# 5. LABORATEGIA
# FUNTZIOAK

## 1 HELBURUAK

Laborategi hau amaitutakoan ondoren aipatzen direnak egiteko gai izango zara:

- Aurredefinitutako funtzio matematikoak main funtzioan erabili.
- Programa baten barruan azpikalkuluak edo azpiprogramak identifikatu eta mugatu daitezkeela eta azpikalkulu horiek burutzen dituzten funtzio independenteak defini daitezkeela ulertu.
- Funtzio berriak definitu eta main funtzioaren barruan erabili.

## 2 MOTIBAZIOA

- Askotan nahiko konplexua den azpikalkulu bat behin baino gehiagotan burutzen da programa berean edota programa desberdinetan agertzen da. Azpikalkulu hori ebazten duen funtzioa definituz, azpikalkulua burutu behar den bakoitzean ez da egongo bere agindu denak idatzi beharrik, funtzioa behin definitzea eta behar den guztietan funtzioari deitzea nahikoa izango baita.
- Azpiproblemak isolatu eta funtzio independenteen bidez ebazteak programak egituratzea eta ulertzen errazagoak izatea ahalbidetzen du.

### 2.1 Funtzioak C-n

- Alde batetik C-n funtzio aurredefinituak ditugu. Adibidez, balio absolutua, **fabs(x)**, erro karratua, **sqrt(x)**, eta berredura, **pow (x, y)**, kalkulatzeko balio duten funtzioak. Funtzio horiek erabiltzeko programaren hasieran edo goi-buruan **#include<math.h>** ipintzea nahikoa izango da.
- Beste aldetik programatzaileak ere **funtzio berriak** definitu eta erabili ditzake. Funtzio bat definitzerakoan itzuliko duen **emaitzaren mota**, **funtzioaren izena** eta **argumentuen motak eta izenak** adierazi beharko dira.
- Funtzioak **emaitzik ez badu** itzultzen edota **argumenturik ez badu,** emaitzaren motari edota argumentuen motei eta izenei dagokien lekuan **void** hitza ipini beharko da.
- Programatzaileak definitutako funtzioak erabiltzeko, aukera bat main funtzioaren aurrean **prototipoak** ipintzea eta main funtzioaren ondoren **definizio** denak jarraian ipintzea da.
- **Programak funtzio batez baino gehiagoz osatuta edo eratuta egongo dira**: **main funtzioa beti edukiko dugu** eta main funtzioaz gain geuk definitutako funtzioak egongo dira.
- **Ordenagailuak beti main funtzioa exekutatzen du** eta main funtziotik beste funtzioren bati deitzen bazaio, orduan funtzio horri dagozkion aginduak burutuko dira pasatzen zaizkion argumentuak kontuan hartuz.
- **Funtzio batek era desberdinetako eginkizunak buru ditzake:**
  - Kalkuluak egin eta emaitza itzuli.
  - Kalkuluak egin eta emaitzak pantailan aurkeztu.
  - Erabiltzaileari datuak eskatu.
  - Emaitzak aurkeztu.
  - Beste funtzioei deitu funtzio horiek azpikalkuluak burutu ditzaten.

# 3 ADIBIDE ARIKETA

## 3.1 Helburua:

- Funtzio matematikoak main funtzioan nola erabiltzen diren azaldu.
- Funtzio berriak nola definitzen diren eta main funtzioaren barruan nola erabiltzen diren erakutsi.

## 3.2 Enuntziatua:

Zero baino handiagoak edo berdinak diren bi zenbaki oso eskatu erabiltzaileari, zenbaki bakoitzaren faktoriala kalkulatu, pantailan aurkeztu eta, jarraian, lehenengo faktorialaren eta bigarren faktorialaren arteko zatiduraren erro karratua 1 baino handiagoa edo berdina bada, lehenengo faktoriala ber bigarren faktoriala kalkulatu eta aurkeztu eta bestela bigarrena ber lehenengoa kalkulatu eta aurkezten duen programa idatzi. Hasierako datuak eskatzerakoan >= 0 diren bi zenbaki lortu arte errepikatu beharko da eskatze-prozesua.

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.

- *kalkulatu_faktoriala*: argumentu bezala 0 baino handiagoa edo berdina den x zenbaki oso bat emanda, x zenbaki horren faktoriala kalkulatzen duen funtzioa.

- *aurkeztu_faktoriala*: argumentu bezala $\geq 0$ den x zenbaki oso bat eta bere faktoriala de f balioa emanda, x-en faktoriala f dela esanez pantailan mezu bat aurkezten duen funtzioa.

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

/* Datuak: Programa honek >= 0 diren bi zenbaki oso eskatuko
           dizkio erabiltzaileari (n1 eta n2). */

/* Emaitzak: n1 eta n2-ren faktorialak kalkulatu eta aurkeztuko ditu
             eta gainera, 1. faktorialaren eta 2. faktorialaren arteko
             zatiduraren erro karratua 1 baino handiagoa edo berdina
             bada, 1. faktoriala ber 2.a kalkulatu eta aurkeztuko du
             eta bestela 2.a ber 1.a. */

/* Aldagaiak:
          - n1, n2: erabiltzaileak emango dituen bi zenbakiak
                    gordetzeko.
          - f1, f2: n1 eta n2-ren faktorialak gordetzeko. */

/* Programan erabiliko diren funtzioen prototipoak. */

int eskatu_zenbaki_handiagoa (int z);
long kalkulatu_faktoriala (int x);
void aurkeztu_faktoriala (int x, long f);
```

```
/* Funtzio nagusia */

void main()
 { /* main funtzioaren hasierako giltza */
   int n1, n2;
   long f1, f2, p;

   n1 = eskatu_zenbaki_handiagoa (-1);/* >= 0 den zenbaki bat eskatu */
   n2 = eskatu_zenbaki_handiagoa (-1);

   f1 = kalkulatu_faktoriala (n1);
   f2 = kalkulatu_faktoriala (n2);

   aurkeztu_faktoriala (n1, f1);
   aurkeztu_faktoriala (n2, f2);

   if (sqrt(f1/f2) >= 1)
     {
       p = pow(f1, f2);
       printf("\nLehenengo faktoriala ber bigarrena %ld da", p);
     }
   else
     {
       p = pow(f2, f1);
       printf("\nBigarren faktoriala ber lehenengoa %ld da ", p);
     }

  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
                erabiliko da. */

   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 */

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

```
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 aurkeztu_faktoriala (int x, long f)

 /* Funtzio honek f zenbakia x-en faktoriala dela esanez mezu bat
    aurkeztuko du pantailan. */

 { /* Funtzioaren hasierako giltza */

   printf("\n%d zenbakiaren faktoriala %ld da.", x, f);

 } /* Funtzioaren bukaerako giltza */

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

# 4 ARIKETAK

## 4.1 1. ariketa

### 4.1.1 Helburua:

Lehenengo ariketaren helburua aurredefinitutako pow funtzioa main funtzioaren barruan erabiltzea da.

### 4.1.2 1. enuntziaturako laguntza:

Gogoratu aurredefinitutako pow funtzio matematikoak berredura kalkulatzen duela, hau da, **pow(x, y)** = $\mathbf{x^y}$ da. Funtzio hori erabili ahal izateko **#include <math.h>** ipini beharko da programaren hasieran edo goi-buruan.

### 4.1.3 Enuntziatua:

Erabiltzaileari zenbakiak binaka eskatuz eta kasu bakoitzean lehenengoa ber bigarrena kalkulatu eta aurkezten duen programa idatzi. Prozesua erabiltzaileari bi zenbaki eskatutakoan honek sartutako bietatik gutxienez bat $\leq 0$ denean bukatuko da.
**Exekuzio-adibidea**: (erabiltzaileak tekleatutako datuak *letra etzanez eta azpimarratuta* ageri dira)

Pantalla

```
>= 1 diren bi zenbaki oso komaz bereiztuta sartu: 3, 2
3 ber 2 9 da.
>= 1 diren bi zenbaki oso komaz bereiztuta sartu: 2, 10
2 ber 10 1024 da.
>= 1 diren bi zenbaki oso komaz bereiztuta sartu: -5, 30
Sakatu tekla bat amaitzeko.
```

## 4.2 2. ariketa

### 4.2.1 Helburua:

2. ariketaren helburua hiru funtzio definitzea eta funtzio horiek main funtzioan erabiltzea da:

- Lehenengoak erabiltzaileari datu bat eskatuko dio pasatzen zaion argumentua kontuan hartuz.
- Bigarrenak bere argumentua erabiliz kalkulu bat burutuko du eta emaitza bat itzuliko du main funtzioan erabilia izan dadin.
- Hirugarrenak emango zaizkion argumentuak kontuan hartuz mezu bat aurkeztuko du pantailan.

### 4.2.2 2. enuntziaturako laguntza:

- Lehenengo n zenbaki beteak aurkitzeko 1etik hasi eta zenbakiak banan-banan pasatuz eta zenbaki bakoitza betea al den ala ez erabakiz (*betea_al_da* funtzioaren bidez) joan beharko da. Prozesua n zenbaki bete aurkitutakoan bukatuko da.

- Zenbaki bat betea al den ala ez erabakitzeko, 1etik hasi eta zenbakiak banan-banan pasatuz eta zatitzaileak direnen batura kalkulatuz joan beharko da. Zatitzaileen batura emandako zenbakiaren berdina bada, zenbakia betea da eta bestela ez.

### 4.2.3 Enuntziatua:

Erabiltzaileari $\geq 1$ den n zenbaki oso bat eskatu eta lehenengo n **zenbaki beteak** kalkulatzen dituen programa idatzi. Erabiltzaileari n balioa eskatzerakoan eskatze-prozesua aipatutako baldintza betetzen duen zenbaki bat lortu arte errepikatu beharko da.

1 baino handiagoa edo berdina den n **zenbaki** oso bat **betea** dela esaten da bere zatitzaileen batura (n bera kontuan hartzeke) n bada.

**Zenbaki bete eta ez beteen adibideak**:

- betea da 1 + 2 + 3 = 6 delako.
- 28 betea da 1 + 2 + 4 + 7 + 14 = 28 delako.
- 12 ez da betea 1 + 2 + 3 + 4 + 6 $\neq$ 12 baita.

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

- *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.

- *betea_al_da*: argumentu bezala ≥ 1 den *x* zenbaki oso bat emanda, zenbakia betea bada, 1, eta betea ez bada, 0 itzultzen duen funtzioa.

- *aurkeztu_betea*: argumentu bezala *x* zenbaki bat eta *x* zenbakiari buruzko *e* erabakia emanda, *e*-ren balioa 1 bada, *x* betea dela esanez mezu bat aurkezten duen funtzioa, eta *e* 0 bada, *x* ez dela betea esanez mezu bat aurkezten duen funtzioa.

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

Pantaila

```
>= 1 den zenbaki oso bat sartu: -4
Sartutako zenbakia ez da egokia.
>= 1 den zenbaki oso bat sartu: 2
6 betea da.
28 betea da.
Sakatu tekla bat amaitzeko.
```

## 4.3 3. ariketa

### 4.3.1 Helburua:

3. ariketaren helburua aurreko ariketan definitu diren hiru funtzioak berriz erabiltzea da. Ariketa biak desberdinak dira baina bietan hiru azpiproblema horiek agertzen dira. Honela funtzioak ariketa desberdinetan errepika daitezkeen azpiproblemak ebazteko eta aldatu ere egin gabe ariketa desberdinetan erabili ahal izateko balio dutela ikus daiteke. Bi ariketa hauetan (2.a eta 3.a) main funtzioa aldatuko da baina beste hiru funtzioak berdin berdinak izango dira.

### 4.3.2 3. enuntziaturako laguntza:

n baino txikiagoak diren zenbaki beteak aurkitzeko 1etik hasi eta zenbakiak banan-banan pasatuz joan beharko da n-ra iritsi arte. Kasu bakoitzean zenbakia betea al den ala ez erabaki beharko da *betea_al_da* funtzioari deituz.

### 4.3.3 Enuntziatua:

Erabiltzaileari osoa eta positiboa (≥ 1) den n zenbaki bat eskatu eta n baino txikiagoak diren zenbaki beteak kalkulatzen dituen programa idatzi. Erabiltzaileari n zenbakia eskatzerakoan aipatutako baldintza betetzen duen zenbaki bat lortu arte errepikatu beharko da eskatze-prozesua.

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

- *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.

- *betea_al_da*: argumentu bezala ≥ 1 den *x* zenbaki oso bat emanda, zenbakia betea bada, 1, eta betea ez bada, 0 itzultzen duen funtzioa.

- *aurkeztu_betea*: argumentu bezala *x* zenbaki bat eta *x* zenbakiari buruzko *e* erabakia emanda, *e*-ren balioa 1 bada, *x* betea dela esanez mezu bat aurkezten duen funtzioa, eta *e* 0 bada, *x* ez dela betea esanez mezu bat aurkezten duen funtzioa.

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

Pantaila

```
>= 1 den zenbaki oso bat sartu: -4
Sartutako zenbakia ez da egokia.
>= 1 den zenbaki oso bat sartu: 40
6 betea da.
28 betea da.
Sakatu tekla bat amaitzeko.
```

## 4.4 4. ariketa

### 4.4.1 Helburua:

4. ariketako helburua bi funtzio berri definitu eta funtzio horiek main funtzioan erabiltzea da.

### 4.4.2 4. enuntziaturako laguntza:

Zenbaki bat erdigunea al den ala ez erabakitzeko, hasteko bere aurreko denen batura kalkulatu beharko da (1etik hasita) eta gero bere ondoren jarraian dauden zenbakien baturekin frogatuz joan beharko da aurrekoen baturaren berdina edo handiagoa den batura bat lortu arte. Batura berdina lortzen bada, zenbakia erdigunea da eta handiagoa den batura bat lortzen bada (berdina dena lortzeke) ez da erdigunea, beste edozein batura ere handiagoa izango baita.

### 4.4.3 Enuntziatua:

Erabiltzaileari **erdigunea** den lehenengo zenbakia aurkeztu eta hurrengoa nahi al duen galdetzen dion programa idatzi. Erabiltzaileak baietz ('b') erantzuten badu, programak erdigunea den hurrengo zenbakia kalkulatu eta aurkeztu eta hurrengoa nahi al duen galdetu beharko dio berriro ere erabiltzaileari. Prozesua erabiltzaielak ezetz ('e') erantzun arte errepikatu beharko da. 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.

1 baino handiagoa edo berdina den n **zenbaki** oso bat **erdigunea** dela esaten da bere aurreko denen batura (1 + 2 + 3 + … + n – 1) bere jarraian datozen zenbaki batzuk batuz lor badaiteke:

Erdiguneak diren eta ez diren zenbakien adibideak:

- 6 erdigunea da 1 + 2 + 3 + 4 + 5 = 15 delako eta 7 + 8 = 15 delako. Beraz, 6ren aurreko denen batura 6ren atzetik jarraian datozen zenbaki batzuk batuz lor daiteke.
- 35 erdigunea da 1 + 2 +...+ 34 = 595 delako eta 36 + 37 +…+ 49 = 595 delako. Ikus daitekeen bezala, 35en aurreko denen batura 35en atzetik jarraian datozen zenbaki batzuk batuz lor daiteke.

- 7 ez da erdigunea 1 + 2 + 3 + 4 + 5 + 6 = 21 delako eta bere atzetik jarraian dauden zenbakiak batuz ezin delako 21 balioa lortu: 8 ≠ 21, 8 + 9 ≠ 21, 8 + 9 + 10 ≠ 21, eta abar. Gainera 8 + 9 + 10 batura 21 baino handiagoa dela ikusiz badakigu zenbaki gehiago batuz ez dela inoiz 21 balioa lortuko.

Jarraian aipatzen diren funtzioak definitu eta erabili behar dira:

- *erdigunea_al_da*: ≥ 1 den x zenbaki oso bat emanda, erdigunea al den ala ez erabakitzen duen funtzioa. Honela, x erdigunea bada 1 balioa itzuliko du eta bestela 0 itzuliko du.

- *beste_erdigunerik_nahi*: erabiltzaileari erdigunea den beste zenbakirik 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

```
Erdigunea den lehenengo zenbakia: 6
Hurrengo erdigunea nahi al duzu?(b/e): q
Sartutako datua ez da egokia.
Hurrengo erdigunea nahi al duzu?(b/e): b
Hurrengo erdigunea: 35
Hurrengo erdigunea nahi al duzu?(b/e): e
Sakatu tekla bat amaitzeko.
```

## 4.5 5. ariketa

### 4.5.1 Helburua:

5. ariketako helburua funtzio berri bat eta lehenago erabili den beste funtzio bat erabiltzea da. Honela azpiprogramak edo azpikalkuluak burutzen dituzten funtzioak definituz programak egituratzeaz gain funtzio horiek ariketa desberdinetan erabil daitezkeela ikus daiteke.

### 4.5.2 5. enuntziaturako laguntza:

- Fibonacciren segidako lehenengo n elementuak lortzeko, main funtzioan 0tik hasita zenbakiak banan-banan pasatuz joan beharko da eta, *kalkulatu_fibonaccia* funtzioari deituz, zenbaki bakoitzaren Fibonaccia kalkulatu beharko da.

- kalkulatu_fibonaccia funtzioa definitzerakoan, zenbaki baten Fibonaccia kalkulatzeko aurreko bi zenbakien Fibonacciak behar direnez, x-en Fibonaccia kalkulatzeko (oinarrizkoak diren) 0 eta 1en Fibonaccietatik abiatu eta zenbaki denen Fibonacciak (2, 3, 4, …) kalkulatuz joan beharko da x-en Fibonaccira iritsi arte. Beraz, azkenean funtzio honek x-en Fibonaccia bakarrik itzuliko badu ere, bere aurreko denen Fibonacciak kalkulatu behar izango dira.

### 4.5.3 Enuntziatua:

Erabiltzaileari ≥ 1 den n zenbaki oso bat eskatu eta **Fibonacciren segida**ko lehenengo n elementuak aurkezten dituen programa idatzi. Fibonacciren segidako lehenengo elementua 0ren Fibonaccia denez, aurkeztu beharreko azkeneko Fibonaccia n – 1 zenbakiarena izango

da. Hasieran n balioa eskatzerakoan eskatze-prozesua aipatutako baldintza betetzen duen zenbaki bat lortu arte errepikatu beharko da.

≥ 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:

- *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.

- *kalkulatu_fibonaccia*: argumentu bezala 0 baino handiagoa edo berdina den x zenbaki oso bat emanda, x zenbaki horren fibonaccia kalkulatzen duen funtzioa.

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

Pantaila

```
>= 1 den zenbaki oso bat sartu: -4
Sartutako datua ez da egokia.
>= 1 den zenbaki oso bat sartu: 4
Fibonaccia(0) = 0
Fibonaccia(1) = 1
Fibonaccia(2) = 1
Fibonaccia(3) = 2
Sakatu tekla bat amaitzeko.
```