pantailan honako hau ikusiko genuke:

Pantaila

```
Sartu 0 posiziorako zenbaki oso bat: 10
Sartu 1 posiziorako zenbaki oso bat: -8
Sartu 2 posiziorako zenbaki oso bat: 29
Sartu 3 posiziorako zenbaki oso bat: 7
Sartu 4 posiziorako zenbaki oso bat: 20
Sartu 5 posiziorako zenbaki oso bat: 34
Sartu 6 posiziorako zenbaki oso bat: -8
Sartu 7 posiziorako zenbaki oso bat: 22
Sartu 8 posiziorako zenbaki oso bat: 1
Sartu 9 posiziorako zenbaki oso bat: 19
```

Taula honela geldituko litzateke:

t

10	-8	29	7	20	34	-8	22	1	19
0	1	2	3	4	5	6	7	8	9

b) Bigarren bertsioan erabiltzaileak osagai denak jarraian (zuriune batez bananduta) tekleatu beharko ditu.

```
int i, zenb;

i = 0; /* i aldagaia t taularen indize bezala erabiliko da. Une
          bakoitzean t taulan libre dagoen lehenengo posizioa zein den
          adieraziko du i aldagaiak. Gainera une bakoitzean une
          horretara arte zenbat zenbaki eskatu diren jakiteko ere
          erabiliko da. */

printf("\nSartu taulako 10 osagaiak zuriunez bananduta eta azkenean "
       "return sakatu: ");

while (i < 10)
 {
   scanf("%d", &zenb);
   t[i] = zenb;
   i = i + 1;
 }
```

Bertsio honetan osagaiak eskatzeko mezu bakarra aurkeztuko zaio erabiltzaileari. Osagai denak zuriune batez bananduta tekleatu ondoren return sakatu beharko da. Beraz kasu honetan return behin bakarrik sakatu behar da, bukaeran.

Erabiltzaileari zenbakiak eskatzerakoan pantailan honako hau ikusiko genuke:

Pantaila

```
Sartu taulako 10 osagaiak zuriunez bananduta eta azkenean
return sakatu: 10 -8 29 7 20 34 -8 22 1 19
```

Taula honela geldituko litzateke:

t

10	-8	29	7	20	34	-8	22	1	19
0	1	2	3	4	5	6	7	8	9

2.2 t taulako lehenengo z elementuak batzen dituen funtzioa

Funtzio honek t taulako 0 posiziotik z – 1 posiziora arteko osagaien batura kalkulatu eta itzuliko du.

- Funtzioaren prototipoa:

```
int kalkulatu_osagaien_batura (int z, int t[]);
```

Funtzioaren prototipoan eta definizioan ez da beharrezkoa bektorearen osagai-kopurua zehaztea. Horregatik t[ ] ipini da. Izan ere taulak zenbat osagai dituen jakitea ez da garrantzitsuena, garrantzitsuena zenbat osagai batu behar diren jakitea da eta hori z parametroak adieraziko digu.

- Funtzioaren definizioa:

```
int kalkulatu_osagaien_batura (int z, int t[])

 /* Funtzio honek t taulako lehenengo z osagaien batura kalkulatu eta
    batura hori itzuliko du. */

 { /* Funtzioaren hasierako giltza. */
   int batura, i;

   batura = 0; /* taulako elementuen batura gordetzeko erabiliko da */

   i = 0; /* i aldagaia t taularen indize bezala erabiliko da. Izan ere, t
             taula posizioz posizio zeharkatu behar da eta une bakoitzean t
             taulako zein posiziotan gauden adieraziko du i aldagaiak. */

   while (i < z)
    {
      batura = batura + t[i];
      i = i + 1;
    }
   return(batura);
 } /* Funtzioaren bukaerako giltza. */
```

- Funtzioaren erabilera:

  Demagun 10 osagai dituen b taula definitu dugula eta dagoeneko taula bete dugula:

  int b[10];

b

10	-8	29	7	20	34	-8	29	7	20
0	1	2	3	4	5	6	7	8	9

  b taulako lehenengo 4 osagaien batura kalkulatu eta v aldagaian gordetzea nahi izango bagenu, honako hau idatzi beharko genuke:

```
v = kalkulatu_osagaien_batura (4, b);
```

  Agindu horren ondorio bezala v aldagaian 38 balioa gordeko litzateke.

  Adibide honetan ere ikusi den bezala, funtzio bat erabiltzerakoan funtzioen argumentuak diren tauletan osagai-kopurua ez da zehaztu behar eta [ ] ere ez da ipini behar.

2.3 t1 taulako lehenengo z elementuak t2 taulan inkrementatuta gordetzen dituen funtzioa

Funtzio honek t1 taulako 0 posiziotik z – 1 posiziora arteko elementuak inkrementatuta t2 taulako pareko posizioan gordeko ditu. Beraz adibidez t2 taulako 3 posizioan t1 taulako 3 posizioko balioa gehi 1 gordeko du.

- Funtzioaren prototipoa:

```
void kopiatu_inkrementatuz (int z, int t1[], int t2[]);
```

Kasu honetan z eta t1 datuak dira eta t2 emaitza da. Emaitza bezala taula bat edukitzea emaitza multzo bat edukitzearen parekoa da. Horregatik funtzio hau 5. motakoa da.

Kasu honetan t2 taula funtzioaren emaitza da, baina emaitza izan arren taula denez ez da * edo & ipini behar.

- Funtzioaren definizioa:

```
void kopiatu_inkrementatuz (int z, int t1[], int t2[])

 /* Funtzio honek t1 taulako lehenengo z osagaiak inkrementatuz t2 taulan
    kopiatuko ditu. */

 {
   int i;

   i = 0; /* i aldagaia t1 eta t2 taulen indize bezala erabiliko da. */

   while (i < z)
    {
      t2[i] = t1[i] + 1;
      i = i + 1;
    }
 }
```

2.4 t taulako lehenengo z osagaiak inkrementatzen dituen funtzioa

Funtzio honek t taulako 0 posiziotik z – 1 posiziora arteko osagaien balioa inkrementatuko du.

- Funtzioaren prototipoa:

```
void kopiatu_inkrementatuz (int z, int t[]);
```

Kasu honetan z datua da eta t aldi berean datua eta emaitza den parametroa da. Izan ere funtzioari t taula ematen zaio datu bezala eta funtzioak taulako lehenengo z osagaiak inkrementatu ondoren t taula itzultzen du emaitza bezala. Beraz t-ren balioa aldatu egingo da. Horregatik funtzio hau 7. motakoa da.

- Funtzioaren definizioa:

```
void inkrementatu_taula (int z, int t[])

 /* Funtzio honek t taulako lehenengo z osagaien balioa inkrementatuko du. */
 { /*
   int i;

   i = 0; /* i aldagaia t taularen indize bezala erabiliko da. */

   while (i < z)
    {
     t[i] = t[i] + 1;
     i = i + 1;
    }
 }
```

- Funtzioaren erabilera:

  Demagun 15 osagai oso dituen b taula definitu dugula eta *bete_taula* funtzioa erabiliz dagoeneko b taulako lehenengo 6 posizioetan balio batzuk gorde ditugula:

```
int b[15];
bete_taula (6, b);
```

  Beraz esate baterako honako egoera hau edukiko dugu:

b

10	-8	29	7	20	34			...	
0	1	2	3	4	5	6	7	...	14

  b taulako lehenengo 4 osagaien balioa inkrementatzea nahi izango bagenu, honako hau idatzi beharko genuke:

```
inkrementatu_taula (4, b);
```

  Taula honela geldituko litzateke:

b

11	**-7**	**30**	**8**	20	34			...	
0	1	2	3	4	5	6	7	...	14

  Adibide honetan ere ikusi den bezala, funtzio bat erabiltzerakoan funtzioen argumentuak diren tauletan osagai-kopurua ez da zehaztu behar eta [ ] ere ez da ipini behar.

3 EBATZITAKO ARIKETAK

3.1 Taula bateko elementuak lehenengotik azkenera eta azkenetik lehenengora aurkezten dituen programa

Ariketa honetan idatzi beharreko programak erabiltzaileari 10 zenbaki eskatu behar dizkio eta zenbaki horiek 10 osagaiko taula batean gorde behar ditu. Zenbakiak taulan gorde ondoren, taulako osagaiak dauden ordenean eta alderantzizko ordenean aurkeztu behar ditu.

```
#include <stdio.h>


void main()
{ /* main funtzioaren hasierako giltza */
  int tau[10];
  int i, zenb;

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

  /* Hasteko taula bete behar da */

  i = 0; /* tau taularen indizea. */

  while (i < 10)
   {
     printf("\nTekleatu %d posiziorako zenbaki oso bat: ", i);
     scanf("%d", &zenb);
     tau[i] = zenb;
```

```
    i = i + 1;
   }

  /* Jarraian taula dagoen bezala aurkeztuko da */

  i = 0; /* tau taularen indizea. */

  printf("\n");

  while (i < 10)
   {
     printf("%d  ", tau[i]);
     i = i + 1;
   }


  i = 9; /* tau taularen indizea. */

  printf("\n");

  while (i >= 0)
   {
     printf("%d  ", tau[i]);
     i = i - 1;
   }

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

3.2 Zenbaki bat taula batean zein posiziotan agertzen den eta guztira zenbat aldiz agertzen den erabakitzen duen programa

Ariketa honetan idatzi beharreko programak hasteko erabiltzaileari 10 zenbaki eskatu behar dizkio eta zenbaki horiek 10 osagaiko taula batean gorde behar ditu. Zenbakiak taulan gorde ondoren, beste zenbaki bat eskatu eta zenbaki hori taulan zein posiziotan agertzen den eta guztira zenbat aldiz agertzen den esan behar du.

Erabili beharreko funtzioak:

- *kalkulatu_agerpen_kopurua*: argumentu bezala z eta x zenbaki osoak eta t taula emanda, t taulako lehenengo z osagaien artean x zenbakia zenbat aldiz agertzen den kalkulatu eta itzultzen duen funtzioa.

```
#include <stdio.h>


int kalkulatu_agerpen_kopurua (int z, int x, int t[]);


/* Programa nagusia */

void main()
{ /* main funtzioaren hasierako giltza */
  int tau[10];
  int zenb, i, agerkop;

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

  i = 0; /* tau taularen indizea. tau taulan libre dagoen lehenengo posizioa
            zein den adierazten du. Gainera une bakoitzean une horretara arte
            zenbat zenbaki eskatu diren jakiteko ere erabiliko da. */
```

```
  printf("\nSartu taulako 10 osagaiak zuriunez bananduta eta azkenean "
         "return sakatu: ");

  while (i < 10)
   {
     scanf("%d", &zenb);
     tau[i] = zenb;
     i = i + 1;
   }

  /* Jarraian zenbaki bat eskatuko da */

  printf ("\nZenbaki oso bat tekleatu: ");
  scanf ("%d", &zenb);

  /* Jarraian agerpenak aztertuko dira */

  agerkop = kalkulatu_agerpen_kopurua (10, zenb, tau);
  printf ("\n%d zenbakia %d aldiz agertzen da.", zenb, agerkop);

  /* Bukatzeko posizioak aurkeztuko dira */

  if (agerkop != 0)
   {
     i = 0;

     printf ("\nPosizioak: ");

     while (i < 10)
      {
        if (tau[i] == zenb)
         {
           printf ("%d  ", i);
         }
        i = i + 1;
      }
   }

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

                          /* Funtzioaren definizioa */

int kalkulatu_agerpen_kopurua (int z, int x, int t[])

 /* Funtzio honek x zenbakia t taulako lehenengo z posizioetan zenbat aldiz
    agertzen den kalkulatu eta itzuliko du. */

 {
   int kont, i;

   i = 0; /* t taularen indizea. t taula posizioz posizio zeharkatzeko
             erabiliko da. */

   kont = 0; /* x-en agerpenak kontatzeko erabiliko da. */

   while (i < z)
    {
      if (t[i] == x)
       {
         kont = kont + 1;
       }
      i = i + 1;
    }

   return (kont);
 }
```

3.3 Zenbaki bat taula batean agertzen al den ala ez erabakitzen duen programa

Ariketa honetan idatzi beharreko programak hasteko erabiltzaileari 10 zenbaki eskatu behar dizkio eta zenbaki horiek 10 osagaiko taula batean gorde behar ditu. Zenbakiak taulan gorde ondoren, beste zenbaki bat eskatu eta zenbaki hori taulan agertzen al den ala ez erabaki behar du.

Erabili beharreko funtzioak:

- *agertzen_al_da*: argumentu bezala z eta x zenbaki osoak eta t taula emanda, t taulako lehenengo z osagaien artean x zenbakia agertzen bada, 1, eta ez bada agertzen, 0 itzultzen duen funtzioa. Hau 1. motako funtzioa da.

```
#include <stdio.h>


/* Programan erabiliko den funtzioaren aipamena edo prototipoa. */

int agertzen_al_da (int z, int x, int t[]);

/* Funtzio nagusia */

void main()
{ /* main funtzioaren hasierako giltza */

  int tau[10];
  int zenb, i;

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

  i = 0; /* tau taularen indizea. */

  printf("\nSartu taulako 10 osagaiak zuriunez bananduta eta azkenean "
         "return sakatu: ");

  while (i < 10)
   {
     scanf("%d", &zenb);
     tau[i] = zenb;
     i = i + 1;
   }

  /* Jarraian zenbaki bat eskatuko da */

  printf ("\nZenbaki oso bat tekleatu: ");
  scanf ("%d", &zenb);

  /* Jarraian agerpenak aztertuko dira */

  if (agertzen_al_da (10, zenb, tau) == 1)
   {
     printf ("\n%d zenbakia gutxienez behin agertzen da.", zenb);
   }
  else
   {
     printf ("\n%d zenbakia ez da agertzen.", zenb);
   }

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

```
                        /* Funtzioaren definizioa */

int agertzen_al_da (int z, int x, int t[])

 /* Funtzio honek x zenbakia t taulako lehenengo z posizioetan gutxienez behin
    agertzen bada, 1, eta behin ere ez bada agertzen, 0 itzuliko du. */

 {
   int aurkitu, i;

   i = 0; /* t taularen indizea. */

   aurkitu = 0; /* aurkitu aldagaiaren balioa 0 denean, oraindik x balioa t
                   taulan ez dugula aurkitu esan nahi du. Une batean aurkitu
                   aldagaiaren balioa 1 bada, x balioa t taulan aurkitu dugula
                   esan nahiko du. */

   /* Funtzio honetan x zenbakia lehenengo aldiz aurkitzen denean bilaketa
      bukatu egingo da, badagoela erabakitzeko behin aurkitzearekin nahikoa
      baita. */

   while ((i < z) && (aurkitu == 0))
    {
      if (t[i] == x)
       {
         aurkitu = 1;
       }
      i = i + 1;
    }

   return (aurkitu);
 }
```

3.4 Osagai-kopuru bera duten bi bektore konparatzen dituen programa

Ariketa honetan idatzi beharreko programak bi bektoreren osagaiak bi tauletan gorde eta jarraian bi bektoreak konparatu behar ditu.

Bektore biek osagai-kopuru bera izango dute baina osagai-kopurua zein den erabiltzaileari galdetu beharko zaio programaren hasieran. Hala ere, osagai-kopurua gehienez 15 izango da.

Erabili beharreko funtzioa:

- *konparatu_taulak*: argumentu bezala z zenbaki osoa eta t1 eta t2 taulak emanda, t1 taulako lehenengo z elementuak t2 taulako lehenengo z elementuen berdinak badira, funtzioak 1 balioa itzuliko du eta bestela 0 balioa itzuliko du.

```
#include <stdio.h>


/* Programan erabiliko den funtzioaren aipamena edo prototipoa. */

int konparatu_bektoreak (int z, int t1[], int t2[]);

/* Funtzio nagusia */

void main()
{ /* main funtzioaren hasierako giltza */
  int bek1[15];
  int bek2[15];
```

```
  int osakop, konp, I, zenb;

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

  /* Hasteko bektoreek zenbat osagai izango dituzten galdetuko zaio
     erabiltzaileari. Bektoreek gutxienez osagai bat eta gehienez 15 izan
     ditzakete */

  printf ("\nAdierazi osagai-kopurua(gutxienez 1 eta gehienez 15): ");
  scanf ("%d", &osakop);

  while ((osakop < 1) || (osakop > 15))
   {
     printf ("\nTekleatutako zenbakia ez da egokia.");
     printf ("\nAdierazi osagai-kopurua (gutxienez 1 eta gehienez 15): ");
     scanf ("%d", &osakop);
   }


  /* Jarraian lehenengo bektorearen osagaiak eskatuko dira */

  printf("\nSartu 1. bektoreko %d osagaiak zuriunez bananduta eta azkenean "
         "return sakatu: ", osakop);
  i = 0;
  while (i < osakop)
   {
     scanf("%d", &zenb);
     bek1[i] = zenb;
     i = i + 1;
   }

  /* Orain bigarren bektorearen osagaiak eskatuko dira */

  printf("\nSartu 2. bektoreko %d osagaiak zuriunez bananduta eta azkenean "
         "return sakatu: ", osakop);
  i = 0;
  while (i < osakop)
   {
     scanf("%d", &zenb);
     bek2[i] = zenb;
     i = i + 1;
   }

  /* Konparaketa egiteko konparatu_bektoreak funtzioa erabiliko da */

  konp = konparatu_bektoreak (osakop, bek1, bek2);

  /* Bukatzeko konparaketaren emaitza aurkeztuko da */

  if (konp == 1)
   {
     printf ("\nBektoreak berdinak dira.");
   }
  else if (konp == 0)
   {
     printf ("\nBektoreak ez dira berdinak.");
   }

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


                        /* Funtzioaren definizioa */

int konparatu_bektoreak (int z, int t1[], int t2[])

  /* Funtzio honek t1 eta t2 tauletako lehenengo z osagaiak berdinak badira,
```

```
     1 itzuliko du emaitza bezala eta bestela 0 itzuliko du. */

 {
   int berdinak, i;

   i = 0; /* t1 eta t2 taulen indizea. Taula horiek posizioz posizio
             zeharkatzeko erabiliko da i aldagaia. */

   berdinak = 1; /* berdinak izeneko aldagaiaren balioa 1 den bitartean, orain
                    arte konparatu diren posizioak berdinak direla esan nahiko
                    du. Une batean berdinak izeneko aldagaiaren balioa 0
                    izatera pasatzen bada, t1 eta t2 tauletan desberdinak
                    diren elementuak aurkitu direla esan nahiko du. */

   /* berdinak ez diren bi elementu aurkituz gero konparaketarekin ez da
      jarraituko. Kasu horretan bai baitakigu taulak desberdinak direla. */

   while ((i < z) && (berdinak == 1))
    {
      if (t1[i] != t2[i])
       {
         berdinak = 0;
       }
      i = i + 1;
    }

   return (berdinak);
 }

                              /********************/
```

3.5 Bektore bateko osagaiak goranzko ordenean al dauden ala ez erabakitzen duen programa

Ariketa honetan idatzi beharreko programak bektore baten osagaiak taula batean gorde eta jarraian bektoreko osagaiak goranzko ordenean al dauden ala ez esanez mezu bat aurkeztu behar du.

Bektoreak gehienez 20 osagai izan ditzake, baina osagai-kopuru hori desberdina izan daiteke programaren exekuzio bakoitzean. Horregatik osagai-kopurua zein den galdetu beharko zaio erabiltzaileari programaren hasieran.

Erabili beharreko funtzioa:

- *erabaki_ordena*: argumentu bezala z zenbaki osoa eta t taula emanda, t taulako lehenengo z elementuak goranzko ordenean baldin badaude, 1, beheranzko ordenean badaude, 0 eta ez goranzko eta ez beheranzko ordenean ez badaude, -1 itzultzen duen funtzioa.

```
#include <stdio.h>


/* Programan erabiliko den funtzioaren aipamena edo prototipoa. */

int erabaki_ordena (int z, int t[]);

/* Funtzio nagusia */

void main()
{ /* main funtzioaren hasierako giltza */
  int bek[20]; /* bektoreak gehienez 20 osagai izan ditzakeenez, 20 osagaiko
                  taula bat definituz badakigu erabiltzaileak tekleatutako
                  edozein bektore taulan ondo sartuko dela. */
```

```
  int osakop, ordena, i, zenb;

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

  /* Hasteko bektoreak zenbat osagai izango dituen galdetuko zaio
     erabiltzaileari. Bektoreak gutxienez osagai bat eta gehienez 20 izan
     ditzake. */

  printf ("\nAdierazi osagai-kopurua(gutxienez 1 eta gehienez 20): ");
  scanf ("%d", &osakop);

  while ((osakop < 1) || (osakop > 20))
   {
     printf ("\nTekleatutako zenbakia ez da egokia.");
     printf ("\nAdierazi osagai-kopurua (gutxienez 1 eta gehienez 20): ");
     scanf ("%d", &osakop);
   }

  /* Jarraian bektorearen osagaiak eskatuko dira */

  printf("\nSartu bektoreko %d osagaiak zuriunez bananduta eta azkenean "
         "return sakatu: ", osakop);
  i = 0;
  while (i < osakop)
   {
     scanf("%d", &zenb);
     bek[i] = zenb;
     i = i + 1;
   }

  /* Taula ordenatuta al dagoen erabakitzeko erabaki_ordena izeneko funtzioa
     erabiliko da */

  ordena = erabaki_ordena (osakop, bek);

  /* Bukatzeko erabakiari dagokion mezua aurkeztuko da */

  if (ordena == 1)
   {
     printf ("\nBektoreko osagaiak goranzko ordenean daude.");
   }
  else
   {
     printf ("\nBektorearen osagaiak ez daude goranzko ordenean.");
   }

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

                          /* Funtzioaren definizioa */

int erabaki_ordena (int z, int t[])

 /* Funtzio honek t taulako lehenengo z osagaiak goranzko ordenean badaude, 1,
    beheranzko ordenean badaude, 0, eta ez goranzko eta ez beheranzko ordenean
    ez badaude, -1 itzuliko du. */

 {
   int goranzko_or, beheranzko_or, i;

   i = 0; /* t taularen indizea. Taula hori posizioz posizio zeharkatzeko
             erabiliko da i aldagaia. */

 /* Hasteko t taulako elementuak goranzko ordenean al dauden begiratuko da.
    Goranzko ordenean badaude, 1 balioa itzuli eta hor bukatuko da funtzioa.
    Baina goranzko ordenean ez badaude, beheranzko ordenean al dauden erabaki
```

```
   beharko da. Beheranzko ordenean badaude, 0 itzuli eta hor bukatuko da
   funtzioa, eta beheranzko ordenean ere ez badaude, orduan -1 itzuli behar
   du funtzioak. */

    goranzko_or = 1; /* goranzko_or izeneko aldagaiaren balioa 1 den
                        bitartean, orain arte aztertu diren posizioak
                        goranzko ordenean daudela esan nahiko du. Une batean
                        goranzko_or izeneko aldagaiaren balioa 0 izatera
                        pasatzen bada, t taulan goranzko ordenean ez dauden
                        bi elementu aurkitu direla esan nahiko du. */

/* Taulako osagaiak zeharkatuz eta binaka konparatuz (0 eta 1 posizioak, 1
   eta 2 posizioak, 2 eta 3 posizioak, eta abar.) joan beharko dugu.
   Konparatutako bikoteak goranzko ordenean dauden bitartean (hau da,
   lehenengoa bigarrena baino txikiagoa edo berdina den bitartean) aurrera
   jarraitu behar da, baina goranzko ordenean ez dauden bi elementu aurkituz
   gero, konparaketarekin ez da jarraituko, kasu horretan bai baitakigu
   taulako elementuak ez daudela goranzko ordenean. */                    →

/* Konparatu beharreko elementuak 0 eta 1 posizioetakoak, 1 eta 2
   posizioetakoak, 2 eta 3 posizioetakoak eta abar dira. Konparatu beharreko
   azkeneko elementuak z - 2 eta z - 1 posizioetakoak dira. Hori dela eta i
   aldagaiak 0tik z - 2 posiziora arte joan behar du. */


  while ((i < (z - 1)) && (goranzko_or == 1))
   {
     if (t[i] > t[I + 1])
      {
        goranzko_or = 0;
      }
     i = i + 1;
   }

  if (goranzko_or == 1)
   {
     return (1);
   }
  else /* Goranzko ordenean ez badago, beheranzko ordenean al dagoen begiratu
          behar da. */
   {
     i = 0;
     beheranzko_or = 1; /* beheranzko_or izeneko aldagaiaren balioa 1 den
                           bitartean, orain arte aztertu diren posizioak
                           beheranzko ordenean daudela esan nahiko du. Une
                           batean beheranzko_or izeneko aldagaiaren balioa 0
                           izatera pasatzen bada, t taulan beheranzko
                           ordenean ez dauden bi elementu aurkitu direla esan
                           nahiko du. */

     while ((i < (z - 1)) && (beheranzko_or == 1))
      {
        if (t[i] < t[i + 1])
         {
           beheranzko_or = 0;
         }
        i = i + 1;
      }

     if (beheranzko_or == 1)
      {
        return (0);
      }
     else /* Beheranzko ordenean ere ez badago, -1 itzuli behar du funtzioak.
             */
      {
        return (-1);
      }
```

```
    }
 }
```

3.6 Taulako zenbaki lehenak aurkitu eta aurkezten dituen programa

Ariketa honetan idatzi beharreko programak hasteko erabiltzaileari 20 zenbaki eskatu behar dizkio eta zenbaki horiek 20 osagaiko taula batean gorde behar ditu. Zenbakiak taulan gorde ondoren, taula posizioz posizio zeharkatu eta lehenak diren elementuak aurkeztu behar ditu.

Luzera handiena segida batean baino gehiagotan ematen bada, lehenengoari buruzko informazioa bakarrik aurkeztu behar du programak.

Erabili beharreko funtzioa:

- *lehena_al_da*: argumentu bezala ≥ 1 den x zenbaki bat emanda, zenbaki hori lehena al den ala ez erabakitzen duen funtzioa. Emandako x zenbakia lehena bada, 1 balioa itzuliko du erantzun bezala, eta bestela 0 itzuliko du.

```
#include <stdio.h>


/* Programan erabiliko den funtzioaren aipamena edo prototipoa. */

int lehena_al_da (int x);

/* Funtzio nagusia */

void main()
{ /* main funtzioaren hasierako giltza */
  int tau[20];
  int i, zenb;

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

  /* Hasteko taula zenbaki positiboz beteko da */

  i = 0; /* tau taularen indizea. tau taulan libre dagoen lehenengo posizioa
            zein den adierazten du. Gainera une bakoitzean une horretara arte
            zenbat zenbaki eskatu diren jakiteko ere erabiliko da. */

  while (i < 20)
   {
     printf("\nTekleatu %d posiziorako >= 1 den zenbaki oso bat: ", i);
     scanf("%d", &zenb);
     while (zenb < 1)
      {
        printf("\nSartutako zenbakia ez da positiboa.");
        printf("\nTekleatu %d posiziorako >= 1 den zenbaki oso bat: ", i);
        scanf("%d", &zenb);
      }

     tau[i] = zenb;
     i = i + 1;
   }

                                                                        →
  /* Orain taula zeharkatuko da lehenak direnak aurkitu eta aurkezteko */

  i = 0; /* tau taularen indizea. Aldagai hau tau taula posizioz posizio
            zeharkatzeko erabiliko da. */
```

```
  while (i < 20)
   {
     if (lehena_al_da(tau[i]) == 1)
      {
        printf("\n%d posizioko elementua lehena da: %d", i, tau[i]);
      }

     i = i + 1;
   }

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


                          /* Funtzioaren definizioa */

int lehena_al_da (int x)

 /* Funtzio honek argumentu bezala >= 1 den x zenbaki osoa emanda, x lehena
    bada, 1, eta bestela 0 itzuliko du. */

 {
   int zenb, kont;

   kont = 0; /* x-en zatitzaileak kontatzeko erabiliko den aldagaia */
   zenb = 1; /* zenbakiak 1etik x-era banan-banan pasatzeko erabiliko den
                aldagaia */

   while (zenb <= x)
    {
      if (x % zenb == 0)
       {
         kont = kont + 1;
       }
      zenb = zenb + 1;
    }

   if (kont == 2)
    {
      return (1);
    }
   else
    {
      return (0);
    }
 }
```

3.7 Bektore bateko osagaiak goranzko ordenean ipintzen dituen programa

Ariketa honetan idatzi beharreko programak bektore baten osagaiak ordenatu behar ditu goranzko ordenean geldi daitezen.

Bektoreak gehienez 20 osagai izan ditzake, baina osagai-kopuru hori desberdina izan daiteke programaren exekuzio bakoitzean. Horregatik osagai-kopurua zein den galdetu beharko zaio erabiltzaileari programaren hasieran.

Erabili beharreko funtzioa:

- *ordenatu_taula*: argumentu bezala p1 eta p2 zenbaki osoak eta t taula emanda, t taulako p1 posiziotik p2 posiziora arteko zatia goranzko ordenean ordenatzen duen funtzioa. Funtzio honek *txikienaren_posizioa* funtzioa erabiliko du.

- *txikienaren_posizioa*: argumentu bezala p1 eta p2 zenbaki osoak eta t taula emanda, t taulako p1 posiziotik p2 posiziora arteko zatian zenbaki txikiena denaren posizioa itzultzen duen funtzioa. Zenbaki txikiena errepikatuta baldin badago, lehenengo agerpenaren posizioa itzuliko du.

```
#include <stdio.h>


/* Programan erabiliko diren funtzioen aipamena edo prototipoa. */

int txikienaren_posizioa (int p1, int p2, int t[]);
void ordenatu_taula (int p1, int p2, int t[]);

                              /* Funtzio nagusia */

void main()
{ /* main funtzioaren hasierako giltza */
  int tau[20]; /* bektoreak gehienez 20 osagai izan ditzakeenez, 20 osagaiko
                  taula bat definituz badakigu erabiltzaileak tekleatutako
                  edozein bektore taulan ondo sartuko dela. */

  int osakop, zenb, i;

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


  /* Hasteko bektoreak zenbat osagai izango dituen galdetuko zaio
     erabiltzaileari. Bektoreak gutxienez osagai bat eta gehienez 20 izan
     ditzake. */

  printf ("\nAdierazi osagai-kopurua(gutxienez 1 eta gehienez 20): ");
  scanf ("%d", &osakop);

  while ((osakop < 1) || (osakop > 20))
   {
     printf ("\nTekleatutako zenbakia ez da egokia.");
     printf ("\nAdierazi osagai-kopurua (gutxienez 1 eta gehienez 20): ");
     scanf ("%d", &osakop);
   }

  /* Jarraian bektorearen osagaiak eskatuko dira */

  printf("\nSartu bektoreko %d osagaiak zuriunez bananduta eta azkenean "
         "return sakatu: ", osakop);
  i = 0;
  while (i < osakop)
   {
     scanf("%d", &zenb);
     tau[i] = zenb;
     i = i + 1;
   }

  /* Orain taula goranzko ordenean ordenatuta ipiniko da */

  ordenatu_taula (0, osakop - 1, tau);

  /* Bukatzeko taula ordenatua aurkeztuko da */

  i = 0;

  while (i < osakop)
   {
```

```
     printf("%d  ", tau[i]);
     i = i + 1;
   }

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

                          /* Funtzioen definizioa */

void ordenatu_taula (int p1, int p2, int t[])

 /* Funtzio honek t taulako p1 eta p2 posizioen arteko zatia goranzko ordenean
    ordenatuko du. */

 {
   int txikiposi, i, lag;

   i = p1; /* t taularen indizea. Taula hori posizioz posizio zeharkatzeko
              erabiliko da i aldagaia. */

   /* Taula ordenatzeko i posiziotik p2 posiziora arteko elementu txikiena
      aurkitu eta i posiziora ekarriko da. Horrela, i aldagaiaren balioa p1
      denean, p1 eta p2 posizioen arteko elementu txikiena aurkitu eta p1
      posiziokoarekin trukatuko da. Era berean, i-ren balioa p1 + 1 denean p1
      + 1 eta p2 posizioen arteko elementu txikiena aurkitu eta p1 + 1
      posiziokoarekin trukatuko da eta abar. Prozesu horren bidez p1 eta p2
      posizioen arteko zatia ordenatzea lortuko da. */

                                                                          →
   while (i < p2)
    {
      txikiposi = txikienaren_posizioa (i, p2, t);
      lag = t[i];
      t[i] = t[txikiposi];
      t[txikiposi] = lag;
      i = i + 1;
    }
 }

                          /*******************/

int txikienaren_posizioa (int p1, int p2, int t[])

 /* Funtzio honek t taulako p1 eta p2 posizioen artean dagoen balio
    txikienaren posizioa itzuliko du. */

 {
   int txikiena, posizioa, i;

   i = p1; /* t taularen indizea. p1 posiziotik p2 posiziora arte t taula
              posizioz posizio zeharkatzeko erabiliko da. */

   txikiena = t[p1]; /* Une bakoitzean une horretara arte zeharkatu diren
                        posizioetako balio txikiena gordetzeko erabiliko da.
                        Hasieran txikiena lehenengoa da, besterik ez baita
                        oraindik aztertu. */

   posizioa = p1; /* Une bakoitzean une horretara arte zeharkatu diren
                     posizioetako balio txikienari dagokion posizioa
                     gordetzeko erabiliko da. Hasieran txikiena p1 posiziokoa
                     denez, p1 balioa gordeko da aldagai honetan. */

   while (i <= p2)
    {
      if (t[i] < txikiena)
```

```
       {
         txikiena = t[i];
         posizioa = i;
       }
      i = i + 1;
    }

   return (posizioa);
 }
```

3.8 Maila desberdinekoak izan daitezkeen bi polinomio batzen dituen programa

Ariketa honetan idatzi beharreko programak maila desberdinekoak izan daitezkeen bi polinomio batuz lortzen den polinomioa kalkulatu eta aurkeztu behar du.

Hasierako bi polinomioak eta batuketaren ondorioz lortzen den polinomioa hiru taulatan gorde behar dira.

Hasieran programak, polinomio bakoitzaren koefizienteak eskatu aurretik, polinomio bakoitzaren maila zein den galdetu behar dio erabiltzaileari.

Polinomioen maila gehienez 20 izan daiteke, beraz polinomioek gehienez 21 koefiziente izan ditzakete eta ondorioz, polinomioak gordetzeko 21 osagaiko taulak definituko dira.

Erabili beharreko funtzioak:

- *batu_polinomioak*: argumentu bezala bi polinomioren mailak (m1 eta m2) eta bi polinomio horien koefizienteak dituzten pol1 eta pol2 taulak emanda, bi polinomio horiek batuz lortzen den polinomio berriaren koefizienteak (pol3) itzultzen dituen funtzioa. Funtzio honek *batu_taulak* eta *kopiatu_taula* funtzioak erabiliko ditu.

- *batu_taulak*: argumentu bezala p1 eta p2 zenbaki osoak eta t1, t2 eta t3 taulak emanda, t1 eta t2 taulak hartu eta p1 posiziotik p2 posiziora arteko elementuak posizioz posizio batu (t1 taulako elementu bakoitza t2-ko bere parekoarekin) eta t3 taulako pareko posizioan lagatzen duen funtzioa.

- *kopiatu_taula*: argumentu bezala p1 eta p2 zenbaki osoak eta t1 eta t2 taulak emanda, t1 taula hartu eta p1 posiziotik p2 posiziora arteko elementuak posizioz posizio t2 taulako pareko posizioetan kopiatzen dituen funtzioa.

```
#include <stdio.h>


/* Programan erabiliko diren funtzioen aipamena edo prototipoak. */

void batu_polinomioak (int m1, float pol1[], int m2, float pol2[],
                       float pol3[]);
void batu_taulak (int p1, int p2, float t1[], float t2[], float t3[]);
void kopiatu_taula (int p1, int p2, float t1[], float t2[]);

/* Funtzio nagusia */

void main()
{ /* main funtzioaren hasierako giltza */
  float poli1[21]; /* Polinomioen maila gehienez 20 izan daitekeenez
                      (beraz gehienez 21 koefiziente), 21 osagaiko
                      taulak definituz badakigu erabiltzaileak
```

```
                    tekleatutako edozein polinomio tauletan ondo
                    sartuko dela. */

float poli2[21];
float poli3[21];
int maila1, maila2, maila3, i;
float koef;

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

/* Hasteko lehenengo polinomioaren maila eskatuko da */

printf ("\nAdierazi lehenengo polinomioaren maila zein "
        "den ([0, %d] tartekoa izan behar du): ", 20);
scanf ("%d", &maila1);

while ((maila1 < 0) || (maila1 > 20))
 {
   printf ("\nTekleatutako zenbakia ez da egokia.");
   printf ("\nAdierazi lehenengo polinomioaren maila zein "
           "den ([0, %d] tartekoa izan behar du): ", 20);
   scanf ("%d", &maila1);
 }

/* Jarraian lehenengo polinomioaren koefizienteak eskatuko dira */

printf("\nOrain 1. polinomioaren koefizienteak tekleatu beharko dituzu.");

 i = 0; /* taulan libre dagoen lehenengo posizioa zein den adierazten du.
           Gainera une bakoitzean une horretara arte zenbat zenbaki eskatu
           diren jakiteko ere erabiliko da. */

 while (i <= maila1)
  {
    printf("\nTekleatu %d mailako koefizientea: ", i);
    scanf("%d", &koef);
    poli1[i] = koef;
    i = i + 1;
  }

/* Orain bigarren polinomioaren maila eskatuko da */

printf ("\nAdierazi bigarren polinomioaren maila zein "
        "den ([0, %d] tartekoa izan behar du): ", 20);
scanf ("%d", &maila2);

while ((maila2 < 0) || (maila2 > z))
 {
   printf ("\nTekleatutako zenbakia ez da egokia.");
   printf ("\nAdierazi bigarren polinomioaren maila zein "
           "den ([0, %d] tartekoa izan behar du): ", 20);
   scanf ("%d", &maila2);
 }

/* Eta jarraian bigarren polinomioaren koefizienteak eskatuko dira */

printf("\nOrain 2. polinomioaren koefizienteak tekleatu beharko dituzu.");

 i = 0; /* taulan libre dagoen lehenengo posizioa zein den adierazten du.
           Gainera une bakoitzean une horretara arte zenbat zenbaki eskatu
           diren jakiteko ere erabiliko da. */

 while (i <= maila2)
  {
    printf("\nTekleatu %d mailako koefizientea: ", i);
    scanf("%d", &koef);
    poli2[i] = koef;
    i = i + 1;
```

```
    }

  /* Polinomioek maila desberdina izan dezaketenez, hasteko polinomio
     berriaren maila kalkulatuko da. */

  if (maila1 >= maila2)
   {
     maila3 = maila1;
   }
  else
   {
     maila3 = maila2;
   }

  /* Jarraian polinomioak batuko dira */

  batu_polinomioak (maila1, polinomioa1, maila2, polinomioa2,
                    polinomioa3);

  /* Bukatzeko hiru polinomioak aurkeztuko dira */

  system("cls");

  printf("\n1. polinomioa: \n");

   i = 0;

   while (i <= maila1)
    {
      printf("%d  ", poli1[i]);
      i = i + 1;
    }

  printf("\n2. polinomioa: \n");

   i = 0;

   while (i <= maila2)
    {
      printf("%d  ", poli2[i]);
      i = i + 1;
    }

  printf("\n3. polinomioa: \n");

   i = 0;

   while (i <= maila3)
    {
      printf("%d  ", poli3[i]);
      i = i + 1;
    }

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

                          /* Funtzioen definizioa */

void batu_polinomioak (int m1, float pol1[], int m2, float pol2[],
                       float pol3[])

 /* Funtzio honek m1 eta m2 mailakoak diren pol1 eta pol2 polinomioak batuz
    lortzen den polinomioaren koefizienteak (pol3) itzuliko ditu. */

 {
   if (m1 >= m2)
    {
```

```
      batu_taulak (0, m2, pol1, pol2, pol3);
      kopiatu_taulak (m2 + 1, m1, pol1, pol3);
    }
   else
    {
      batu_taulak (0, m1, pol1, pol2, pol3);
      kopiatu_taulak (m1 + 1, m2, pol2, pol3);
    }
 }
                              /********************/

void batu_taulak (int p1, int p2, float t1[], float t2[], float t3[])

 /* Funtzio honek t1 eta t2 tauletako p1 posiziotik p2 posiziora arteko osagai
    bakoitza bere parekoarekin batu eta t3-ko pareko posizioan gordeko du. */

 {
   int i;

   i = p1; /* t1, t2 eta t3 taulen indizea. Aldagai hau t1, t2 eta t3 taulak
              posizioz posizio zeharkatzeko erabiliko da. */

   while (i <= p2)
    {
      t3[i] = t1[i] + t2[i];
      i = i + 1;
    }
 }
                              /********************/

void kopiatu_taula (int p1, int p2, float t1[], float t2[])

 /* Funtzio honek t1 taulako p1 posiziotik p2 posiziora arteko osagaiak t2
    taulako pareko posizioan kopiatuko ditu. */

 {
   int i;

   i = p1; /* t1 eta t2 taulen indizea. Aldagai hau t1 eta t2 taulak posizioz
              posizio zeharkatzeko erabiliko da. */

   while (i <= p2)
    {
      t2[i] = t1[i];
      i = i + 1;
    }
 }
                              /********************/
```

3.9 Ikasle-talde bateko notei dagozkien kalifikazioak aurkezten dituen programa (azkeneko nota negatiboa ez da gordeko)

Ariketa honetan idatzi beharreko programak talde bateko ikasleen notak eskatu, taula batean gorde eta jarraian nota bakoitzari dagokion kalifikazioa aurkeztu behar du. Azkeneko nota bezala zenbaki negatibo bat emango da. Beraz programak zenbaki negatibo bat eskuratzen duenean, notak eskatzeko prozesua bukatu egin behar du.

Gehienez 100 nota positibo emango zaizkio eta **azkeneko zenbaki negatiboa ez du gorde behar**. Ondorioz notak gordetzeko 100 osagaiko taula bat edukitzea nahikoa izango da.

```
#include <stdio.h>


void main()
{ /* main funtzioaren hasierako giltza */
  float notataula[100];

  int notakop, i;
  float notabat;

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

  /* Hasteko notak eskatuko dira */

  i = 0; /* Aldagai hau indize bezala (une bakoitzean taulan
            libre dagoen lehenengo posizioa zein den adierazteko) eta gainera
            une bakoitzean une horretara arte zenbat nota eskuratu diren
            jakiteko ere erabiliko da. */

  printf("\nJarraian notak banan-banan eskatuko zaizkizu. Azkenak "
         "negatiboa izan behar du eta gehienez %d izan daitezke.", 100);
  printf("\nTekleatu %d. nota: ", i);
  scanf("%d", &notabat);

  while (notabat >= 0)
   {
     notataula[i] = notabat;
     i = i + 1;
     printf("\nTekleatu %d. nota: ", i);
     scanf("%d", &notabat);
   }

  notakop = i;

                                                                    →
  /* Orain kalifikazioak aurkeztuko dira */

  if (notakop == 0)
   {
      printf ("\nEz da notarik sartu.");
   }
  else
   {
     i = 0;
     while (i < notakop)
      {
           if ((notataula[i] >= 0) && (notataula[i] < 5))
            {
              printf ("\nGutxiegi.");
            }
           else if ((notataula[i] >= 5) && (notataula[i] < 7))
            {
              printf ("\nNahikoa.");
            }
           else if ((notataula[i] >= 7) && (notataula[i] < 9))
            {
              printf ("\nOso ongi.");
            }
           else if ((notataula[i] >= 9) && (notataula[i] < 10))
            {
              printf ("\nBikain.");
            }
           else if (notataula[i] == 10)
            {
              printf ("\nOhorezko matrikula.");
            }

           i = i + 1;
```

```
      }
   }

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

3.10 Ikasle-talde bateko notei dagozkien kalifikazioak aurkezten dituen programa (azkeneko nota negatiboa ere taulan gordez)

Ariketa honetan idatzi beharreko programak talde bateko ikasleen notak eskatu, taula batean gorde eta jarraian nota bakoitzari dagokion kalifikazioa aurkeztu behar du.

Azkeneko nota bezala zenbaki negatibo bat emango da. Beraz programak zenbaki negatibo bat eskuratzen duenean, notak eskatzeko prozesua bukatu egin behar du.

Gehienez 100 nota positibo emango zaizkio eta **azkeneko zenbaki negatiboa ere gorde egin behar du**. Ondorioz notak gordetzeko 101 osagaiko taula bat edukitzea nahikoa izango da.

```
#include <stdio.h>

void main()
{ /* main funtzioaren hasierako giltza */
  float notataula[101];
  int i;
  float notabat;

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

  /* Hasteko notak eskatuko dira.
     Gehienez 100 nota izan daitezke eta gainera noten bukaera adierazteko
     erabiltzaileak azkeneko nota bezala zenbaki negatibo bat tekleatu behar
     du. Zenbaki negatibo hori ere taulan gordeko da. Erabiltzaileak 100 nota
     baino gehiago ez dituela tekleatuko, azkeneko zenbakia negatiboa izango
     dela eta notak egokiak izango direla suposatuko da. Ez da ezer
     egiaztatuko.*/

  i = 0; /* taulan libre dagoen lehenengo posizioa zein den adierazteko
            erabiliko da. */
  printf("\nJarraian notak banan-banan eskatuko zaizkizu. Azkenak "
         "negatiboa izan behar du eta gehienez (negatiboa kontuan "
         "hartzeke) %d nota sar ditzakezu.", 100);
  printf("\nTekleatu %d. nota: ", i);
  scanf("%d", &notabat);

  while (notabat >= 0)
   {
     notataula[i] = notabat;
     i = i + 1;
     printf("\nTekleatu %d. nota: ", i);
     scanf("%d", &notabat);
   }

  notataula[i] = notabat;                                          →

  /* Jarraian kalifikazioak aurkeztuko dira. Kasu honetan azkeneko nota
     negatiboa ere taulan gorde denez, nota negatibo hori agertu arte
     jarraituko da aurrera. */

  i = 0;

  if (notataula[i] < 0)
   {
     printf ("\nEz da notarik sartu.");
   }
```

```
  else
   {
     while (notataula[i] >= 0)
      {
        if ((notataula[i] >= 0) && (notataula[i] < 5))
         {
           printf ("\nGutxiegi.");
         }
        else if ((notataula[i] >= 5) && (notataula[i] < 7))
         {
           printf ("\nNahikoa.");
         }
        else if ((notataula[i] >= 7) && (notataula[i] < 9))
         {
           printf ("\nOso ongi.");
         }
        else if ((notataula[i] >= 9) && (notataula[i] < 10))
         {
           printf ("\nBikain.");
         }
        else if (notataula[i] == 10)
         {
           printf ("\nOhorezko matrikula.");
         }

        i = i + 1;
      }
   }

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