6. LABORATEGIA
FUNTZIOAK

1 HELBURUAK

Laborategi hau amaitutakoan ondoren aipatzen direnak egiteko gai izango zara:

- Funtzio matematikoak edozein funtzioren barnean erabili (ez main-en bakarrik).
- Problemetan azpiproblemak identifikatu eta mugatu eta azpiproblema horiek ebazten dituzten funtzioak definitu daitezkeen bezala, **azpiproblemetan ere era berean azpiproblema txikiagoak identifikatu** eta isolatu daitezkeela eta azpiproblema txikiago horiek ebazten dituzten funtzioak defini daitezkeela ulertu.
- Beste funtzioei deitzen dieten edo beste funtzioak erabiltzen dituzten funtzioak definitu eta erabili.

2 MOTIBAZIOA

- Problema bat azpiproblematan zatitzen denean, azpiproblema bakoitza azpiproblema txikiagotan zatitzeko beharra edo nahia sortzen da, horrela maila desberdinak sortuz. Baina kalkulu bera maila desberdinetan eta azpiproblema desberdinetan ager daiteke. Kalkulu hori burutzen duen funtzio independente bat edukiz gero, funtzio hori edozein lekutan erabiltzea izango litzateke egokiena, maila kontuan hartzeke. Horregatik, funtzioak main funtzioan bakarrik erabili beharrean edozein funtzioren barruan erabiltzea komeni da.
- Edozein funtziotatik beste funtzioei deitzeko aukeraren bidez funtzioen erabilpena orokortuz, programak oraindik gehiago egituratzea eta ulertzen errazagoak egitea lortzen da.

2.1 Funtzioetatik beste funtzioei deituz

- Funtzio batean beste funtzioak erabiltzeko orduan ez dago inolako mugarik.
- Funtzio batzuetatik beste funtzio batzuei deitu arren, funtzio denak era independentean definitzen dira: main funtzioaren aurrean funtzio denen **prototipoak** ipiniko dira eta funtzioen **definizioak** main funtzioaren ondoren, denak jarraian, ipiniko dira. Funtzioak definitzerakoan ordenak ez du garrantzirik.

3 ADIBIDE ARIKETA

3.1 Helburua:

- Funtzio matematikoak edozein funtziotan (ez main-en bakarrik) nola erabili daitezkeen azaldu.
- Funtzio batzuetatik beste funtzio batzuei deitzen zaienean funtzioak nola definitu eta erabili behar diren azaldu.

3.2 Enuntziatua:

Erabiltzaileari ≥ 1 den n zenbaki oso bat eskatu eta alde batetik, lehenengo n faktorialak eta beraien erro karratuak kalkulatu eta aurkeztu eta beste aldetik, lehenengo n zenbaki lehenak kalkulatzen dituen programa. Hasieran, n eskatzerakoan, eskatze-prozesua ≥ 1 den zenbaki oso bat lortu arte errepikatu beharko da.

Erabili beharreko funtzioak:

- *eskatu_zenbaki_handiagoa*: argumentu bezala z zenbaki oso bat emanda, erabiltzaileari z baino handiagoa den zenbaki bat eskatzen dion funtzioa. Datua eskatzeko prozesua datu egoki bat lortu arte errepikatuko da.
- *lehenengo_faktorialak_erroekin*: ≥ 1 den z zenbaki oso bat emanda, 0 eta z – 1 zenbakien arteko (biak barne) balio denen faktorialak eta faktorial horien erroak kalkulatu eta pantailan aurkezten dituen funtzioa.
- *kalkulatu_faktoriala*: argumentu bezala 0 baino handiagoa edo berdina den x zenbaki oso bat emanda, x zenbaki horren faktoriala kalkulatzen duen funtzioa.
- *lehenengo_lehenak*: ≥ 1 den z zenbaki oso bat emanda, lehenengo z zenbaki lehenak kalkulatu eta pantailan aurkezten dituen funtzioa.
    - *lehena_al_da*: argumentu bezala ≥ 1 den *x* zenbaki oso bat emanda, zenbakia lehena bada, 1, eta lehena ez bada, 0 itzultzen duen funtzioa.

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

/* Datuak: Programa honek erabiltzaileari >= 1 den n zenbaki bat
           eskatuko dio. */

/* Emaitzak: Lehenengo n faktorialak, faktorial horien erro karratuak
             eta lehenengo n zenbaki lehenak kalkulatu eta aurkeztuko
             ditu. */

/* Aldagaiak:
       - n: erabiltzaileak tekleatuko duen zenbaki osoa gordetzeko. */

/* Programan erabiliko diren funtzioen prototipoak. */
int eskatu_zenbaki_handiagoa (int z);
void lehenengo_faktorialak_erroekin (int z);
long kalkulatu_faktoriala (int x);
void lehenengo_lehenak (int z);
int lehena_al_da (int x);


/* Funtzio nagusia*/
void main()
 { /* main funtzioaren hasierako giltza */
   int n;

   n = eskatu_zenbaki_handiagoa(0);/* >= 1 den zenbaki bat eskatu */

   lehenengo_faktorialak_erroekin(n);
   lehenengo_lehenak(n);

   printf("\nSakatu tekla bat amaitzeko.");
```

```
   getch();  /*  Erabiltzaileak  tekla  bat  sakatu  zain  egongo  da
ordenagailua. */
 } /* main funtzioaren bukaerako giltza */

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

                /* Erabilitako funtzioen definizioak */

int eskatu_zenbaki_handiagoa (int z)
 /* Funtzio honek erabiltzaileari z baino handiagoa den zenbaki oso
    bat eskatuko dio eta erabiltzaileak zenbaki egoki bat
    tekleatutakoan, zenbaki egoki hori itzuliko du emaitza bezala.
    Zenbakia eskatzeko prozesua zenbaki egoki bat lortu arte
    errepikatuko da. */

 { /* Funtzioaren hasierako giltza */
   int zenb; /* erabiltzaileak tekleatuko duen zenbakia jasotzeko. */

   do
   { printf("\n%d baino handiagoa den zenbaki oso bat tekleatu: ", z);
     scanf("%d", &zenb);
     if (zenb <= z)
      {
        printf("\nTekleatutako zenbakia ez da egokia.");
      }
   } while (zenb <= z);

   return(zenb);
 } /* Funtzioaren bukaerako giltza */

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

void lehenengo_faktorialak_erroekin (int z)

 /* Funtzio honek lehenengo z faktorialak eta beraien erro karratuak
    kalkulatu eta aurkeztuko ditu (z >= 1). */

 {
   int zenb, fakto;
   float r;


   printf("\nLehenengo %d faktorialak beraien erro karratuekin: ", z);

   for (zenb = 0; zenb < z; zenb = zenb + 1)
    {
      fakto = kalkulatu_faktoriala(zenb);
      r = sqrt(fakto);
      printf("\n%d(r)en faktoriala %d da eta faktorial horren "
             "erro karratua %f da.", zenb, fakto, r);
    }
 }

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

long kalkulatu_faktoriala (int x)

 /* Funtzio honek argumentu bezala >= 0 den x zenbaki osoa emanda,
```

```
    bere faktoriala kalkulatu eta itzuliko du. */

 { /* Funtzioaren hasierako giltza */
   long e;
   int zenb;

   if (x == 0)
    {
      e = 1;
    }
   else
    {
      e = 1;
      for (zenb = 1; zenb <= x; zenb = zenb + 1)
       {
         e = e * zenb;
       }
    }
   return(e);
 } /* Funtzioaren bukaerako giltza */

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


void lehenengo lehenak (int z)

 /* Funtzio honek lehenengo z zenbaki lehenak aurkitu eta aurkeztuko
    ditu. */

 {
   int zenb, kont;

   zenb = 1; /* 1etik hasi eta zenbakiak banan-banan pasatuz joateko
                erabiliko da. While-eko buelta bakoitzean zenb-en
                dagoen balioa lehena al den ala ez aztertuko da. */

   kont = 0; /* Aldagai honek une bakoitzean une horretara arte zenbat
                zenbaki lehen aurkitu diren adieraziko digu. */

   printf("\nLehenengo %d zenbaki lehenak honako hauek dira: ", z);

   while (kont < z)
    {
      if (lehena_al_da(zenb) == 1)
       {
         printf("%d  ");
         kont = kont + 1;
       }
      zenb = zenb + 1;
    }
 }

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

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. */
```

```
{ /* Funtzioaren hasierako giltza */
  int zenb, kont;

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

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

  if (kont == 2)
   {
     return (1);
   }
  else
   {
     return (0);
   }
} /* Funtzioaren bukaerako giltza */

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

4 ARIKETAK

4.1 1. ariketa

4.1.1 Helburua:

1. ariketako helburua main funtzioa ez den funtzio batean sqrt funtzio matematiko aurredefinitua erabiltzea da.

4.1.2 1. enuntziaturako laguntza:

- Erro karratua sqrt funtzio matematiko aurredefinitua erabiliz kalkulatzen da, hau da, sqrt(x) ipintzen badugu, $\sqrt{x}$ kalkulatuko dugu. Funtzio hori erabili ahal izateko programaren hasieran edo goi-buruan **#include <math.h>** ipini beharko da.

- sqrt funtzioa *erro_positiboa_kalkulatu* eta *erro_negatiboa_kalkulatu* funtzioen barruan erabiliko da.

- *erro_positiboa_kalkulatu* eta *erro_negatiboa_kalkulatu* funtzioak *kalkulatu_erroak* funtzioaren barruan erabiliko dira.

- Ekuazioak erro bakarra duenean, erro bakar hori kalkulatzeko *erro_positiboa_kalkulatu* edo *erro_negatiboa_kalkulatu* erabil daiteke, hau da, bietako edozein.

4.1.3 Enuntziatua:

Erabiltzaileari bigarren mailako ekuazio baten hiru koefizienteak eskatu, erroak kalkulatu eta aurkeztu (erroak baldin baditu behintzat) eta erabiltzaileak bigarren mailako koefiziente bezala 0 tekleatu arte prozesua errepikatzen duen programa idatzi.

Bigarren mailako ekuazio batek honako itxura hau izaten du: $ax^2 + bx + c = 0$. Ekuazio horretan a, b eta c koefizienteak dira eta a bigarren mailako koefizientea da.

Erroak kalkulatzeko formula orokorra honako hau da: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Bigarren mailako ekuazio baten kasuan bi erro edo erro bat edukitzea edo errorik ez edukitzea gerta daiteke:

- $b^2 - 4ac < 0$ baldin bada, ez du errorik izango.
- $b^2 - 4ac = 0$ baldin bada, erro bakarra izango du.
- $b^2 - 4ac > 0$ baldin bada, bi erro izango ditu.

Bigarren mailako ekuazioen adibideak:

- $2x^2 + 1x + 2 = 0$ ekuazioak ez du errorik.
- $2x^2 + 4x + 2 = 0$ ekuazioak erro bakarra du: -1
- $2x^2 + 5x + 2 = 0$ ekuazioak bi erro ditu: -0.5 y -2

Jarraian aipatzen diren **funtzioak** definitu eta erabili behar dira:

- *kalkulatu_erroak*: bigarren mailako ekuazio baten a, b eta c koefizienteak emanda, ondoren zehazten dena egiten duen funtzioa:
  - Ekuazioak errorik ez badu, hori adierazten duen mezu bat aurkeztuko du.
  - Ekuazioak erro bakarra badu, erro hori kalkulatu eta aurkeztuko du, ekuazioak erro bakarra duela jakinaraziz.
  - Ekuazioak bi erro baditu, kalkulatu eta aurkeztu egingo ditu.

- *erro_positiboa_kalkulatu*: $b^2 - 4ac \geq 0$ baldintza betetzen duten bigarren mailako ekuazio baten a, b eta c koefizienteak emanda, $\frac{-b + \sqrt{b^2 - 4ac}}{2a}$ kalkulatu eta itzultzen duen funtzioa.

- *erro_negatiboa_kalkulatu*: $b^2 - 4ac \geq 0$ baldintza betetzen duten bigarren mailako ekuazio baten a, b eta c koefizienteak emanda, $\frac{-b - \sqrt{b^2 - 4ac}}{2a}$ kalkulatu eta itzultzen duen funtzioa.

Exekuzio-adibidea: (erabiltzaileak tekleatutako datuak *letra etzanez eta azpimarratuta* ageri dira)

Pantaila

```
Bigarren mailako ekuazio baten koefizienteak sartu: 2, 1, 2
Ekuazioak ez du errorik.
Bigarren mailako ekuazio baten koefizienteak sartu: 2, 4, 2
Ekuazioak erro bakarra du: -1
```

```
Bigarren mailako ekuazio baten koefizienteak sartu: 2, 5, 2
Ekuazioak bi erro ditu: -0.5 y -2
Bigarren mailako ekuazio baten koefizienteak sartu: 0, 3, 5
Sakatu tekla bat amaitzeko.
```

4.2 2. ariketa

4.2.1 Helburua:

2. ariketako helburua bi funtzio definitzea eta bietako bat bestearen barruan erabiltzea da. Funtzio biek emaitza bat kalkulatu eta itzuliko dute.

4.2.2 2. enuntziaturako laguntza:

- nire_pow funtzioa definitzeko, hau da x^y nola kalkulatzen den adierazteko, $x^y = x * x * x * ...$ dela kontuan hartu beharko da (x zenbakia y aldiz biderkatuta). Funtzio honek osoak eta positiboak diren zenbakientzat bakarrik balioko du eta beraz ez da aurredefinitutako pow funtzioa bezain orokorra, pow-k zenbaki errealentzat ere balio du eta.

- *nire_pow* funtzioa *erroa_kalkulatu* funtzioan erabiliko da.

- *erroa_kalkulatu* definitzerakoan $\sqrt[y]{x}$ hurbiltzeko, 0^y, 1^y, 2^y, 3^y, eta abar kalkulatuz joan beharko da x baino handiagoa den berredura bat lortu arte. Emaitza x baino handiagoa den berreduraren aurreko berredurako zenbakia izango da. Adibidez $\lfloor\sqrt[3]{85}\rfloor$ kalkulatzeko 0^3, 1^3, 2^3, 3^3, 4^3 eta 5^3 kalkulatuko lirateke. 5^3 balioa 85 baino handiagoa denez, emaitza 4 da (aurreko berredurako zenbakia).

4.2.3 Enuntziatua:

Erabiltzaileari m (≥ 1) eta n (≥ 1) bi zenbaki oso eskatu, $\lfloor\sqrt[n]{m}\rfloor$ kalkulatu (m-ren n-garren mailako erroaren hurbilpena azpitik) eta jarraian erabiltzaileari beste errorik kalkulatu nahi al duen galdetzen dion programa idatzi. Erabiltzaileak baietz ('s') erantzuten badio, programak beste bi zenbaki oso, m (≥ 1) eta n (≥ 1), eskatu beharko dizkio erabiltzaileari eta hauei dagokien erroa kalkulatu beharko du. Erabiltzaileak errorik ez duela kalkulatu nahi gehiago esan arte ('n') errepikatu beharko da prozesua. Erabiltzaileak 'b' edo 'e' tekleatuz erantzun behar duen bakoitzean, ez da beste erantzunik onartuko eta galdetze-prozesua bi karaktere horietakoren bat lortu arte errepikatuko da. Bestalde, m eta n eskatzerakoan ≥ 1 direla egiaztatu beharko da, zenbaki egokiak lortu arte eskatze-prozesua errepikatuz. **Aurredefinitutako funtzio matematikoak ezingo dira erabili**.

Erroen hurbilpenen adibideak:
Hurbilpen hauek azpitik egindako hurbilpenak direla esaten da benetako emaitzaren zati osoa bakarrik kalkulatzen delako:

- ✓ $\lfloor\sqrt[3]{8}\rfloor = 2$
- ✓ $\lfloor\sqrt[3]{10}\rfloor = 2$

- ✓ $\lfloor \sqrt[3]{29} \rfloor = 3$

Jarraian aipatzen diren **funtzioak** definitu eta erabili behar dira:

- *erroa_kalkulatu*: x (≥ 1) eta y (≥ 1) bi zenbaki oso emanda, $\lfloor \sqrt[y]{x} \rfloor$ kalkulatu eta itzultzen duen funtzioa.
- *nire_pow*: x (≥ 1) eta y (≥ 1) bi zenbaki oso emanda, pow erabiltzeke x^y kalkulatu eta itzultzen duen funtzioa.
- *galdetu_erroa*: erabiltzaileari beste errorik kalkulatu nahi al duen ala ez galdetzen dion funtzioa. Galdetze-prozesua 'b' edo 'e' lortu arte errepikatuko da.

Exekuzio-adibidea: (erabiltzaileak tekleatutako datuak *letra etzanez eta azpimarratuta* ageri dira)

Pantaila

```
>= 1 diren bi zenbaki oso sartu: 5, -2
Sartutako datuak ez dira egokiak.
>= 1 diren bi zenbaki oso sartu: 10, 3
10(r)en erroa 3(r)ekiko 2 da.
Beste errorik kalkulatu nahi al duzu? (b/e): q
Tekleatutako datua ez da egokia.
Beste errorik kalkulatu nahi al duzu? (b/e): b
>= 1 diren bi zenbaki oso sartu: 29, 3
29(r)en erroa 3(r)ekiko 3 da.
Beste errorik kalkulatu nahi al duzu? (b/e): e
Sakatu tekla bat amaitzeko.
```

4.3 3. ariketa

4.3.1 Helburua:

3. ariketako helburua hiru funtzio definitu eta beraietako bat beste baten barruan erabiltzea da. Horrela, funtzio batzuk main funtzioan erabiliko direla eta beste batzuk aldiz beste funtzioetan erabiliko direla ikusi ahal izango dugu. Gainera funtzioetako bat lehendik definituta dago (beste ariketa batean) eta hori abantaila handia da.

4.3.2 3. enuntziaturako laguntza:

- *tartagliaren_triangelua_aurkeztu* funtzioa definitzerakoan (0, 0)-tik (x – 1, x – 1)-erainoko bikote denak sortuz (lerro eta zutabeen arteko konbinazio posible denak alegia) eta bikote bakoitzeko *konbinatorioa_kalkulatu* funtzioari deituz joan beharko da.
- *konbinatorioa_kalkulatu* funtzioak *kalkulatu_faktoriala* funtzioa erabiliko du.
- Gogora dezagun 0ren faktoriala 1 dela eta beste edozein x zenbakiren faktoriala kalkulatzeko 1etik x-erainoko zenbakiak biderkatu behar direla.

4.3.3 Enuntziatua:

Erabiltzaileari ≥ 1 den n zenbaki oso bat eskatu, n-ri dagokion Tartaglia-ren triangelua pantailan aurkeztu, berriro beste n balio bat eskatu, dagokion triangelua aurkeztu eta

horrela erabiltzaileak zenbaki negatibo bat tekleatu arte prozesua errepikatzen duen programa idatzi.

Adibidez n = 5 balioarentzat Tartagliaren triangeluak 5 lerro eta 5 zutabe izango lituzke eta honako itxura hau izango luke:

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
```

Lerroetan eta zutabeetan fijatuz:

	zu = 0	zu = 1	zu = 2	zu = 3	zu = 4
le = 0	1				
le = 1	1	1			
le = 2	1	2	1		
le = 3	1	3	3	1	
le = 4	1	4	6	4	1

le lerroko eta zu zutabeko elementua honako formularen bidez kalkulatzen da (zenbaki konbinatorioa): $\binom{le}{zu} = \frac{le!}{zu!(le-zu)!}$

Jarraian aipatzen diren **funtzioak** definitu eta erabili behar dira:

- *tartagliaren_triangelua_aurkeztu*: ≥ 1 den x zenbaki oso bat emanda, x lerro eta x zutabe dituen Tartagliaren triangelua pantailan aurkezten duen funtzioa.
- *konbinatorioa_kalkulatu*: x eta y (x ≥ 0, y ≥ 0, x ≥ y) bi zenbaki oso emanda, $\binom{x}{y}$ kalkulatu eta itzultzen duen funtzioa.
- *kalkulatu_faktoriala*: ≥ 0 den x zenbaki oso bat emanda, x-en faktoriala kalkulatu eta itzultzen duen funtzioa.

Exekuzio-adibidea: (erabiltzaileak tekleatutako datuak *letra etzanez eta azpimarratuta* ageri dira)

Pantaila

```
>= 1 den zenbaki oso bat sartu: 2
Dagokion Tartagliaren triangelua:
1
1 1
1 2 1
>= 1 den zenbaki oso bat sartu: 4
Dagokion Tartagliaren triangelua:
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
>= 1 den zenbaki oso bat sartu: -9
Sakatu tekla bat amaitzeko.
```

4.4 4. ariketa

4.4.1 Helburua:

4. ariketako helburua bi funtzio berri definitu eta bietako bat bestearen barruan erabiltzea da, programak egituratzen ikasiz.

4.4.2 4. enuntziaturako laguntza:

- *logaritmoak_aurkeztu* funtzioa x eta y-rekiko definitzerakoan, logaritmoen oinarriak 2tik x-eraino aldatuz joan beharko dugu eta oinarri bakoitzeko *logaritmoa_kalkulatu* funtzioari deitu beharko zaio argumentu bezala oinarria eta y pasatuz.

- *logaritmoa_kalkulatu* funtzioa x eta y-rekiko definitzerakoan, x balioa bere buruaz biderkatuz joan beharko da (x * x * x * x * ...) y baino handiagoa den balio bat lortu arte. Funtzioak itzuli beharreko emaitza biderkaketa-kopurua ken 1 izango da.

4.4.3 Enuntziatua:

Erabiltzaileari osoak diren m eta n (m ≥ 1, n ≥ 1 eta n ≥ m) bi zenbaki eskatu eta $\lfloor \log_2 n \rfloor$, $\lfloor \log_3 n \rfloor$, $\lfloor \log_4 n \rfloor$, ..., $\lfloor \log_m n \rfloor$ elementuak (azpitik gelditzen diren logaritmoen hurbilpen osoak) eta beraien arteko batura kalkulatu eta aurkeztu eta jarraian erabiltzaileari beste bi baliorentzat kalkuluak errepikatu nahi al dituen galdetzen dion programa bat idatzi. Erabiltzaileak baietz ('b') erantzuten badu, programak beste bi balio (m eta n) eskatu beharko dizkio erabiltzaileari eta balio berri horiekin dagozkien logaritmoak eta batura kalkulatu beharko ditu. Prozesua erabiltzaileak logaritmo gehiago ez ('e') duela nahi erantzun arte errepikatu beharko da. Erabiltzaileak 'b' edo 'e' tekleatuz erantzun behar duen bakoitzean ez da beste erantzunik onartuko, galdetze-prozesua bi karaktere horietakoren bat lortu arte errepikatuz. Bestalde, m eta n eskatzerakoan ere ≥ 1 direla eta gainera n ≥ m betetzen dela egiaztatu beharko da. **Aurredefinitutako funtzio matematikoak ezin dira erabili**.

Logaritmo baten azpitik gelditzen den hurbilpen osoa kalkulatzerakoan, hau da $\lfloor \log_m n \rfloor$ kalkulatzerakoan, n baino handiagoa den balio bat lortzeko m bere buruaz zenbat aldiz biderkatu behar den (m * m * m * …) kalkulatu beharko da. Emaitza biderkaketa-kopuru hori ken 1 izango da:

Logaritmoen azpitik gelditzen diren hurbilpen osoak:

- $\lfloor \log_2 8 \rfloor = 3$ da, 8 baino handiagoa den balio bat lortzeko 2 zenbakia 4 aldiz biderkatu behar baita bere buruaz (2 * 2 * 2 * 2) eta ondorioz emaitza 4 – 1 da, 3 beraz.
- $\lfloor \log_2 12 \rfloor = 3$ da, 12 baino handiagoa den balio bat lortzeko 2 zenbakia 4 aldiz biderkatu behar baita bere buruaz (2 * 2 * 2 * 2) eta ondorioz emaitza 4 – 1 da, 3 beraz.
- $\lfloor \log_4 20 \rfloor = 2$ da, 20 baino handiagoa den balio bat lortzeko 4 zenbakia 3 aldiz biderkatu behar baita bere buruaz (4 * 4 * 4) eta ondorioz emaitza 3 – 1 da, 2 alegia.

Jarraian aipatzen diren **funtzioak** definitu eta erabili behar dira:

- *logaritmoak_aurkeztu*: osoak diren x eta y (x ≥ 1, y ≥ 1 eta y ≥ x) bi zenbaki emanda, $\lfloor \log_2 y \rfloor$, $\lfloor \log_3 y \rfloor$, $\lfloor \log_4 y \rfloor$, ..., $\lfloor \log_x y \rfloor$ balioak eta balio horien batura kalkulatzen eta aurkezten dituen funtzioa.

- *logaritmoa_kalkulatu*: x eta y (x ≥ 1, y ≥ 1, y ≥ x) bi zenbaki oso emanda, $\lfloor \log_x y \rfloor$ kalkulatu eta itzultzen duen funtzioa.

- *logaritmo_galdetu*: erabiltzaileari beste logaritmo-sekuentziarik kalkulatu nahi al duen galdetzen dion funtzioa. Galdetze-prozesua 'b' edo 'e' lortu arte errepikatu beharko da.

Exekuzio-adibidea: (erabiltzaileak tekleatutako datuak *letra etzanez eta azpimarratuta* ageri dira)

Pantaila

```
>= 1 eta 1.a <= 2.a diren bi zenbaki oso sartu: 5, -2
Sartutako datuak ez dira egokiak.
>= 1 eta 1.a <= 2.a diren bi zenbaki oso sartu: 4, 20
log(2)20 = 4  log(3)20 = 2  log(4)20 = 2
Batura 8 da.
Logaritmo gehiago nahi al duzu? (b/e): q
Tekleatutako datua ez da egokia.
Logaritmo gehiago nahi al duzu? (b/e): b
>= 1 eta 1.a <= 2.a diren bi zenbaki oso sartu: 3, 50
log(2)50 = 5  log(3)50 = 3
Batura 8 da.
Logaritmo gehiago nahi al duzu? (b/e): e
Sakatu tekla bat amaitzeko.
```

4.5 5. ariketa

4.5.1 Helburua:

5. ariketako helburua funtzio berri bat definitzea eta lehendik (beste ariketa batean) definituta dagoen funtzio bat lehenengoaren barruan erabiltzea da.

4.5.2 5. enuntziaturako laguntza:

- main funtzioan erabiltzaileari n zenbaki bat eskatuko zaio eta *erabaki_fibonaccia* funtzioari pasatuko zaio eta funtzio horrek itzultzen duen emaitzarekin mezu bat aurkeztuko da, gero berriro beste zenbaki bat eskatu eta funtzio horri pasatu eta abar, erabiltzaileak zenbaki negatibo bat eman arte.

- *erabaki_fibonaccia* funtzioan 0ren Fibonaccia, 1ena, 2rena eta abar kalkulatuz joan beharko da, x-en berdina edo handiagoa den Fibonacci bat lortu arte. Kasuren batean x balioa lortzen bada Fibonacci bezala, aurkitu dugu nahi genuen zenbakia baina emaitza bezala x lortu baino lehen x baino handiagoa den Fibonacci bat agertzen bada, x inoren Fibonaccia ez dela esan nahiko du horrek. Fibonacciak kalkulatzeko *kalkulatu_fibonaccia* funtzioari deitu beharko zaio.

4.5.3 Enuntziatua:

Erabiltzaileari ≥ 0 den n zenbaki oso bat eskatu eta bere Fibonaccia n den zenbarik existitzen al den ala ez erabaki eta prozesu bera behin eta berriz, erabiltzaileak zenbaki negatibo bat tekleatu arte, errepikatzen duen programa bat idatzi.

≥ 1 den zenbaki oso baten **Fibonaccia** honela definitzen da:

Fibonaccia(0) = 0
Fibonaccia(1) = 1
Fibonaccia(x) = Fibonaccia(x – 2) + Fibonaccia(x – 1)

Fibonaccien adibideak:

x	0	1	2	3	4	5	6	7	8	9	10	11	…
Fibonaccia(x)	0	1	1	2	3	5	8	13	21	34	55	89	…

Jarraian aipatzen diren **funtzioak** definitu eta erabili behar dira:

- *erabaki_fibonaccia*: ≥ 0 den x zenbaki oso bat emanda, x beste zenbakiren baten Fibonaccia al den ala ez erabakitzen duen funtzioa. Ez bada, -1 itzuli beharko du eta bestela beste zenbaki hori itzuli beharko du.

- *kalkulatu_fibonaccia*: ≥ 0 den x zenbaki oso bat emanda, x-en Fibonaccia kalkulatzen duen funtzioa.

Exekuzio-adibidea: (erabiltzaileak tekleatutako datuak *letra etzanez eta azpimarratuta* ageri dira)

Pantaila

```
>= 0 den zenbaki oso bat sartu: 4
4 ez da inoren Fibonaccia.
>= 0 den zenbaki oso bat sartu: 8
8 6(r)en Fibonacci da.
>= 0 den zenbaki oso bat sartu: -5
Sakatu tekla bat amaitzeko.
```