5. GAIA

FUNTZIOAK

1 PROGRAMAK FUNTZIOAK ERABILIZ

Orain arteko programetan funtzio bakarra idatzi dugu: *main* funtzioa edo funtzio nagusia. Eta programak egin beharreko dena funtzio horretan zehaztu dugu. Hala ere normalean problema bat ebazteko problema azpiproblematan (problema txikiagotan) zatitzea eta azpiproblema horiek bakoitza bere aldetik ebaztea errazagoa izaten da.
Gai honetan problema bat azpiproblematan, hau da, problema txikiagotan zatitzen ikasiko dugu. Problema bat azpiproblematan zatitu ondoren, azpiproblema horiek ebazten dituzten funtzioak definituko ditugu, azpiproblema bakoitzeko funtzio bat definituz.

1.1 Funtzioak erabiltzearen abantailak

- Programa batean azpiproblema bat behin baino gehiagotan ebatzi behar denean, azpiproblema hori ebazten duen funtzioa behin definitu eta gero behar den adina aldiz erabil dezakegu.

- Programak ulerterrazagoak dira. Esate baterako, main funtzioan azpiproblema bakoitza ebazteko behar diren agindu denak ipini beharrean, azpiproblema bakoitza ebazten duen funtzioa erabiltzen bada, ia lengoaia naturalean idatzita bezala geldituko zaigu main funtzioa.

- Programetan aldaketak egitea errazagoa da. Askotan aldaketek funtzio bati edo gutxi batzuei eragingo die eta eragina jasango duten funtzioez bakarrik arduratu beharko dugu.

- Problema desberdinetan azpiproblema bera agertzen bazaigu, azpiproblema hori ebazten duen funtzioa programa horietan denetan erabili ahal izango dugu.

1.2 Programen itxura berria

Orain arte idatzi ditugun programetan funtzio bakarra, main funtzioa, idatzi dugu. Programa horietan ordenagailuak main funtzioan idatzitako aginduak exekutatuko ditu.
Hemendik aurrera idatziko ditugun programetan funtzio bat baino gehiago idatziko dugu. Main funtzioa beti azalduko da eta gainera beste funtzio batzuk ere idatziko ditugu. Beraz programa funtzio-multzo bat bezala ikus edo uler dezakegu.

1.3 Programen exekuzioa

Esan bezala, hemendik aurrera idatziko ditugun programetan funtzio desberdinak edukiko ditugu eta horren aurrean honako galdera hau sortzen da:
Ordenagailuak nola exekutatzen du funtzio bat baino gehiago dituen programa?
Ordenagailuak main funtzioa bakarrik exekutatzen du, hau da, main funtzioan ipinitakoa jarraitzen du. Main funtzioan beste funtzio bati deitzen bazaio, orduan beste funtzio hori exekutatuko du, baina funtzioari deitzerakoan parentesi artean ipinitako datuekin.

1.4 Funtzioen inguruko kontzeptuak

Funtzioak maneiatzerakoan hiru kontzeptu bereiztu behar dira:

- Funtzioaren **prototipoa**: funtzioen prototipoak main funtzioaren aurretik ipintzen dira, include-en eta konstanteen ondoren.
- Funtzioen **definizioa**: funtzioen definizioak main bukatu ondoren ipintzen dira.
- Funtzioen **erabilera**: funtzio bat main-en barruan edo beste edozein funtzioren barruan erabil daiteke.

Posible da main funtzioaren aurrean beste funtzioak zuzenean definitzea ere. Kasu horretan prototipoak ez dira ipini beharko eta main-en ondoren funtzioak berriz definitu beharrik ere ez dago (lehenago definitu baitira).

1.5 Funtzioen eskema orokorra

Funtzio bat definitzerakoan lau gauza hartu behar dira kontuan:

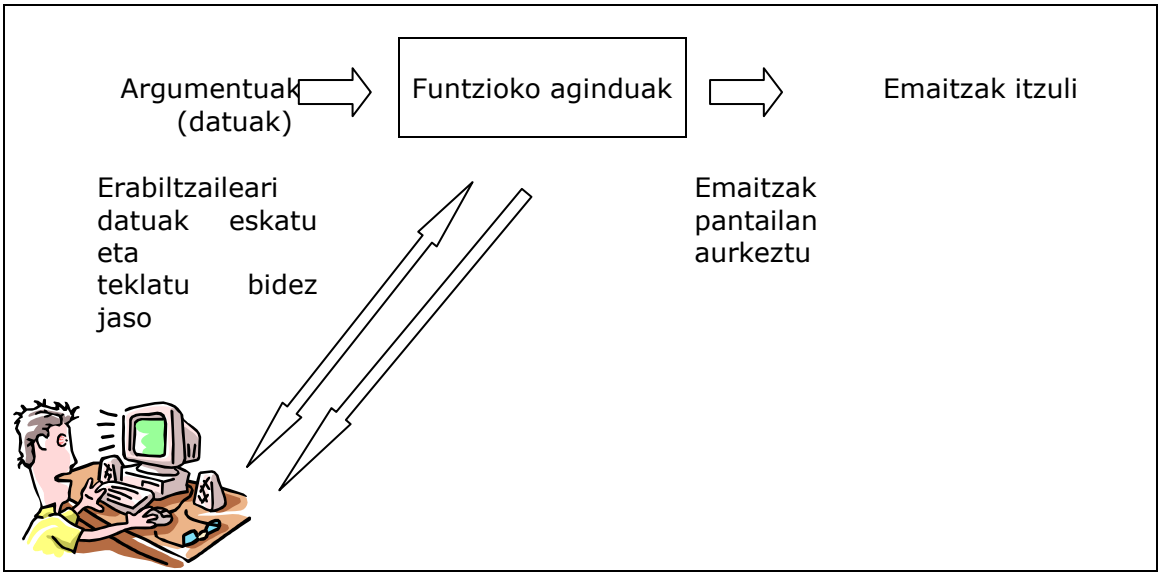
- Funtzioaren argumentuak (edo **datuak**).
- Funtzioak itzultzen dituen **emaitzak**.
- Funtzioak erabiltzaileari **ezer tekleatzeko eskatzen al dion** ala ez.
- Funtzioak **pantailan ezer aurkezten al duen** ala ez.

Hemendik aurrera "parametro formal" eta "benetako parametro" kontzeptuak erabili beharrean "argumentu" kontzeptua erabiliko da normalean, beste kontzeptu bien orokorpen bezala:

- Parametro formalak funtzioaren prototipoan eta definizioan funtzioaren ondoren parentesi artean ipintzen diren izenak dira. Datuak edo emaitzak izan daitezke.

- Benetako parametroak (edo parametro errealak) funtzioari deitzerakoan parametro formalen ordez ipintzen diren balioak (aldagaiak, konstanteak, espresioak, …) dira.

Hona hemen funtzioen eskema orokorra. Bertan funtzioak jaso ditzakeen datuak, itzul ditzakeen emaitzak eta erabiltzailearekin izan dezakeen harremana ageri dira:



1.6 Aurredefinitutako funtzioak eta geuk definitutako funtzioak

Dagoeneko funtzio batzuk erabiltzen badakigu: printf, scanf, system, getch, …
Funtzio horiek C lengoaian aurredefinituta daude eta guk ez daukagu definitu beharrik, zuzenean erabili ditzakegu. Baina erabili ahal izateko zein fitxategitan dauden zehaztu beharko dugu. Fitxategi horiek **h luzapena** izan ohi dute eta **include** aginduaren bidez aipatu beharko dira. (stdio.h, conio.h, …).
Aurredefinitutako funtzio asko daude C lengoaian baina jarraian guk erabiliko ditugunak aipatuko dira, funtzio bakoitzaren kasuan h luzapena duen zein fitxategitan dagoen adieraziz:

1.6.1 Funtzio matematikoak

Funtzio matematiko denak math.h fitxategian daude eta ondorioz, funtzio hauetakoren bat erabili nahi badugu, programaren hasieran beste include-ez gain **#include < math.h >** ipini beharko dugu.
Guretzat garrantzitsuenak fabs, sqrt eta pow dira:

- **fabs**
  fabs(x): x-en balio absolutua itzuliko du, hau da, |x| kalkulatuko du.

  Adibidea:

```
int zenb = 372, kop = -40, a, b;
a = fabs(zenb);   /* a-ri 372 esleituko zaio */
b = fabs(kop);    /* b-ri 40 esleituko zaio */
```

- **sqrt**
  sqrt(x): x-en erro karratua itzuliko du, hau da, $\sqrt{x}$ kalkulatuko du.

  x negatiboa bada, errorea gertatuko da, beraz geuk arduratu beharko dugu x positiboa dela ziurtatzeaz.

  Gogoratu erro karratuaren emaitza zenbaki erreal bat izango dela eta ondorioz, emaitza hori float edo double motako aldagai batean gordetzea komeniko zaigu. Erro karratuaren emaitza int edo long motako aldagai batean gordetzen bada, zati osoa bakarrik gordeko da.

  Adibidea:

```
int a = 49, b, c;
float r = 23, s;
b = sqrt(a);   /* b-ri 7 esleituko zaio */
c = sqrt(r);   /* c-ri 4 esleituko zaio */
s = sqrt(r);   /* s-ri 4.795832 esleituko zaio */
```

  Adibidea:

  $\sqrt{\sqrt{\frac{341}{4}+10}}$ espresioa C lengoaian sqrt(sqrt(341 / 4) + 10) bezala adieraziko litzateke.

  Adibidea:

```
float r = 23, s, t;
s = sqrt((r * 3) + 5);   /* s-ri 8.602325 esleituko zaio */
```

```
t = sqrt(sqrt(341 / 4) + 10);   /* t-ri 4.384010 esleituko zaio */
```

Adibidea:

```
float s, t;
s = sqrt(-200);   /* Errorea */
t = sqrt(sqrt(30 - 40);   /* Errorea */
```

Bi esleipen horiek errorea sortuko lukete zenbaki negatiboen erro karratua ezin baita kalkulatu.

Adibidea:
Demagun s aldagaiari r aldagaiaren balio absolutuaren erro karratua esleitu nahi diogula:

$$\sqrt{|r|}$$

```
float r = -200, s;
s = sqrt(fabs(-200));   /* s-ri 14.142136 balioa esleituko zaio */
```

Adibidea:
Demagun orain s aldagaiari r aldagaiaren erro karratua esleitu nahi diogula r positiboa baldin bada, eta bestela -1 esleitu nahi diogula:

r $\geq$ 0 baldin bada, s-ri $\sqrt{r}$ esleitu eta bestela s-ri -1 esleitu.

```
float r, s;

if (r >= 0)
 {
   s = sqrt(r);
 }
else
 {
   s = -1;
 }
```

- **pow**
  pow(x, y): x ber y itzuliko du, hau da, x^y kalkulatzeko balio du.

  Kasu partikular bezala, $\sqrt[n]{x^p}$ erako espresioak kalkulatzeko ere balio du, izan ere $\sqrt[n]{x^p} = x^{p/n}$ da.

  Adibidea:
  Demagun honako espresio hauek kalkulatu nahi ditugula: 5^7, 23^{-7}, $(12*3)^{2-3}$, $\sqrt[4]{20.07}$, $\sqrt[6]{4^3}$.

  C lengoaian ondoren adierazten den erara idatzi beharko genituzke:

```
long a;
      double r, s, t, v;
      a = pow(5, 7);
      r = pow(23, -7);
      s = pow((12 * 3), (2 - 3));
      t = pow(20.07, (1.0 / 4));
      v = pow(4, (3.0 / 6));
```

Adibide honetan long eta double motako aldagaiak definitu dira berredurarekin int eta float-en mugak erraz gainditzen direlako normalean.

Ondoren aipatzen diren kasuetan pow(x, y) funtzioak errorea emango luke:

- ✓ x = 0 eta y ≤ 0 betetzen denean.
- ✓ x negatiboa izanda, y osoa ez denean.

Adibidea:
Demagun honako formula hau kalkulatzen duen programa idatzi nahi dugula:

$$\sqrt[7]{\frac{(((2*x)/y)-y^4)^2}{(|x+y|)^5}}+x$$

Formula hori honako kasu hauetan ezin da kalkulatu:

- ✓ y zero denean.
- ✓ x + y zero denean.

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

/* Datuak: Programa honek erabiltzaileari bi zenbaki erreal eskatuko dizkio.
*/

/* Emaitzak: Ahal bada, aipatutako formula kalkulatu eta emaitza
             aurkeztuko da eta bestela mezu bat aurkeztuko da. */

/* Aldagaiak:
          - x, y: erabiltzaileak tekleatuko dituen bi zenbakiak
                  gordetzeko erabiliko dira.
          - emaitza: formula kalkulagarria baldin bada, hemen gordeko da
                     emaitza. */

void main()
{
  float x, y, emaitza;

  printf("\nSartu bi zenbaki erreal komaz bereiztura: ");
  printf("%f, %f", &x, &y);

  if ((y == 0)
   {
     printf("\nFormula ezin da kalkulatu y-ren balioa 0 delako.");
   }
  else if (x + y == 0)
   {
     printf("\nFormula ezin da kalkulatu x + y-ren balioa 0 delako.");
   }
  else
   {
     emaitza = pow(pow(((2 * x) / y) - pow(y, 4), 2) / pow(fabs(x + y), 5),
               (1.0 / 7)) + x;
     printf("\nFormularen balioa %f da.", emaitza);
   }

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

Hiru funtzio horietaz aparte (fabs, sqrt eta pow) beste funtzio matematiko batzuk ere badaude. Logaritmo mota desberdinak, sinua, kosinua eta abar kalkulatzeko balio duten funtzioak adibidez, baina hauek ez ditugu erabiliko.

- exp(x): e^x kalkulatzeko balio du.
- log(x): x-en logaritmo nepertarra, hau da, *ln x* kalkulatzeko balio du.
- log10(x): 10 oinarriko x-en logaritmoa kalkulatzeko balio du.
- sin(x), cos(x), tan(x): sinua, kosinua eta tangentea kalkulatzeko.
- asin(x), acos(x), atan(x): arku sinua, arku kosinua eta arku tangentea kalkulatzeko.
- sinh(x), cosh(x), tanh(x): sinu, kosinu eta tangente hiperbolikoa kalkulatzeko.
- …

1.6.2 Karaktereekin erlazionatutako funtzioak

Atal honetan karaktereekin erlazionatutako aurredefinitutako funtzio batzuk aipatuko dira. Funtzio hauek erabiltzeko **#include < ctype.h >** ipini beharko dugu programaren hasieran.

- **isupper**
  isupper(x): char motakoa den x balioa maiuskula bada, 1 itzuliko du eta bestela 0.

Adibidea:

```
int ma;
char c;
c = 'A';
ma = isupper(c); /* ma aldagaian 1 balioa gordeko da. */
ma = isupper(c + 1); /* ma aldagaian 1 balioa gordeko da. */
ma = isupper('r'); /* ma aldagaian 0 balioa gordeko da. */
ma = isupper('&'); /* ma aldagaian 0 balioa gordeko da. */
```

- **islower**
  islower(x): char motakoa den x balioa minuskula bada, 1 itzuliko du eta bestela 0.

- **isalpha**
  isalpha(x): char motakoa den x balioa letra bat bada, 1 itzuliko du eta bestela 0.

- **isdigit**
  isdigit(x): char motakoa den x balioa digito bat bada, 1 itzuliko du eta bestela 0.

- **toupper**
  toupper(x): char motakoa den x balioari dagokion maiuskula itzuliko du. Argumentu bezala emandako x balioa maiuskula baldin bada edo letra ez bada, x bera itzultzen du.

Adibidea:

```
char kar1, kar2;
kar1 = 'a';
kar2 = toupper(kar1); /* kar2 aldagaian 'A' balioa gordeko da. */
kar2 = toupper(kar1 + 1); /* kar2 aldagaian 'B' balioa gordeko da. */
```

```
kar2 = toupper('R'); /* kar2 aldagaian 'R' balioa gordeko da. */
kar2 = toupper('&'); /* kar2 aldagaian '&' balioa gordeko da. */
```

- **tolower**
  tolower(x): char motakoa den x balioari dagokion minuskula itzultzen du. Argumentu bezala emandako x balioa minuskula baldin bada edo letra ez bada, x bera itzuliko du.

1.6.3 Karaktere-kateekin erlazionatutako funtzioak

Jarraian karaktere-kateekin erlazionatutako aurredefinitutako funtzio batzuk aipatuko dira. Funtzio hauek erabiltzeko **#include < string.h >** ipini beharko dugu programaren hasieran.

- **strlen**
  strlen(kat): kat karaktere-kateak zenbat karaktere dituen itzuliko du.

- **strcmp**
  strcmp(kat1, kat2): kat1 eta kat2 karakterezko kateak berdinak badira, 0 itzuliko du eta bestela 1.

1.6.4 Geuk definitutako funtzioak

Aurredefinitutako funtzioak erabiltzeaz gain geuk funtzio berriak defini eta erabil ditzakegu nahi izanez gero. Gai honetan funtzio berriak nola definitu eta nola erabili azalduko da.
Funtzio berriak definitzerakoan bi aukera ditugu:

- Funtzioak h luzapena duen fitxategi batean gorde eta programaren hasieran include baten bidez fitxategi hori aipatu.
- Funtzioak zuzenean programan bertan idatzi. Kasu honetan bi erarata egin daiteke hori:
  - Main funtzioaren aurretik beste funtzioen prototipoak ipini eta main bukatutakoan beste funtzioen definizioak eman, edo
  - Zuzenean main funtzioaren aurretik beste funtzioen definizioak ipini. Kasu honetan ez da prototiporik ipini beharko.

1.7 Zer egin dezake funtzio batek?

Funtzioak definituz programa azpiprogramatan zatitzea lortzen da eta eginkizun desberdinak aparte lantzea.
Funtzio batek honako eginkizunak buru ditzake:

- Argumentuak erabiliz kalkuluak burutu eta lortutako emaitza edo emaitzak itzuli beste nonbaiten erabil daitezen.
- Erabiltzaileari zerbait tekleatzeko eskatu, erabiltzaileak tekleatutakoa jaso eta itzuli, beste nonbaiten erabilia izan dadin.
- Mezuak eta emaitzak aurkeztu.
- Argumentuak erabiliz kalkuluak burutu eta emaitzak aurkeztu.
- Erabiltzaileari zerbait tekleatzeko eskatu, erabiltzaileak tekleatutakoa jaso eta jasotako datu horiekin kalkuluak burutu eta emaitzak aurkeztu.
- Funtzio batek beste funtzioei ere dei diezaieke azpikalkuluak burutzeko.

1.8 RETURN agindua

Aurreko puntuan aipatu den bezala funtzioek era desberdinetako eginkizunak buru ditzakete. Funtzio baten helburua emaitza bat kalkulatu eta itzultzea denean (beste

nonbaiten erabiltzeko), **return** aginduaren bidez zehaztu beharko da itzuli beharreko balioa zein den. Return aginduaren bidez zenbakizko eta karakterezko balio sinpleak bakarrik itzul daitezke, hau da, zenbaki bat edo karaktere bat. Ezingo dira bi zenbaki edo gehiago edo bi karaktere edo gehiago itzuli, bat bakarrik itzuli ahal izango da.
Ondorengo ataletan adibideak erabiliz era desberdinetako eginkizunak burutzen dituzten funtzioak aurkeztuko dira. Era desberdinetako funtzioen ezaugarriak nahasteko orduan ez dago inolako mugarik. Emango diren adibideek ohikoenak diren ezaugarriak dituzte.

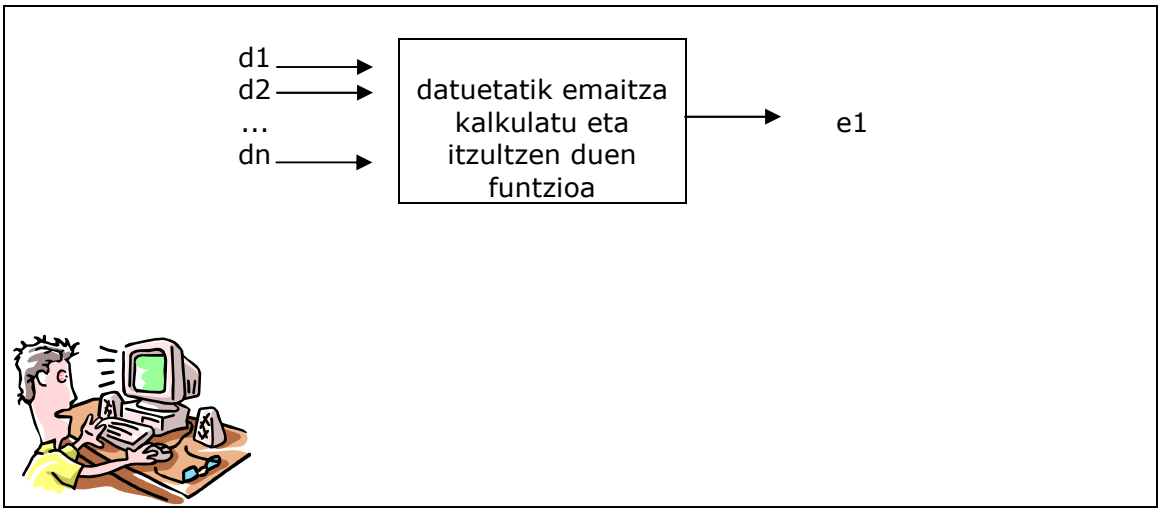
2 ARGUMENTUAK ERABILIZ EMAITZA BAKARRA KALKULATZEN DUTEN FUNTZIOAK

2.1 Ezaugarriak eta definizioa

Era honetako funtzioek argumentu bezala datu bat edo datu batzuk hartu, eragiketa edo kalkulu batzuk burutu eta emaitza bat itzuliko dute.
Funtzioak itzulitako emaitzarekin gauza desberdinak egin daitezke:

- Aldagai batean gorde.
- Espresio batean erabili.

Funtzio hauek ingurunearekin duten harremana grafikoki honela azal dezakegu:



d1, d2, ..., dn datu bezala pasatutako argumentuak dira eta e1 funtzioak itzultzen duen emaitza.
Grafikoan ikus daitekeen bezala, era honetako funtzioek erabiltzaileari ez diote ezer tekleatzeko eskatzen eta ez dute pantailan ezer aurkezten.
Dagoeneko ezagutzen ditugun funtzio matematiko denak (fabs, sqrt, pow, sin, ...) era honetako funtzioak dira: datu bat edo gehiago hartu eta emaitza bat itzultzen dute.
Jarraian mota honetako funtzioen prototipoen, definizioen eta erabiltzeko eren ezaugarri orokorrak azalduko dira:

- **Prototipoa**: prototipoak honako itxura hau izango du:

```
emaitzaren_mota funtzioaren_izena (datuak diren parametro formalen motak eta izenak);
```

- **Definizioa**: definizioak honako eskema hau jarraituko du:

```
emaitzaren_mota funtzioaren_izena (datuak diren parametro formalen motak eta izenak)
```

```
{
  aginduak
  return(emaitza);
}
```

return agindua funtzioak emaitza bezala zein balio itzuliko duen zehazteko erabiltzen da.

- **Erabilpena**: funtzioak itzulitakoa aldagai batean gorde daiteke edo espresio batean edo konparazio batean erabil daiteke.

  a) Funtzioak itzultzen duena aldagai bati esleitzeko honako hau egin beharko genuke:

```
aldagaia = funtzioaren_izena (argumentuen izenak);
```

Adibidea:

```
int n = 5, m;
m = pow (n + 1, 3); /* m-ri 216 esleituko zaio */
```

  b) Funtzioak itzultzen duena aldagai batean gorde beharrean espresio baten zati bezala ipintzeko aukera ere badugu:

```
funtzioaren_izena (argumentuen izenak);
```

Adibideak:

```
int n = 5, m;
m = n + pow (n + 1, 3); /* m-ri 223 esleituko zaio */
```

```
int n = 5, m;
if (pow (n + 1, 3) < 300)
 {
   m = 0;
 }
```

2.2 Adibideak

2.2.1 Faktoriala kalkulatzen duen funtzioa

Jarraian zenbaki baten faktoriala kalkulatzeko balio duen funtzioa idatziko da.
Funtzio honek 0 baino handiagoa edo berdina den x zenbaki oso baten faktoriala kalkulatzeko balioko du. Funtzioaren datua (edo parametro formala) x izango da, funtzioak long motako emaitza itzuliko du eta funtzioaren izena "kalkulatu_faktoriala" izango da:

- **Prototipoa**:

```
long kalkulatu_faktoriala (int x);
```

Emaitzaren mota — funtzioaren izena — Datuaren mota eta izena

- **Definizioa**:

  kalkulatu_faktoriala izeneko funtzioaren definizioan ≥ 0 den edozein zenbaki izan daitekeen x-en faktoriala kalkulatzeko burutu beharrekoa zehaztu behar da.

  x edozein zenbaki positibo (≥ 0) izanda, bere faktoriala honela kalkulatuko litzateke:

  - x-en balioa 0 bada, bere faktoriala 1 da.
  - x > 0 bada, 1etik x-era bitarteko zenbakiak banan-banan pasatu eta biderkaketa burutuz lortzen den zenbakia izango da emaitza.

```
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 */
```

return agindua funtzioak emaitza bezala zein balio itzuliko duen zehazteko erabiltzen da.

- **Erabilpena**:

  Jarraian oraintxe definitu dugun *kalkulatu_faktoriala* izeneko funtzioaren erabilpen-adibide batzuk datoz.
  Funtzioak itzulitako emaitzarekin gauza desberdinak egin daitezke:

- Aldagai batean gorde.
- Espresio batean erabili.

  Adibidea:
  Demagun 7 zenbakiaren faktoriala long motako n aldagaian gorde nahi dugula eta m aldagaiak duen balioaren faktoriala p aldagaian gorde nahi dugula:

```
long n, p;
int m = 12;

n = kalkulatu_faktoriala (7);
p = kalkulatu_faktoriala (m);
```

Adibidea:
Demagun 5en faktoriala m baino handiagoa bada, n-ri (m + 2)-ren faktoriala esleitu nahi diogula eta bestela -1 esleitu nahi diogula:

```
long n;
int m = 9;

if (kalkulatu_faktoriala(5) > m)
 {
   n = kalkulatu_faktoriala (m + 2);
 }
else
 {
   n = -1;
 }
```

Adibidea:
Orain demagun (m + 3)-ren faktorialaren faktoriala esleitu nahi diogula n aldagaiari:

```
long n;
int m = 6;

n = kalkulatu_faktoriala (kalkulatu_faktoriala (m + 3));
```

2.2.2 Zenbaki bat lehena al den ala ez erabakitzen duen funtzioa

Jarraian 1 baino handiagoa edo berdina den x zenbaki oso bat lehena al den ala ez erabakitzen duen funtzioa idatziko da:

- **Prototipoa**:

```
int lehena_al_da (int x);
```

Emaitzaren mota — Funtzioaren izena — Datuaren mota eta izena

- **Definizioa**:
lehena_al_da izeneko funtzioaren definizioan edozein zenbaki positibo izan daitekeen x balioa lehena al den ala ez erabakitzeko burutu beharrekoa zehaztu behar da.
x edozein zenbaki oso eta positibo (≥ 1) izanda, lehena al den ala ez erabakitzeko, bere zatitzaileak kontatu behar dira. Bi zatitzaile baditu, orduan lehena da, eta bi zatitzaile baino gehiago edo gutxiago baditu, orduan ez da lehena. Funtzioaren emaitza beti 0 edo 1 izango da. Emandako x zenbakia lehena ez dela adierazteko 0 itzuliko du eta lehena dela adierazteko 1 itzuliko du:

```
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 */
```

- **Erabilpena**:

  Jarraian oraintxe definitu dugun *lehena_al_da* izeneko funtzioaren erabilpen-adibide batzuk datoz.

  Adibidea:
  Demagun 7 zenbakia eta n aldagaian dagoen zenbakia lehenak al diren ala ez erabaki eta erantzunak er1 eta er2 izeneko aldagaietan gorde nahi ditugula. Honako hau idatzi beharko genuke:

```
int n = 20, er1, er2;

er1 = lehena_al_da (7);
er2 = lehena_al_da (n);
```

  Adibidea:
  Demagun (n + 3) lehena bada, m aldagaian (n * 2) eta bestela (n / 2) gordetzea nahi dugula. Honako hau idatz dezakegu:

```
int n = 23, m;

if (lehena_al_da (n + 3) == 1)
 {
   m = n * 2;
 }
else
 {
   m = n / 2;
 }
```

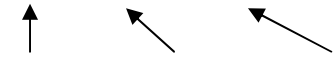
2.2.3 x eta y, bi zenbaki oso eta positibo (≥ 1) emanda, x^y kalkulatzen duen funtzioa

Jarraian bi zenbaki oso eta positiboren arteko berredura kalkulatzeko balio duen funtzioa idatziko da. Era honetako berredura bat kalkulatzeko aurredefinitutako pow funtzioa erabil daiteke, baina hemen geuk definituko dugu. Gure funtzioak

zenbaki oso eta positiboentzat bakarrik balioko du eta beraz ez da pow bezain orokorra izango, pow-k zenbaki errealentzat ere balio baitu.
Funtzioaren argumentuak (parametro formalak) x eta y izango dira eta funtzioak emaitza bat itzuliko du, x^y:

- **Prototipoa**:

```
int nire_pow (int x, int y);
```



Emaitzaren mota | Funtzioaren izena | Datuen motak eta izenak

- **Definizioa**:

nire_pow izeneko funtzioaren definizioan edozein zenbaki positibo (≥ 1) eta oso izan daitezkeen x eta y-ren arteko berredura kalkulatzeko burutu beharrekoa zehaztu behar da: x^y kalkulatzeko x balioa y aldiz biderkatu beharko da:

```
int nire_pow (int x, int y)

 /* Funtzio honek argumentu bezala >= 1 diren x eta y zenbaki osoak
    emanda, x ber y itzuliko du. */

 { /* Funtzioaren hasierako giltza */
   int ber, kont;

   /* kont aldagaia x zenbat aldiz biderkatu den kontatzeko
      erabiliko da */

   ber = 1; /* berredura kalkulatuz joateko */

   for (kont = 0; kont < y; kont = kont + 1)
    {
      ber = ber * x;
    }

   return (ber);
} /* Funtzioaren bukaerako giltza */
```

- **Erabilpena**:

Jarraian oraintxe definitu dugun *nire_pow* izeneko funtzioaren erabilpen-adibide batzuk datoz.

Adibidea:
Demagun b aldagaian 7^n balioa gorde nahi dugula. Honako hau idatzi beharko genuke:

```
int n, b;

b = nire_pow (7, n);
```

Adibidea:
Demagun n^5 balioa m baino handiagoa bada, p aldagaian (n * 2) eta bestela (n / 2) gordetzea nahi dugula. Honako hau idatz dezakegu:

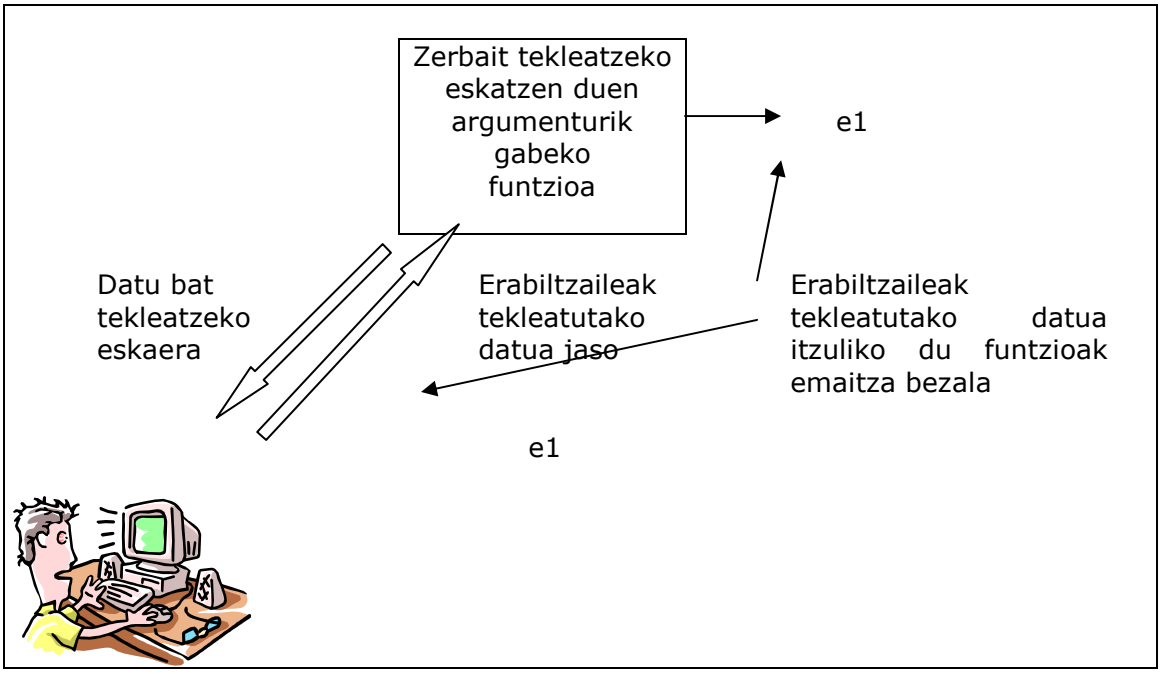
```
int m, n, p;

if (nire_pow (n, 5) > m)
 {
   p = n * 2;
 }
else
 {
   p = n / 2;
 }
```

3 ERABILTZAILEARI ZERBAIT TEKLEATZEKO ESKATU ETA TEKLEATUTAKOA JASO ETA ITZULTZEN DUTEN FUNTZIOAK

3.1 Ezaugarriak eta definizioa

Erabiltzaileari zerbait tekleatzeko eskatzen dioten funtzioen kasuan gerta daiteke argumenturik ez edukitzea edo edukitzea, baina beti balio bakarra eskatuko diote erabiltzaileari eta ondorioz emaitza bakarra itzuliko dute. Funtzio hauek erabiltzaileari zenbaki edo karaktere bat eskatuko diote eta erabiltzaileak zenbaki bat edo karaktere bat tekleatzen duenean, hori jaso eta itzuliko dute emaitza bezala (**return** agindua erabiliz). Adibideetan ikusiko den bezala, batzutan erabiltzaileak tekleatu beharrekoak baldintzaren bat bete beharko du eta baldintza hori betetzen duen balio bat tekleatu arte eskatze-prozesua errepikatzeaz ere funtzioa bera arduratuko da.
Funtzioak itzulitako zenbakia edo karakterea normalean aldagai batean gordeko dugu.

a) Argumenturik ez duten era honetako funtzioak ingurunearekin duten harremana grafikoki honela azal dezakegu:



e1 erabiltzaileak tekleatutakoa eta funtzioak itzuliko duen emaitza da.
Jarraian argumenturik ez duten era honetako funtzioen prototipoen, definizioen eta erabilpenen ezaugarri orokorrak azalduko dira:

- **Prototipoa**: argumenturik ez duen era honetako funtzio baten prototipoak honako itxura hau izango du:

```
emaitzaren_mota funtzioaren_izena (void);
```

Argumenturik ez dagoenez, parentesi artean void hitza ipini behar da.

- **Definizioa**: definizioak honako formato hau izango du:

```
emaitzaren_mota funtzioaren_izena (void)
  {
    aldagaiak
    aginduak
    return(emaitza);
  }
```

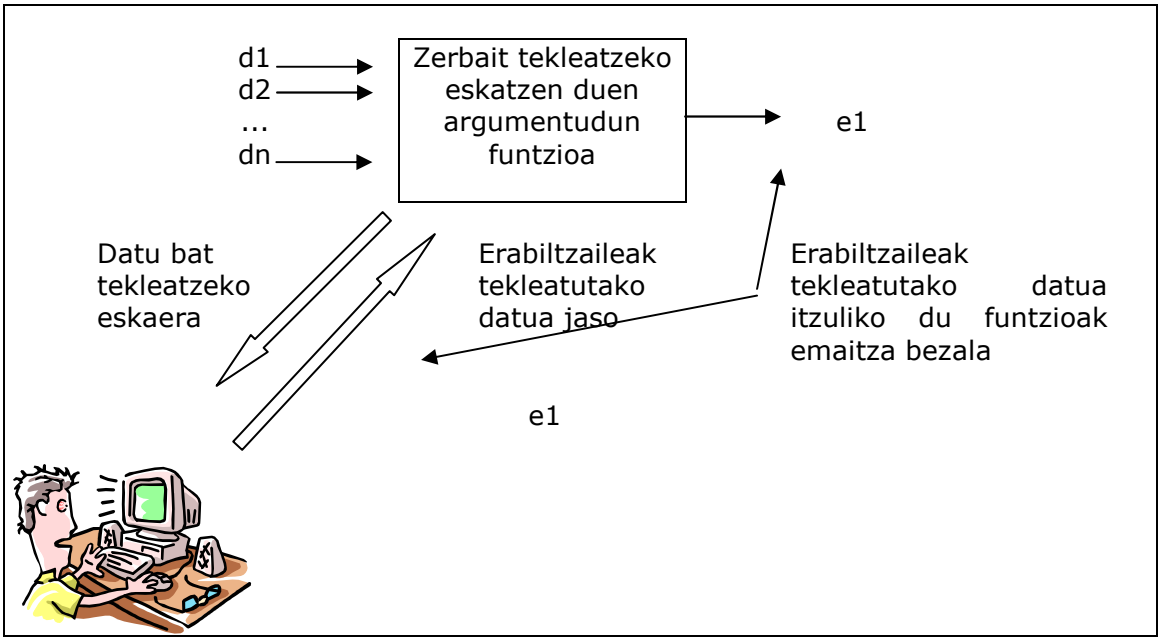
return agindua funtzioak emaitza bezala zein balio itzuliko duen zehazteko erabiltzen da.

- **Erabilpena**: era honetako funtzio bat main-en edo beste edozein funtziotan erabiltzerakoan normalean emaitza aldagai batean gordeko da:

```
aldagaia = funtzioaren_izena ( );
```

Funtzioa erabiltzerakoan parentesiak hutsik ipini behar dira daturik ez dagoelako.

b) Erabiltzaileari balio bat eskatzen dioten argumentudun funtzioek ingurunearekin duten harremana grafikoki honela azal dezakegu:



d1, d2, ..., dn datu bezala pasatutako argumentuak dira eta e1 funtzioak itzultzen duen emaitza.

Jarraian era honetako argumentudun funtzioen prototipoek, definizioek eta erabilpenek dituzten ezaugarri orokorrak azalduko dira:

- **Prototipoa**: prototipoak honako itxura hau izango du:

emaitzaren_mota *funtzioaren_izena* (datuak diren parametro formalen motak eta izenak);

- **Definizioa**: definizioak honako formato hau izango du:

```
emaitzaren_mota funtzioaren_izena (datuak diren parametro formalen
motak eta izenak)
  {
    aldagaiak
    aginduak
    return(emaitza);
  }
```

return agindua funtzioak emaitza bezala zein balio itzuliko duen zehazteko erabiltzen da.

- **Erabilpena**: era honetako funtzio bat main-en edo beste edozein funtziotan erabiltzerakoan, funtzioak itzuliko duen emaitza bakarra normalean aldagai batean gordeko da:

aldagaia = funtzioaren_izena (argumentuen izenak);

3.2 Adibideak

3.2.1 Zenbaki oso bat eskatzen duen funtzioa

Jarraian erabiltzaileari zenbaki oso bat tekleatzeko eskatuko dion funtzio bat idatziko dugu. Erabiltzaileak zenbaki oso bat tekleatutakoan, zenbaki hori jaso eta emaitza bezala itzuliko du funtzioak. Beraz, funtzioak ez du daturik, int motako emaitza itzuliko du eta funtzioaren izena "eskatu_zenbaki_oso_bat" izango da:

- **Prototipoa**:

```
int
eskatu_zenbaki_oso_bat(void);
```

Emaitzaren mota — funtzioaren izena — Daturik ez duela adierazteko void ipintzen da

- **Definizioa**:
eskatu_zenbaki_oso_bat izeneko funtzioaren definizioan zenbaki oso bat eskatu eta jasotzeko burutu beharrekoa zehaztu behar da:

```
int eskatu_zenbaki_oso_bat(void)

 /* Funtzio honek erabiltzaileari zenbaki oso bat eskatuko dio eta
    erabiltzaileak zenbaki bat tekleatutakoan, zenbaki hori itzuliko du
    emaitza bezala. */

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

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

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

- **Erabilpena**:
  Jarraian oraintxe definitu dugun *eskatu_zenbaki_oso_bat* izeneko funtzioaren erabilpen-adibide batzuk datoz.

  Adibidea:
  Demagun erabiltzaileari zenbaki oso bat eskatu nahi diogula. Hori egiteko printf eta scanf bat idatzi beharrean oraintxe definitu dugun *eskatu_zenbaki_oso_bat* funtzioa erabiltzea nahikoa izango da (adibide honetan zenbakia n aldagaian gordeko da):

```
int n;
n = eskatu_zenbaki_oso_bat();
```

  Adibidea:
  Demagun orain erabiltzaileari bi zenbaki oso tekleatzeko eskatu nahi diogula. Hori egiteko aukera bat printf eta scanf idaztea da, baina oraintxe definitu dugun funtzio hori erabiliz egitekotan, funtzioari bi aldiz deitu beharko genioke, funtzio horrek aldi bakoitzean zenbaki bakar bat lortzen baitu:

```
int n, m;

n = eskatu_zenbaki_oso_bat();
m = eskatu_zenbaki_oso_bat();
```

  Erabilpen-adibide hauetan ikusi den bezala, *eskatu_zenbaki_oso_bat* funtzioari deitzerakoan guk ez diogu daturik eman behar. Horregatik parentesiak hutsik daude.

3.2.2 Zero baino handiagoa den zenbaki oso bat eskatzen duen funtzioa

Jarraian erabiltzaileari ≥ 1 den zenbaki oso bat eskatzen dion eta argumenturik ez duen funtzio bat definituko dugu. Eskatze-prozesua erabiltzaileak ≥ 1 den zenbaki bat tekleatu arte errepikatuko du. Zenbaki egoki bat lortutakoan zenbaki hori itzuliko du emaitza bezala.
Beraz, funtzioak ez du daturik, int motako emaitza itzuliko du eta funtzioaren izena "eskatu_zenbaki_positiboa" izango da:

- **Prototipoa**:

```
int  eskatu_zenbaki_positiboa
(void);
```

  Emaitzaren mota — Funtzioaren izena — Daturik ez duela adierazteko void ipintzen da

- **Definizioa**:

  eskatu_zenbaki_positiboa izeneko funtzioaren definizioan zenbaki oso eta positibo bat eskatu eta jasotzeko burutu beharrekoa zehaztu behar da:

```
int eskatu_zenbaki_positiboa(void)
```

```
/* Funtzio honek erabiltzaileari zenbaki oso eta positibo bat (>= 1)
   eskatuko dio eta eskatze-prozesua erabiltzaileak zenbaki egoki bat
   tekleatutakoan amaituko da. Funtzioak azkeneko zenbaki egoki hori
   itzuliko du emaitza bezala. */

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

  do
  { printf("\nZenbaki oso eta positibo bat (>= 1) tekleatu: ");
    scanf("%d", &zenb);
    if (zenb <= 0)
     {
       printf("\nTekleatutako zenbakia ez da egokia.");
     }
  } while (zenb <= 0);

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

- **Erabilpena**:

  Jarraian oraintxe definitu dugun *eskatu_zenbaki_positiboa* izeneko funtzioaren erabilpen-adibide batzuk datoz.

  Adibidea:
  Demagun erabiltzaileari osoa den zenbaki positibo bat eskatu nahi diogula. Hori egiteko oraintxe definitu dugun *eskatu_zenbaki_positiboa* funtzioa erabiltzea nahikoa izango da (adibide honetan zenbakia n aldagaian gordeko da):

  ```
  int n;
  n = eskatu_zenbaki_positiboa();
  ```

  Adibidea:
  Demagun orain erabiltzaileari osoak diren bi zenbaki positibo tekleatzeko eskatu nahi diogula. Hori egiteko funtzioari bi aldiz deitu beharko genioke, funtzio horrek aldi bakoitzean zenbaki bakar bat lortzen baitu:

  ```
  int n, m;

  n = eskatu_zenbaki_positiboa();
  m = eskatu_zenbaki_positiboa();
  ```

  Erabilpen-adibide hauetan ikusi den bezala, *eskatu_zenbaki_positiboa* funtzioari deitzerakoan guk ez diogu daturik eman behar. Horregatik parentesiak hutsik daude.

3.2.3 z baino handiagoa den zenbaki oso bat eskatzen duen funtzioa

Argumentu edo datu bezala osoa den z zenbaki bat emanda, erabiltzaileari z baino handiagoa den zenbaki oso bat tekleatzeko eskatuko dion funtzioa definituko dugu jarraian. Zenbakia eskatzeko prozesua z baino handiagoa den zenbaki bat lortu arte errepikatuko da. Erabiltzaileak zenbaki egoki bat tekleatutakoan, zenbaki hori jaso eta emaitza bezala itzuliko du:

- **Prototipoa**:

```
int eskatu_zenbaki_handiagoa
            (int z);
```

Emaitzaren mota — Funtzioaren izena — Datuaren mota eta izena

- **Definizioa**:

eskatu_zenbaki_handiagoa izeneko funtzioaren definizioan datu bezala emandako zenbakia baino handiagoa den zenbaki oso bat eskatu eta jasotzeko burutu beharrekoa zehaztu behar da:

```
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 */
```

- **Erabilpena**:

Jarraian oraintxe definitu dugun *eskatu_zenbaki_handiagoa* izeneko funtzioaren erabilpen-adibide batzuk datoz.

Adibidea:

Demagun erabiltzaileari 5 baino handiagoa den zenbaki oso bat eskatu nahi diogula. Hori egiteko *eskatu_zenbaki_handiagoa* izeneko funtzioari datu bezala 5 emanez deitzea nahikoa da:

```
int n;
n = eskatu_zenbaki_handiagoa(5);
```

Adibidea:

Demagun orain erabiltzaileari ≥ 0 eta ≥ 1 diren bi zenbaki oso tekleatzeko eskatu nahi diogula eta zenbakiak m eta n aldagaietan gorde nahi ditugula. Hori egiteko aukera bat oraintxe definitu dugun *eskatu_zenbaki_handiagoa* izeneko funtzioa erabiltzea da. Baina funtzio hori erabiliz egitekotan, funtzioari bi aldiz deitu beharko genioke, funtzio horrek aldi bakoitzean zenbaki bakar bat lortzen baitu. Lehenengo deian datu bezala -1 eman

beharko diogu (-1 baino handiagoa den zenbaki bat, hau da, ≥ 0 den zenbaki bat lortzeko) eta bigarren deian datu bezala 0 eman beharko diogu (0 baino handiagoa den zenbaki bat, hau da, ≥ 1 den zenbaki bat lortzeko):

```
int m, n;

m = eskatu_zenbaki_handiagoa(-1);
n = eskatu_zenbaki_handiagoa(0);
```

Erabilpen-adibide hauetan ikusi den bezala, *eskatu_zenbaki_handiagoa* funtzioari deitzerakoan guk datu bat, zenbaki oso bat, eman beharko diogu edo parentesi artean ipini beharko diogu. Funtzioak guk emandako zenbakia baino handiagoa den zenbaki bat eskatuko dio erabiltzaileari eta zenbakia eskatzeko prozesua erabiltzaileak zenbaki egoki bat tekleatu arte errepikatuko du. Erabiltzaileak zenbaki egoki bat tekleatutakoan, funtzioak erabiltzaileak tekleatutako zenbaki egoki hori itzuliko du emaitza bezala.

3.2.4 [x, y] tarteko zenbaki oso bat eskatzen duen funtzioa

Argumentu edo datu bezala osoak diren x eta y (x ≤ y) bi zenbaki emanda, erabiltzaileari [x, y] tartekoa den zenbaki oso bat tekleatzeko eskatuko dion funtzioa definituko dugu jarraian. Funtzioari deitzerakoan bi zenbaki eman beharko dizkiogu datu bezala edo argumentu bezala. Zenbakia eskatzeko prozesua [x, y] tartekoa den zenbaki bat lortu arte errepikatuko da. Erabiltzaileak zenbaki egoki bat tekleatutakoan, zenbaki hori jaso eta emaitza bezala itzuliko du funtzio honek:

- **Prototipoa**:

```
int eskatu_tarteko_zenbakia (int
                x, int y);
```

Emaitzaren mota — Funtzioaren izena — Datuen motak eta izenak

- **Definizioa**:

eskatu_tarteko_zenbakia izeneko funtzioaren definizioan datu bezala emandako x eta y zenbakien artean dagoen zenbaki oso bat eskatu eta jasotzeko burutu beharrekoa zehaztu behar da:

```
int eskatu_tarteko_zenbakia (int x, int y)

 /* Funtzio honek erabiltzaileari [x, y] tartekoa den zenbaki oso bat
    eskatuko dio eta erabiltzaileak tarte horretakoa den zenbaki bat
    tekleatutakoan, zenbaki hori itzuliko du emaitza bezala. Zenbakia
    eskatzeko prozesua zenbaki egoki bat lortu arte errepikatuko da.
    Gainera x <= y dela suposatuko da. */

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

   do
    {
      printf("\n[%d, %d] tartekoa den zenbaki oso bat tekleatu: ",
             x, y);
      scanf("%d", &zenb);
```

```
      if ((zenb < x) || (zenb > y))
       {
         printf("\nTekleatutako zenbakia ez da egokia.");
       }
    } while ((zenb < x) || (zenb > y));

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

- **Erabilpena**:

  Jarraian oraintxe definitu dugun *eskatu_tarteko_zenbakia* izeneko funtzioaren erabilpen-adibide batzuk datoz.

  Adibidea:
  Demagun erabiltzaileari [3, 12] tarteko zenbaki oso bat eskatu nahi diogula. Hori egiteko *eskatu_tarteko_zenbakia* izeneko funtzioari datu bezala 3 eta 12 emanez deitzea nahikoa da:

```
int n;
n = eskatu_tarteko_zenbakia(3, 12);
```

  Adibidea:
  Demagun orain erabiltzaileari [0, 10] tarteko bi zenbaki oso tekleatzeko eskatu nahi diogula eta zenbakiak m eta n aldagaietan gorde nahi ditugula. Hori egiteko aukera bat oraintxe definitu dugun *eskatu_tarteko_zenbakia* izeneko funtzioa erabiltzea da. Baina funtzio hori erabiliz egitekotan, funtzioari bi aldiz deitu beharko genioke, funtzio horrek aldi bakoitzean zenbaki bakar bat lortzen baitu. Lehenengo deian datu bezala 0 eta 10 eman beharko dizkiogu eta bigarren deian ere datu bezala 0 eta 10 balioak eman beharko dizkiogu:

```
int m, n;

m = eskatu_tarteko_zenbakia(0, 10);
n = eskatu_tarteko_zenbakia(0, 10);
```

  Adibidea:
  Demagun orain erabiltzaileari [-20, 15] tarteko zenbaki oso bat eta [30, 60] tarteko zenbaki oso bat tekleatzeko eskatu nahi diogula eta zenbakiak m eta n aldagaietan gorde nahi ditugula. Hori egiteko aukera bat oraintxe definitu dugun *eskatu_tarteko_zenbakia* izeneko funtzioa erabiltzea da. Funtzio hori erabiliz egitekotan, funtzioari bi aldiz deitu beharko genioke, funtzio horrek aldi bakoitzean zenbaki bakar bat lortzen baitu. Lehenengo deian datu bezala -20 eta 15 eman beharko dizkiogu eta bigarren deian 30 eta 60 balioak eman beharko dizkiogu:

```
int m, n;

m = eskatu_tarteko_zenbakia(-20, 15);
n = eskatu_tarteko_zenbakia(30, 60);
```

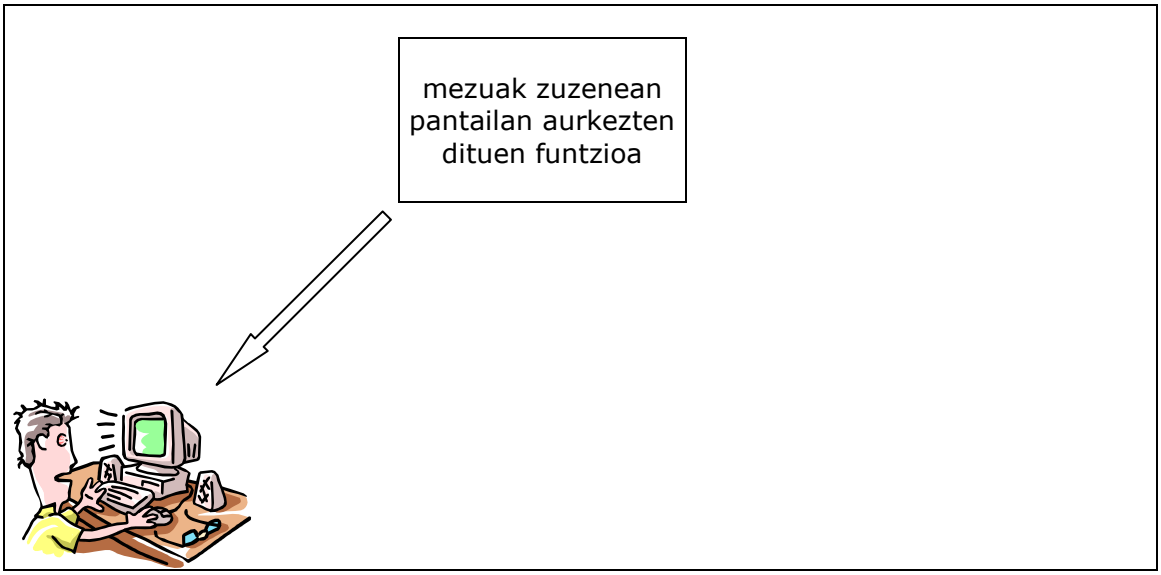
  Erabilpen-adibide hauetan ikusi den bezala, *eskatu_tarteko_zenbakia* funtzioari deitzerakoan guk bi datu, bi zenbaki oso, eman behar dizkiogu edo ipini behar dizkiogu parentesi artean. Funtzioak guk emandako zenbaki horiek mugatzen duten tartekoa den zenbaki bat eskatuko dio erabiltzaileari eta zenbakia eskatzeko prozesua erabiltzaileak zenbaki egoki bat tekleatu arte errepikatuko du. Erabiltzaileak zenbaki egoki bat tekleatutakoan, funtzioak erabiltzaileak tekleatutako zenbaki egoki hori itzuliko du emaitza bezala.

4 MEZUAK ETA EMANDAKO ARGUMENTUAK PANTAILAN AURKEZTEN DITUZTEN FUNTZIOAK

4.1 Ezaugarriak eta definizioa

Era honetako funtzioak ere argumentudunak edo argumenturik gabeak izan daitezke. Argumenturik ez badute, testu hutsezko mezu bat idazteko balioko dute eta argumentudunak badira, argumentuen balioak pantailan aurkezteko balio izan ohi dute.

a) Mezuak aurkezten dituzten argumenturik gabeko funtzioek ingurunearekin duten harremana grafikoki honela azal dezakegu:



- **Prototipoa**: prototipoak honako itxura hau izango du:

```
void funtzioaren_izena (void);
```

Funtzio hauek emaitzik ez dutenez itzultzen, emaitzaren motari dagokion lekuan void ipintzen da eta daturik ere ez dutenez, parentesi artean ere void ipintzen da.

- **Definizioa**: definizioak honako itxura hau izango du:

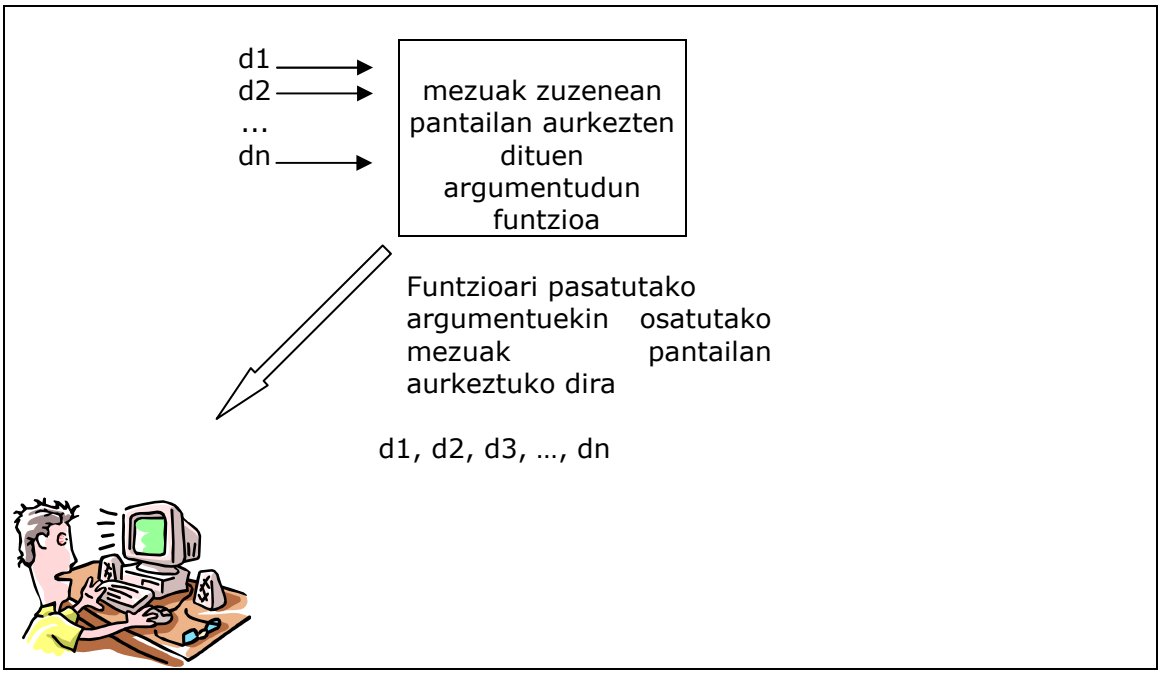
```
void funtzioaren_izena (void)
  {
    aldagaiak
    aginduak
  }
```

- **Erabilpena**: era honetako funtzio bat main-en edo beste edozein funtziotan erabiltzerakoan, funtzioa honela idatziko da:

```
funtzioaren_izena
();
```

Emaitzik ez dagoenez ezkerrean ez da aldagairik ipini behar, ez baitago emaitza gorde beharrik eta daturik ere ez dagoenez, parentesiak hutsik ipini behar dira.

b) Mezuak aurkezten dituzten argumentudun funtzioek ingurunearekin duten harremana grafikoki honela azal dezakegu:



d1, d2, ..., dn datu bezala pasatutako argumentuak dira.

Era honetako funtzioen prototipoen, definizioen eta erabilpenen ezaugarriak aipatuko dira jarraian:

- **Prototipoa**: prototipoak honako itxura hau izango du:

```
void funtzioaren_izena (datuak diren parametro formalen motak eta
```

  Funtzio hauek emaitzik ez dutenez itzultzen, emaitzaren motari dagokion lekuan void ipintzen da.

- **Definizioa**: definizioak honako formato hau izango du:

```
void funtzioaren_izena (datuak diren parametro formalen motak eta izenak)
  {
    aldagaiak
    aginduak
  }
```

- **Erabilpena**:

```
funtzioaren_izena (argumentuen
```

Emaitzik ez dagoenez ezkerrean ez da aldagairik ipini behar, ez baitago emaitza gorde beharrik.

4.2 Adibideak

4.2.1 Erabiltzaileak tekleatu duen zenbaki-segida hutsa izan dela esaten duen funtzioa

Zenbaki-segida bat eskatzerakoan zenbaki-segida hutsa (zenbakirik gabea) izan dela esaten duen mezua aurkezten duen funtzioa idatziko dugu jarraian.

Funtzio honek ez du daturik eta ez du emaitzik eta a) erakoa izango litzateke:

- **Prototipoa**:

  void segida_hutsa_da (void);

  Emaitzik ez duelako | Funtzioaren izena | Datuarik ez duelako

- **Definizioa**:

  segida_hutsa_da izeneko funtzioaren definizioan erabiltzaileak tekleatu duen zenbaki-segida hutsa izan dela esanez mezu bat aurkezteko egin beharrekoa zehaztu behar da:

```
void segida_hutsa_da (void)

 /* Funtzio honek erabiltzaileak tekleatu duen zenbaki-segida hutsa izan
    dela adieraziz mezu bat aurkeztuko du pantailan. */

 { /* Funtzioaren hasierako giltza */

   printf("\nZenbaki-segida hutsa da.");

 } /* Funtzioaren bukaerako giltza */
```

- **Erabilpena**:

  Hona hemen oraintxe definitu dugun *segida_hutsa_da* izeneko funtzioaren erabilpen-adibide bat. Kasu honetan funtzioa erabiltzeko era bakarra dago:

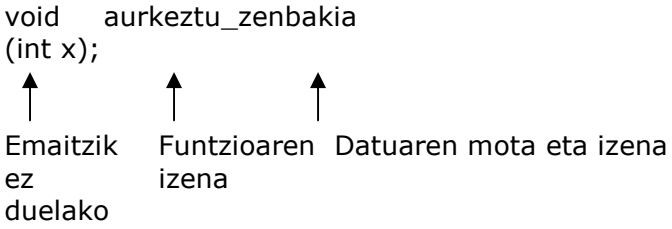
```
segida_hutsa_da ();
```

4.2.2 x zenbakia pantailan aurkezten duen funtzioa

Osoa den x zenbaki bat emanda, x zenbakia pantailan aurkezten duen funtzioa idatziko dugu jarraian.

Funtzioak datu bat izango du, hau da, funtzioari deitzerakoan zenbaki bat ipini beharko diogu parentesi artean, eta funtzioak zenbaki hori pantailan aurkeztuko du. Funtzio hau b) erakoa izango da:

- **Prototipoa**:



- **Definizioa**:

  aurkeztu_zenbakia izeneko funtzioaren definizioan datu bezala emandako x zenbakia pantailan aurkezteko egin beharrekoa zehaztu behar da:

```
void aurkeztu_zenbakia (int x)

 /* Funtzio honek x zenbakia pantailan aurkeztuko du. */

 { /* Funtzioaren hasierako giltza */

   printf("\n%d", x);

 } /* Funtzioaren bukaerako giltza */
```

- **Erabilpena**:

  Jarraian oraintxe definitu dugun *aurkeztu_zenbakia* izeneko funtzioaren erabilpen-adibide batzuk datoz.

  Adibidea:

  Demagun pantailan 5, -68 eta 40 zenbakiak aurkeztu nahi ditugula. Hori *aurkeztu_zenbakia* izeneko funtzioarekin honela egin dezakegu:

```
aurkeztu_zenbakia(5);
aurkeztu_zenbakia(-68);
aurkeztu_zenbakia(40);
```

  Adibidea:

  Pantailan n-ren balioa eta m + 1 balioa aurkeztea nahi izanez gero, *aurkeztu_zenbakia* funtzioa erabiliz honela egin ahal izango genuke:

```
int m = 10, n = 20;

aurkeztu_zenbakia(n);
aurkeztu_zenbakia(m + 1);
```

  Erabilpen-adibide hauetan ikusi den bezala, *aurkeztu_zenbakia* funtzioari deitzerakoan guk datu bat, zenbaki oso bat, eman behar diogu edo ipini behar diogu parentesi artean. Funtzioak guk emandako zenbaki hori pantailan aurkeztuko du. Beraz funtzioak ez du emaitzik itzultzen.

4.2.3 x zenbakiaren faktoriala aurkezten duen funtzioa

Osoa den x zenbaki bat eta f bere faktoriala emanda, x zenbakiaren faktoriala f dela esaten duen funtzioa idatzi nahi da. Funtzioari deitzerakoan zenbaki bat eta bere faktoriala eman beharko zaizkio datu bezala, eta funtzioak bi zenbaki horiek aurkeztuko ditu, baina ez du emaitzik itzuliko. Funtzio hau ere b) erakoa izango da:

- **Prototipoa**:

  void aurkeztu_faktoriala (int x,
  long f);

  Emaitzik ez duelako — Funtzioaren izena — Datuen motak eta izenak

- **Definizioa**:

  aurkeztu_faktoriala izeneko funtzioaren definizioan datu bezala emandako x zenbakia eta bere faktoriala, f, pantailan aurkezteko egin beharrekoa zehaztu behar da:

```
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 */
```

- **Erabilpena**:

  Jarraian oraintxe definitu dugun *aurkeztu_faktoriala* izeneko funtzioaren erabilpen-adibide batzuk datoz.

  Adibidea:
  Demagun 120 zenbakia 5en faktoriala dela esanez mezu bat aurkeztu nahi dugula pantailan. Hori *aurkeztu_faktoriala* funtzioa erabiliz honela egingo genuke:

```
aurkeztu_faktoriala(5, 120);
```

  Adibidea:
  Orain 8ren faktoriala kalkulatu eta aurkeztuko dugu:

```
long fakt;

fakt = kalkulatu_faktoriala(8);
aurkeztu_faktoriala(8, fakt);
```

  Adibidea:
  Orain n eta (m + 3)-ren faktorialak kalkulatu eta aurkezteko egin beharrekoa idatziko dugu:

```
int n = 10, m = 6;
long fakt1, fakt2;

fakt1 = kalkulatu_faktoriala(n);
aurkeztu_faktoriala(n, fakt1);
fakt2 = kalkulatu_faktoriala(m + 3);
aurkeztu_faktoriala(m + 3, fakt2);
```

4.2.4 x zenbakia lehena dela edo ez dela esanez mezu bat aurkezten duen funtzioa

Osoa den x zenbaki bat eta zenbaki hori lehena al den ala ez hartutako e erabakia (0 edo 1) emanda, x lehena dela edo ez dela esanez mezu bat aurkezten duen funtzioa idatziko da jarraian.

Bigarren argumentuak, e-k, 0 edo 1 balioko du beti eta bere esanahia honako hau da:

- e-ren balioa 0 bada, x ez dela lehena adierazi nahi da.
- e-ren balioa 1 bada, x lehena dela adierazi nahi da.

- **Prototipoa**:

```
void aurkeztu_lehena (int x,
int e);
```

Emaitzik ez duelako — Funtzioaren izena — Datuen motak eta izenak

- **Definizioa**:

aurkeztu_lehena izeneko funtzioaren definizioan datu bezala emandako x zenbakia eta zenbaki hori lehena al den ala ez adierazten duen e balioa emanda, x lehena dela edo ez dela esanez mezu bat pantailan aurkezteko egin beharrekoa zehaztu behar da:

```
void aurkeztu_lehena (int x, int e)

 /* Funtzio honek e-ren balioa 1 bada, x zenbakia lehena dela esanez
    mezu bat aurkeztuko du eta e-ren balioa 0 bada, x zenbakia ez dela
    lehena esanez mezu bat aurkeztuko du pantailan. */

 { /* Funtzioaren hasierako giltza */
   if (e == 1)
    {
      printf("\n%d zenbakia lehena da.", x);
    }
   else
    {
      printf("\n%d zenbakia ez da lehena.", x);
    }
 } /* Funtzioaren bukaerako giltza */
```

- **Erabilpena**:

Jarraian oraintxe definitu dugun *aurkeztu_lehena* izeneko funtzioaren erabilpen-adibide batzuk datoz.

Adibidea:
Demagun 19 zenbakia lehena dela esanez mezu bat aurkeztu nahi dugula pantailan. Hori *aurkeztu_lehena* funtzioa erabiliz honela egingo genuke:

```
aurkeztu_lehena(19, 1);
```

Adibidea:
Orain demagun 27 zenbakia ez dela lehena esanez mezu bat aurkeztu nahi dugula pantailan. Hori *aurkeztu_lehena* funtzioa erabiliz honela egingo genuke:

```
aurkeztu_lehena(27, 0);
```

Adibidea:
Orain n aldagaian daukagun balioa lehena al den ala ez aztertu eta dagokion mezua aurkezteko egin beharrekoa azalduko dugu. Kasu honetan n aldagaian daukagun balioa lehena al den erabakitzeko *lehena_al_da* funtzioa erabiliko dugu:

```
int n = 20;

if (lehena_al_da (n) == 1)
 {
   aurkeztu_lehena (n, 1);
 }
else
 {
   aurkeztu_lehena (n, 0);
 }
```

Gauza bera beste era honetara ere egin daiteke:

```
int n = 20;

aurkeztu_lehena (n, lehena_al_da(n));
```

5 KALKULUAK BURUTU, KALKULATUTAKO EMAITZAK AURKEZTU ETA BESTE FUNTZIOEI DEITZEAZ ARDURATZEN DIREN FUNTZIOAK

5.1 Ezaugarriak eta definizioa

Era honetako funtzioak ere argumentudunak edo argumenturik gabekoak izan daitezke. Argumenturik gabekoek beraiek lortu beharko dituzte datuak gero kalkuluren batzuk burutu eta emaitzak aurkeztuz bukatzeko. Bai datuak lortzeko, bai kalkuluak burutzeko eta bai emaitzak aurkezteko orduan beraiek zuzenean egin dezakete egin beharrekoa edo bestela beste funtzio batzuei deituz. Argumentudun funtzioen kasuan, argumentuak izango dira datuak normalean.

Kasu honetan prototipoaren itxura eta funtzioak erabiltzeko era aurreko atalean azaldutakoen berdina izango da.

5.2 Adibideak

5.2.1 Erabiltzaileak tekleatutako zenbaki-segidako balio handiena eta txikiena zein diren erabakitzen duen funtzioa

Zenbaki-segida bateko elementuak banan-banan eskatu eta balio handiena eta txikiena zein diren erabaki eta erabakiaren berri emanez mezu bat aurkezten duen funtzioa idatziko da jarraian. Funtzio horrek, egin beharreko gauza gehienak beste funtzioei deituz egingo ditu, izan ere, *eskatu_segidako_zenbakia*, *aurkeztu_max_min* eta *segida_hutsa_da* funtzioak erabiliko ditu. Funtzio horietako lehenengoak erabiltzaileari zenbakiak banan-banan eskatuz joateko balio du (dei bakoitzean zenbaki bat eskatuko du), bigarrenak balio handiena eta txikiena kalkulatu ondoren aurkezteko eta hirugarrenak zenbaki-segida hutsa izan bada, horren berri emanez mezu bat aurkezteko.

Funtzio honek ez du daturik eta ez du emaitzik:

- **Prototipoa**:

  void kalkulatu_segidako_max_min (void);

  Emaitzik ez duelako — Funtzioaren izena — Datuarik ez duelako

- **Definizioa**:

  kalkulatu_segidako_max_min izeneko funtzioaren definizioan erabiltzailea tekleatuz joango den zenbaki-segidako balio handiena eta txikiena zein diren erabakitzeko egin beharrekoa zehaztu behar da:

```
void kalkulatu_segidako_max_min (void)

 /* Funtzio honek erabiltzaileari zenbaki-segida bateko elementuak
    banan-banan eskatuko dizkio eta zenbaki-segidako balio handiena eta
    txikiena kalkulatu eta aurkeztuko ditu. Zenbaki-segidako elementu
    denek >= 0 izan behar dute, azkenak izan ezik, azkenak negatiboa
    izan behar baitu. */

 { /* Funtzioaren hasierako giltza */
   int zenbakia, handi, txiki, kont;

   zenbakia = eskatu_segidako_zenbakia(1);

   if (zenbakia < 0)
    {
      segida_hutsa_da();
    }
   else
    {
      kont = 1; /* Orain arte zenbat zenbaki eskatu ditugun adierazten
                   digu. */
      handi = zenbakia;
      txiki = zenbakia; /* Hasieran lehenengo zenbakia da orain arteko
                           handiena eta txikiena. */

      zenbakia = eskatu_segidako_zenbakia(kont + 1); /* segidako
                                          bigarren zenbakia lortzeko */

      while (zenbakia >= 0)
       {
         kont = kont + 1;

         if (zenbakia > handi)
          {
```

```
            handi = zenbakia;
          }
        else if (zenbakia < txiki)
          {
            txiki = zenbakia;
          }
        zenbakia = eskatu_segidako_zenbakia(kont + 1); /* Hurrengo
                                                  zenbakia eskatzeko */
      }
     aurkeztu_segidako_max_min(handi, txiki);
   }
} /* Funtzioaren bukaerako giltza */
```

- **Erabilpena**:

  Hona hemen oraintxe definitu dugun *kalkulatu_segidako_max_min* izeneko funtzioaren erabilpen-adibide bat. Kasu honetan ere funtzioa erabiltzeko era bakarra dago:

```
kalkulatu_segidako_max_min
();
```

5.2.2 Lehenengo z zenbaki lehenen faktorialak kalkulatu eta aurkezten dituen funtzioa

Osoa eta ≥ 1 den z zenbaki bat emanda, lehenengo z zenbaki lehenen faktorialak kalkulatu eta aurkezten dituen funtzioa idatziko da jarraian.

Funtzio honek aurretik azaldu ditugun *kalkulatu_faktoriala*, *lehena_al_da*, *aurkeztu_lehen*a eta *aurkeztu_faktoriala* funtzioak erabiliko ditu.

Funtzio honetan funtzioari emandako argumentua edo datua kontuan hartuz eta beste funtzio batzuei deituz kalkuluak burutu eta emaitzak aurkeztu beharko dira.

- **Prototipoa**:

```
void aurkeztu_lehenen_faktorialak (int z);
```

Emaitzik duelako itzultzen. Emaitzak pantailan aurkeztuko dira.	ez	Funtzioaren izena	Datuaren mota eta izena

- **Definizioa**:

  aurkeztu_lehenen_faktorialak izeneko funtzioaren definizioan lehenengo z zenbaki lehenen faktorialak kalkulatu eta aurkezteko egin beharrekoa zehaztu behar da:

```
void aurkeztu_lehenen_faktorialak (int z)

 /* Funtzio honek lehenengo z zenbaki lehenen faktorialak kalkulatu eta
    aurkeztuko ditu pantailan. */

 { /* Funtzioaren hasierako giltza */
```

```
   int zenb, kont;
   long fakt;

   zenb = 1; /* Aldagai hau zenbakiak banan-banan pasatzeko erabiliko
                da. */
   kont = 0; /* Aldagai hau une bakoitzean une horretara arte zenbat
                zenbaki lehenen faktorialak aurkeztu diren jakiteko
                erabiliko da. */

   while (kont < z)
    {
      if (lehena_al_da (zenb) == 1)
       {
         aurkeztu_lehena(zenb, 1);
         fakt = kalkulatu_faktoriala(zenb);
         aurkeztu_faktoriala(zenb, fakt);
         kont = kont + 1;
       }
      zenb = zenb + 1;
    }
} /* Funtzioaren bukaerako giltza */
```

- **Erabilpena**:

  Jarraian oraintxe definitu dugun *aurkeztu_lehenen_faktorialak* izeneko funtzioaren erabilpen-adibide batzuk datoz.

  Adibidea:

  Demagun lehenengo lau zenbaki lehenen faktorialak kalkulatu eta aurkeztu nahi ditugula pantailan. Hori *aurkeztu_lehenen_faktorialak* funtzioa erabiliz honela egingo genuke:

```
aurkeztu_lehenen_faktorialak
(4);
```

  Agindu horrekin pantailan 2, 3, 5 eta 7ren faktorialak aurkeztuko lizkiguke ordenagailuak:

Pantaila

```
2 lehena da.
2(r)en faktoriala 2 da.
3 lehena da.
3(r)en faktoriala 6 da.
5 lehena da.
5(r)en faktoriala 120 da.
7 lehena da.
7(r)en faktoriala 5040 da.
```

6 EBATZITAKO ARIKETAK

6.1 Sei, zortzi eta hamabosten faktorialak kalkulatzen dituen programa

Ariketa honetan 6, 8 eta 15en faktorialak kalkulatu eta pantailan aurkezten dituen programa idatzi behar da.

6.1.1 Lehenengo bertsioa: *kalkulatu_faktoriala* funtzioa erabiliz

Zeroren berdina edo handiagoa den zenbaki oso baten faktoriala kalkulatzen duen funtzioa nola definitzen den azaldu dugu lehenago, baina ariketa honen bidez funtzio hori programa batean non kokatu eta nola erabili ikusiko dugu.

Erabili beharreko funtzioak:

- *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 ≥ 0 den x zenbaki bat eta f bere faktoriala emanda, f zenbakia x zenbakiaren faktoriala dela esaten duen mezua aurkeztuko du funtzio honek.

```
#include <stdio.h>

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

/* Emaitzak: A, B eta C-ren faktorialak kalkulatu eta aurkeztuko ditu
             pantailan. */

/* Konstanteak:
          - A, B, C: Programak hiru konstante hauen faktorialak kalkulatu eta
                     aurkeztuko ditu. */

/* Aldagaiak:
          - f1, f2, f3: A, B eta C-ren faktorialak gordetzeko erabiliko dira
                        hurrenez-hurren. */

#define A 6
#define B 8
#define C 15

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

long kalkulatu_faktoriala (int x);

void aurkeztu_faktoriala (int x, long f);

/* Funtzio nagusia */

void main()
 { /* main funtzioaren hasierako giltza */
   long f1, f2, f3;

   f1 = kalkulatu_faktoriala (A);
   f2 = kalkulatu_faktoriala (B);
   f3 = kalkulatu_faktoriala (C);

   aurkeztu_faktoriala (A, f1);
   aurkeztu_faktoriala (B, f2);
   aurkeztu_faktoriala (C, f3);

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

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

/* Erabilitako funtzioen definizioak */

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 */
```

Ordenagailuak main funtzioa exekutatzen du beti. Eta main funtzioan beste funtzio bat erabiltzen bada, hau da, main funtzioan beste funtzio bati deitzen bazaio, orduan funtzioari deitzerakoan ipinitako argumentuekin funtzioaren definizioan parametro formalekin egiten dena egingo du.

Ariketan honetan main funtzioan *kalkulatu_faktoriala* funtzioa hiru aldiz erabiltzen da: *kalkulatu_faktoriala (A)*, *kalkulatu_faktoriala (B)* eta *kalkulatu_faktoriala (C)*. A, B eta C argumentuak direla esaten da eta funtzioaren definizioan erabili den x izena parametro formala dela esaten da.

Ariketa honetan idatzitako programa exekutatzerakoan, hau da, main funtzioa exekutatzerakoan, ordenagailuak *kalkulatu_faktoriala(A)* aurkitzen duenean, *kalkulatu_faktoriala* izeneko funtzioaren definizioan x-ekin egiten dena orain A balioarekin egingo du, eta horrela A balioaren faktoriala kalkulatuko du.

6.1.2 Bigarren bertsioa: Funtziorik erabiltzeke

Jarraian ariketa bera funtziorik gabe egingo da. Hemen ikusi ahal izango denez, A, B eta C-ren faktorialak kalkulatzeko, zenbaki baten faktoriala kalkulatzeko jarraitu beharreko prozesua hiru aldiz ipini beharko da:

```
#include <stdio.h>

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

/* Emaitzak: A, B eta C-ren faktorialak kalkulatu eta aurkeztuko ditu
             pantailan. */
```

```
/* Konstanteak:
          - A, B, C: Programak hiru konstante hauen faktorialak kalkulatu eta
                     aurkeztuko ditu. */

/* Aldagaiak:
          - f1, f2, f3: A, B eta C-ren faktorialak gordetzeko erabiliko dira
                        hurrenez-hurren. */

#define A 6
#define B 8
#define C 15

void main()
 { /* main funtzioaren hasierako giltza */
   int zenb;
   long f1, f2, f3;

   /* A-ren faktorialaren kalkulua */
   if (A == 0)
    {
      f1 = 1;
    }
   else
    {
      f1 = 1;
      for (zenb = 1; zenb <= A; zenb = zenb + 1)
       {
         f1 = f1 * zenb;
       }
    }

   /* B-ren faktorialaren kalkulua */
   if (B == 0)
    {
      f2 = 1;
    }
   else
    {
      f2 = 1;
      for (zenb = 1; zenb <= B; zenb = zenb + 1)
       {
         f2 = f2 * zenb;
       }
    }



   /* C-ren faktorialaren kalkulua */
   if (C == 0)
    {
      f3 = 1;
    }
   else
    {
      f3 = 1;
      for (zenb = 1; zenb <= C; zenb = zenb + 1)
       {
         f3 = f3 * zenb;
       }
    }

   printf("\n%d(r)en faktoriala %ld da.", A, f1);
   printf("\n%d(r)en faktoriala %ld da.", B, f2);
   printf("\n%d(r)en faktoriala %ld da.", C, f3);

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

Ariketaren bigarren bertsio hau aztertuz, *kalkulatu_faktoriala* izeneko funtzioa definitzearen abantailak ikus ditzakegu:

- 1. bertsioko programa laburragoa da.
- 1. bertsioa ulerterrazagoa da.
- 1. bertsioan faktoriala nola kalkulatzen den behin bakarrik azaldu da eta 2. bertsioan hiru aldiz.

6.2 Erabiltzaileak emandako hiru zenbaki lehenak al diren ala ez erabaki

Ariketa honetan idatzi beharreko programak ≥ 1 diren m, n eta p hiru zenbaki oso tekleatzeko eskatuko dio erabiltzaileari eta tekleatutako zenbaki bakoitza lehena al den ala ez esanez mezu bat aurkeztuko du.

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.

- *lehena_al_da*: argumentu bezala ≥ 1 den x zenbaki oso 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.

- *aurkeztu_lehena*: argumentu bezala x zenbaki bat eta x zenbakiari buruzko e erabakia (1 edo 0) emanda, funtzio honek e-ren balioa 1 bada, x lehena dela esanez mezu bat aurkeztu behar du eta e-ren balioa 0 bada, x ez dela lehena esanez mezu bat aurkeztu behar du.

```
#include <stdio.h>

/* Datuak: Programa honek >= 1 diren hiru zenbaki eskatuko dizkio
           erabiltzaileari. */

/* Emaitzak: Erabiltzaileak tekleatutako hiru zenbakiak lehenak al diren ala
             ez esanez hiru mezu aurkeztuko ditu pantailan. */

/* Aldagaiak:
          - m, n eta p: Erabiltzaileak tekleatu beharreko hiru zenbakiak
                        jasotzeko. */

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

int eskatu_zenbaki_handiagoa (int z);
int lehena_al_da (int x);
void aurkeztu_lehena (int x, int e);

/* Funtzio nagusia */

void main()
 { /* main funtzioaren hasierako giltza */
   int m, n, p;

   m = eskatu_zenbaki_handiagoa(0);
   n = eskatu_zenbaki_handiagoa(0);
   p = eskatu_zenbaki_handiagoa(0);

   if (lehena_al_da (m) == 1)
```

```
    {
      aurkeztu_lehena (m, 1);
    }
   else
    {
      aurkeztu_lehena (m, 0);
    }

   if (lehena_al_da (n) == 1)
    {
      aurkeztu_lehena (n, 1);
    }
   else
    {
      aurkeztu_lehena (n, 0);
    }

   if (lehena_al_da (p) == 1)
    {
      aurkeztu_lehena (p, 1);
    }
   else
    {
      aurkeztu_lehena (p, 0);
    }

   printf("\nSakatu tekla bat amaitzeko.");
   getch(); /* Guk 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 */

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

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 */

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

void aurkeztu_lehena (int x, int e)

 /* Funtzio honek e-ren balioa 1 bada, x zenbakia lehena dela esanez mezu bat
    aurkeztuko du eta e-ren balioa 0 bada, x zenbakia ez dela lehena esanez
    mezu bat aurkeztuko du pantailan. */
{
   if (e == 1)
    {
      printf("\n%d zenbakia lehena da.", x);
    }
   else
    {
      printf("\n%d zenbakia ez da lehena.", x);
    }
}

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

6.3 Lehenengo n zenbaki lehenak aurkitu eta aurkeztu

Ariketa honetan idatzi beharreko programak erabiltzaileari ≥ 1 den n zenbaki oso bat tekleatzeko eskatuko dio eta baldintza hori betetzen duen n zenbakia lortu ondoren, lehenengo n zenbaki lehenak aurkitu eta aurkeztuko ditu.

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.

- *aurkitu_lehenak*: argumentu bezala ≥ 1 diren v eta z zenbaki osoak emanda, v-tik hasita lehenengo z zenbaki lehenak aurkitu eta aurkezten dituen funtzioa. Funtzio honek *lehena_al_da* eta *aurkeztu_lehena* funtzioak erabili behar ditu.

- *lehena_al_da*: argumentu bezala ≥ 1 den x zenbaki oso 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.

- *aurkeztu_lehena*: argumentu bezala x zenbaki bat eta x zenbakiari buruzko e erabakia (1 edo 0) emanda, funtzio honek e-ren balioa 1 bada, x lehena dela esanez mezu bat aurkeztu behar du eta e-ren balioa 0 bada, x ez dela lehena esanez mezu bat aurkeztu behar du.

```
#include <stdio.h>

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

/* Emaitzak: Lehenengo n zenbaki lehenak aurkeztuko ditu pantailan. */

/* Aldagaiak:
          - n: Erabiltzaileak tekleatu beharreko zenbakia jasotzeko. */

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

int eskatu_zenbaki_handiagoa (int z);
void aurkitu_lehenak (int v, int z);
int lehena_al_da (int x);
void aurkeztu_lehena (int x, int e);

/* Funtzio nagusia */

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

   n = eskatu_zenbaki_handiagoa(0);

   aurkitu_lehenak (1, n); /* 1etik hasita lehenengo n zenbaki lehenak
                              aurkituko ditu */

   printf("\nSakatu tekla bat amaitzeko.");
   getch(); /* Guk 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 */

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

```
void aurkitu_lehenak (int v, int z)

 /* Funtzio honek v-tik hasita lehenengo z zenbaki lehenak aurkitu eta
    aurkeztuko ditu. v eta z >= 1 izango dira. */

 {
   int zenb, kont;

   zenb = v; /* Zenbakiak v-tik hasita banan-banan pasatzeko erabiliko da.
                While-eko buelta bakoitzean zenb aldagaian 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("\n%d-tik hasita lehenengo %d zenbaki lehenak honako hauek dira: ",
          v, z);

   while (kont < z)
    {
      if (lehena_al_da (zenb) == 1)
       {
         aurkeztu_lehena (zenb, 1);
         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 */

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

void aurkeztu_lehena (int x, int e)

 /* Funtzio honek e-ren balioa 1 bada, x zenbakia lehena dela esanez mezu bat
    aurkeztuko du eta e-ren balioa 0 bada, x zenbakia ez dela lehena esanez
```

```
    mezu bat aurkeztuko du pantailan. */

 {
   if (e == 1)
    {
      printf("\n%d zenbakia lehena da.", x);
    }
   else
    {
      printf("\n%d zenbakia ez da lehena.", x);
    }
 }

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

6.4 n baino txikiagoak diren zenbaki lehenak aurkitu eta aurkeztu

Ariketa honetan idatzi beharreko programak erabiltzaileari ≥ 1 den n zenbaki oso bat tekleatzeko eskatuko dio eta baldintza hori betetzen duen n zenbakia lortu ondoren, n baino txikiagoak diren zenbaki lehenak aurkitu eta aurkeztuko ditu.

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.

- *aurkitu_lehen_txikiagoak*: argumentu bezala z zenbaki oso bat emanda, z baino txikiagoak diren zenbaki lehenak aurkitu eta aurkezten dituen funtzioa. Funtzio honek *lehena_al_da* eta *aurkeztu_lehena* funtzioak erabili behar ditu.

- *lehena_al_da*: argumentu bezala ≥ 1 den x zenbaki oso 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.

- *aurkeztu_lehena*: argumentu bezala x zenbaki bat eta x zenbakiari buruzko e erabakia (1 edo 0) emanda, funtzio honek e-ren balioa 1 bada, x lehena dela esanez mezu bat aurkeztu behar du eta e-ren balioa 0 bada, x ez dela lehena esanez mezu bat aurkeztu behar du.

```
#include <stdio.h>

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

/* Emaitzak: n baino txikiagoak diren zenbaki lehenak aurkeztuko ditu
             pantailan. */

/* Aldagaiak:
          - n: Erabiltzaileak tekleatu beharreko zenbakia jasotzeko. */

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

int eskatu_zenbaki_handiagoa (int z);
void aurkitu_lehen_txikiagoak (int z);
int lehena_al_da (int x);
void aurkeztu_lehena (int x, int e);
```

→

```
/* Funtzio nagusia */

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

   n = eskatu_zenbaki_handiagoa (0); /* 0 baino handiagoa den zenbaki oso bat
                                        nahi delako */

   aurkitu_lehen_txikiagoak (n); /* n baino txikiagoak diren zenbaki lehenak
                                    aurkituko ditu */

   printf("\nSakatu tekla bat amaitzeko.");
   getch(); /* Guk 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 */

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

void aurkitu_lehen_txikiagoak (int z)

 /* Funtzio honek z baino txikiagoak diren  zenbaki lehenak aurkitu eta
    aurkeztuko ditu. z >= 1 izango da. */
 {
   int zenb;

   zenb = 1; /* Zenbakiak 1etik hasita banan-banan pasatzeko erabiliko da.
                While-eko buelta bakoitzean zenb aldagaian dagoen balioa
                lehena al den ala ez aztertuko da. */

   printf("\n%d baino txikiagoak diren zenbaki lehenak honako hauek dira: ",
          z);

   while (zenb < z)
    {
      if (lehena_al_da (zenb) == 1)
       {
         aurkeztu_lehena (zenb, 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 */

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

void aurkeztu_lehena (int x, int e)

 /* Funtzio honek e-ren balioa 1 bada, x zenbakia lehena dela esanez mezu bat
    aurkeztuko du eta e-ren balioa 0 bada, x zenbakia ez dela lehena esanez
    mezu bat aurkeztuko du pantailan. */

 {
   if (e == 1)
    {
      printf("\n%d zenbakia lehena da.", x);
    }
   else
    {
      printf("\n%d zenbakia ez da lehena.", x);
    }
 }

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

6.5 n zenbakia beste zenbakiren baten faktoriala al den erabaki

Ariketa honetan idatzi beharreko programak erabiltzaileari ≥ 1 den n zenbaki oso bat tekleatzeko eskatuko dio eta baldintza hori betetzen duen n zenbakia lortu ondoren, n beste zenbakiren baten faktoriala al den ala ez erabakiko du, hau da, m! = n betetzen duen m zebakirik existitzen al den ala ez erabakiko du.

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.

- *erabaki_faktoriala*: argumentu bezala ≥ 1 den z zenbaki oso bat emanda, z beste zenbakiren baten faktoriala al den ala ez erabaki eta hartutako erabakiaren berri emanez mezu bat aurkeztuko duen funtzioa. Funtzio honek *kalkulatu_faktoriala* funtzioa erabili behar du.

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

```
#include <stdio.h>

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

/* Emaitzak: n zenbakia zenbakiren baten faktoriala bada, beste zenbaki hori
             aurkeztuko du eta bestela, n ez dela beste zenbaki baten
             faktoriala esanez mezu bat aurkeztuko du. */

/* Aldagaiak:
          - n: Erabiltzaileak tekleatu beharreko zenbakia jasotzeko. */

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

int eskatu_zenbaki_handiagoa (int z);
void erabaki_faktoriala (int z);
long kalkulatu_faktoriala (int x);

/* Funtzio nagusia */

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

   n = eskatu_zenbaki_handiagoa (0); /* 0 baino handiagoa den zenbaki oso bat
                                        nahi delako */

   erabaki_faktoriala (n); /* n beste zenbakiren baten faktoriala al den
                              erabakiko du */

   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 */

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

void erabaki_faktoriala (int z)

 /* Funtzio honek z zenbakia beste zenbakiren baten faktoriala al den ala ez
    erabaki eta erabakitakoaren berri emanez mezu bat aurkeztuko du. */

 {
   int zenb;
   long fakt;

   if (z == 1) /* Kasu berezia */
    {
      printf ("\n%d 0 eta 1en faktoriala da.", z);
    }
   else
    {
      zenb = 2; /* Zenbakiak 2tik hasita banan-banan pasatzeko erabiliko da.
                   While-eko buelta bakoitzean zenb aldagaian dagoen balioaren
                   faktoriala z al den ala ez aztertuko da. */

      fakt = kalkulatu_faktoriala (zenb);

      while (fakt < z)
       {
         zenb = zenb + 1;
         fakt = kalkulatu_faktoriala (zenb);
       }
      if (fakt == z)
       {
         printf("\n%d %d zenbakiaren faktoriala da.", z, zenb);
       }
      else
       {
         printf("\nEz dago faktorial bezala %d duen zenbakirik.", z);
       }
    }
 }

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

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

6.6 Lehenengo n zenbaki lehenen faktorialak aurkitu eta aurkeztu

Ariketa honetan idatzi beharreko programak erabiltzaileari ≥ 1 den n zenbaki oso bat tekleatzeko eskatuko dio eta baldintza hori betetzen duen n zenbakia lortu ondoren, lehenengo n zenbaki lehenen faktorialak aurkitu eta aurkeztuko ditu.

Erabili beharreko funtzioak:

- *eskatu_zenbaki_handiag*oa: 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.

- *aurkeztu_lehenen_faktorialak*: argumentu bezala z zenbaki oso bat emanda, lehenengo z zenbaki lehenen faktorialak kalkulatu eta aurkezten dituen funtzioa. Funtzio honek *lehena_al_da*, *kalkulatu_faktoriala*, *aurkeztu_lehena* eta *aurkeztu_faktoriala* funtzioak erabili behar ditu.

- *lehena_al_da*: argumentu bezala ≥ 1 den x zenbaki oso 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.

- *aurkeztu_lehena*: argumentu bezala x zenbaki bat eta x zenbakiari buruzko e erabakia (1 edo 0) emanda, funtzio honek e-ren balioa 1 bada, x lehena dela esanez mezu bat aurkeztu behar du eta e-ren balioa 0 bada, x ez dela lehena esanez mezu bat aurkeztu behar du.

- *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 ≥ 0 den x zenbaki bat eta f bere faktoriala emanda, f zenbakia x zenbakiaren faktoriala dela esaten duen mezua aurkeztuko du funtzio honek.

```
#include <stdio.h>

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

/* Emaitzak: Lehenengo n zenbaki lehenen faktorialak aurkeztuko ditu
             pantailan. */

/* Aldagaiak:
          - n: Erabiltzaileak tekleatu beharreko zenbakia jasotzeko. */

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

int eskatu_zenbaki_handiagoa (int z);
void aurkeztu_lehenen_faktorialak (int z);
int lehena_al_da (int x);
void aurkeztu_lehena (int x, int e);
long kalkulatu_faktoriala (int x);
void aurkeztu_faktoriala (int x);

/* Funtzio nagusia */

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

   n = eskatu_zenbaki_handiagoa (0);

   aurkeztu_lehenen_faktorialak (n); /* letik hasita lehenengo n zenbaki
                                        lehenen faktorialak aurkituko ditu */

   printf("\nSakatu tekla bat amaitzeko.");
   getch(); /* Guk 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 */

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


void aurkeztu_lehenen_faktorialak (int z)
```

```
 /* Funtzio honek lehenengo z zenbaki lehenen faktorialak kalkulatu eta
    aurkeztuko ditu pantailan. */

 {
   int zenb, kont;
   long fakt;

   zenb = 1; /* Aldagai hau zenbakiak banan-banan pasatzeko erabiliko da. */
   kont = 0; /* Aldagai hau une bakoitzean une horretara arte zenbat zenbaki
                lehenen faktorialak aurkeztu diren jakiteko erabiliko da. */

   while (kont < z)
    {
      if (lehena_al_da (zenb) == 1)
       {
         aurkeztu_lehena (zenb, 1);
         fakt = kalkulatu_faktoriala (zenb);
         aurkeztu_faktoriala (zenb, fakt);
         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 */

                              /********************/
void aurkeztu_lehena (int x, int e)

 /* Funtzio honek e-ren balioa 1 bada, x zenbakia lehena dela esanez mezu bat
    aurkeztuko du eta e-ren balioa 0 bada, x zenbakia ez dela lehena esanez
    mezu bat aurkeztuko du pantailan. */

 {
   if (e == 1)
    {
      printf("\n%d zenbakia lehena da.", x);
    }
```

```
   else
    {
      printf("\n%d zenbakia ez da lehena.", x);
    }
 }

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

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. */

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

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

6.7 Lehenengo n faktorial lehenak aurkitu eta aurkeztu

Ariketa honetan idatzi beharreko programak erabiltzaileari ≥ 1 den n zenbaki oso bat tekleatzeko eskatuko dio eta baldintza hori betetzen duen n zenbakia lortu ondoren, lehenengo n faktorial lehenak aurkitu eta aurkeztuko ditu.

Erabili beharreko funtzioak:

- *galdetu_zenbat_faktorial_lehen*: erabiltzaileari zenbat faktorial lehen nahi dituen galdetzen dion funtzioa, hau da, ≥ 1 den zenbaki oso bat tekleatzeko eskatzen dion funtzioa. Datua eskatzeko prozesua datu egoki bat lortu arte errepikatuko da.

- *kalkulatu_faktorial_lehenak*: argumentu bezala z zenbaki oso bat emanda, lehenengo z faktorial lehenak kalkulatu eta aurkezten dituen funtzioa. Funtzio honek *kalkulatu_faktoriala*, *lehena_al_da* eta *aurkeztu_faktoriala* funtzioak erabili behar ditu.

- *kalkulatu_faktoriala*: argumentu bezala ≥ 0 den x zenbaki bat emanda, x zenbaki horren faktoriala kalkulatu eta itzultzen duen 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.

- *aurkeztu_faktoriala*: argumentu bezala x zenbaki bat eta x-en faktoriala den f zenbakia emanda, f zenbakia x-en faktoriala dela esanez mezu bat aurkezten duen funtzioa.

```
#include <stdio.h>

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

/* Emaitzak: Lehenengo n faktorial lehenak aurkeztuko ditu pantailan. */

/* Aldagaiak:
          - n: Erabiltzaileak tekleatu beharreko zenbakia jasotzeko. */

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

int galdetu_zenbat_faktorial_lehen (void);
void kalkulatu_faktorial_lehenak (int z);
int kalkulatu_faktoriala (int x);
int lehena_al_da (int x);
void aurkeztu_faktoriala (int x, int f);

/* Funtzio nagusia */

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

   n = galdetu_zenbat_faktorial_lehen();

   kalkulatu_faktorial_lehenak(n); /* lehenengo n faktorial lehenak aurkituko
                                      ditu */

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

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

                    /* Erabilitako funtzioen definizioak */

int galdetu_zenbat_faktorial_lehen (void)

 /* Funtzio honek erabiltzaileari >= 1 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. */

 {
   int zenb; /* erabiltzaileak tekleatuko duen zenbakia jasotzeko erabiliko
                da. */

   do
    {
      printf("\nZenbat faktorial lehen nahi dituzu? (>= 1 den zenbaki oso "
             "bat tekleatu): ");
      scanf("%d", &zenb);
      if (zenb < 1)
```

```
        {
          printf("\nTekleatutako zenbakia ez da egokia.");
        }
    } while (zenb <= 0);

  return(zenb);
}

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

void kalkulatu_faktorial_lehenak (int z)

 /* Funtzio honek lehenengo z faktorial lehenak aurkitu eta aurkeztuko ditu. z
    >= 1 izango da. */

{
  int zenb, kont, fakto;

  zenb = 0; /* Zenbakiak 0tik hasita banan-banan pasatzeko erabiliko da.
               While-eko buelta bakoitzean zenb aldagaian dagoen balioaren
               faktoriala kalkulatuko da eta lehena al den ala ez aztertuko
               da. */

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

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

  while (kont < z)
   {
     fakto = kalkulatu_faktoriala(zenb);
     if (lehena_al_da (fakto) == 1)
      {
        aurkeztu_faktoriala (zenb, 1);
        kont = kont + 1;
      }
     zenb = zenb + 1;
   }
}

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


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 */

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

```
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 */

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

void aurkeztu_faktoriala (int x, long f)

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

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

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

6.8 Zenbaki-segida batek zenbat elementu dituen kontatu

Ariketa honetan zenbaki oso eta positiboz (≥ 0) osatutako segida bateko elementuak banan-banan eskatu eta azkenean segidan zenbat elementu dauden esanez mezu bat aurkezten duen programa idatzi behar da.

Segidako azkeneko elementua negatiboa izango da. Zenbaki-segida bat hutsa dela kontsideratuko da segidako lehenengo elementua negatiboa baldin bada.

Erabili beharreko funtzioak:

- *eskatu_kontatu_eta_aurkeztu*: zenbaki-segidako elementuak banan-banan eskatu eta segidako elementu-kopurua kalkulatzen eta aurkezten duen funtzioa. Funtzio honek *eskatu_segidako_zenbakia* eta *aurkeztu_kopurua* funtzioak erabiliko ditu.

- *eskatu_segidako_zenbakia*: argumentu bezala osoa den z zenbaki bat emanda, zenbaki segidako z. zenbakia eskatu eta jasotzen duen funtzioa. Erabiltzaileak tekleatutako zenbakia itzuliko du funtzioak emaitza bezala.

- *aurkeztu_kopurua*: argumentu bezala osoa den e zenbaki bat emanda, zenbaki-segidak e elementu dituela esaten duen mezu bat aurkezten duen funtzioa.

```
#include <stdio.h>

/* Datuak: Programa honek zenbaki-segida bateko elementuak banan-banan
           eskatuko dizkio erabiltzaileari, prozesua erabiltzaileak zenbaki
           negatibo bat tekleatzen duenean bukatuz. */

/* Emaitzak: Erabiltzaileak tekleatutako zenbaki-segidan zenbat elementu
             dauden esanez mezu bat aurkeztuko du programa honek. */

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

void eskatu_kontatu_eta_aurkeztu (void);
int eskatu_segidako_zenbakia (int z);
void aurkeztu_kopurua (int e);

/* Funtzio nagusia */

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

   eskatu_kontatu_eta_aurkeztu();

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

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

                     /* Erabilitako funtzioen definizioak */

void eskatu_kontatu_eta_aurkeztu (void)

 /* Funtzio honek erabiltzaileari zenbaki-segida bateko elementuak banan-banan
    eskatuko dizkio eta zenbaki-segidaren elementu-kopurua kalkulatu eta
    kopuru hori aurkeztuko du. Zenbaki-segidako elementu denek >= 0 izan behar
    dute, azkenak izan ezik, azkenak negatiboa izan behar baitu.*/

 {
   int zenbakia, kont;

   kont = 1; /* Aldagai honek zenbatgarren zenbakia eskatu behar dugun
                adierazten digu. Lehenengo aldian lehenengo zenbakia eskatu
                behar dugu. */

   zenbakia = eskatu_segidako_zenbakia (kont);

   while (zenbakia >= 0)
    {
      kont = kont + 1;
      zenbakia = eskatu_segidako_zenbakia (kont);
    }

   /* Jaso den azken zenbakia ez da kontatu behar negatiboa delako */

   aurkeztu_kopurua(kont - 1);
 }

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

int eskatu_segidako_zenbakia (int z)

 /* Funtzio honek erabiltzaileari zenbaki-segidako z. zenbakia eskatuko dio.
 */
```

```
 {
   int zenb; /* Erabiltzaileak tekleatuko duen zenbakia jasotzeko erabiliko
                da.*/

   printf("\nTekleatu zenbaki-segidako %d. zenbakia: ", z);
   scanf("%d", &zenb);

   return(zenb);
 }

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

void aurkeztu_kopurua (int e)

 /* Funtzio honek zenbaki-segidan e zenbaki daudela esanez mezu bat aurkeztuko
    du. */

 {
   printf("\nZenbaki-segidan %d zenbaki daude.", e);
 }

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

6.9 Zenbaki-segida bateko balio handiena eta txikiena

Ariketa honetan zenbaki oso eta positiboz (≥ 0) osatutako segida bateko elementuak banan-banan eskatu eta azkenean segidako balio handiena eta txikiena zein diren esanez mezu bat aurkezten duen programa idatzi behar da.

Segidako azkeneko elementua negatiboa izango da. Zenbaki-segida bat hutsa dela kontsideratuko da segidako lehenengo elementua negatiboa baldin bada.

Zenbaki-segida bat hutsa denean mezu bat aurkeztuko da zenbaki-segida hutsa dela esanez.

Erabili beharreko funtzioak:

- *kalkulatu_segidako_max_min*: zenbaki-segidako elementuak banan-banan eskatu eta segidako balio handiena eta txikiena kalkulatu eta aurkezten dituen funtzioa. Funtzio honek *eskatu_segidako_zenbakia*, *aurkeztu_segidako_max_min* eta *segida_hutsa_da* funtzioak erabiliko ditu.

- *eskatu_segidako_zenbakia*: argumentu bezala osoa den z zenbaki bat emanda, zenbaki-segidako z. zenbakia eskatu eta jasotzen duen funtzioa. Erabiltzaileak tekleatutako zenbakia itzuliko du funtzioak emaitza bezala.

- *aurkeztu_segidako_max_min*: argumentu bezala osoak diren h eta t bi zenbaki oso emanda, zenbaki-segidako balio handiena h eta txikiena t direla esanez mezu bat aurkezten duen funtzioa.

- *segida_hutsa_da*: zenbaki-segida hutsa dela esanez mezu bat aurkezten duen funtzioa.

```
#include <stdio.h>

/* Datuak: Programa honek zenbaki-segida bateko elementuak banan-banan
           eskatuko dizkio erabiltzaileari, prozesua erabiltzaileak zenbaki
           negatibo bat tekleatzen duenean bukatuz. */
```

```
/* Emaitzak: Erabiltzaileak tekleatutako zenbaki-segidako balio handiena eta
             txikiena zein diren esanez mezu bat aurkeztuko du programa honek.
*/

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

void kalkulatu_segidako_max_min (void);
int eskatu_segidako_zenbakia (int z);
void aurkeztu_segidako_max_min (int h, int t);
void segida_hutsa_da (void);

/* Funtzio nagusia */

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

   kalkulatu_segidako_max_min ();

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

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

                     /* Erabilitako funtzioen definizioak */

void kalkulatu_segidako_max_min (void)

 /* Funtzio honek erabiltzaileari zenbaki-segida bateko elementuak banan-banan
    eskatuko dizkio eta zenbaki-segidako balio handiena eta txikiena kalkulatu
    eta aurkeztuko ditu. Zenbaki-segidako elementu denek >= 0 izan behar dute,
    azkenak izan ezik, azkenak negatiboa izan behar baitu.*/

 {
   int zenbakia, kont, handi, txiki;

   zenbakia = eskatu_segidako_zenbakia(1);

   if (zenbakia < 0)
    {
      segida_hutsa_da()
    }
   else
    {
      kont = 1; /* Orain arte zenbat zenbaki eskatu ditugun adierazten digu */
      handi = zenbakia;
      txiki = zenbakia; /* Hasieran lehenengo zenbakia da orain arteko
                           handiena eta txikiena. */
      zenbakia = eskatu_segidako_zenbakia(kont + 1);
      while (zenbakia >= 0)
       {
         kont = kont + 1;

         if (zenbakia > handi)
          {
            handi = zenbakia;
          }

         if (zenbakia < txiki)
          {
            txiki = zenbakia;
          }

         zenbakia = eskatu_segidako_zenbakia(kont + 1);
       }

      aurkeztu_segidako_max_min (handi, txiki);
    }
```

```
 }

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

int eskatu_segidako_zenbakia (int z)

 /* Funtzio honek erabiltzaileari zenbaki-segidako z. zenbakia eskatuko dio.
 */

 {
   int zenb; /* Erabiltzaileak tekleatuko duen zenbakia jasotzeko erabiliko
                da.*/

   printf("\nTekleatu zenbaki-segidako %d. zenbakia: ", z);
   scanf("%d", &zenb);

   return(zenb);
 }

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

void aurkeztu_segidako_max_min (int h, int t)

 /* Funtzio honek zenbaki-segidan balio handiena h eta txikiena t direla
    esanez mezu bat aurkeztuko du. */

 {
   printf("\nZenbaki-segidako balio handiena %d da eta txikiena %d da.", h,
           t);
 }

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

void segida_hutsa_da (void)

 /* Funtzio honek zenbaki-segidan balio handiena h eta txikiena t direla
    esanez mezu bat aurkeztuko du. */

 {
   printf("\nZenbaki-segida hutsa da.");
 }

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

7 DATUAK DIREN PARAMETROEI BURUZKO OHARRA

Datuak diren parametroak ez dira aldatu behar inoiz funtzioaren barruan.

1. adibidea:

Esate baterako demagun datu bezala x eta y argumentuak emanda (x eta y zenbaki osoak izango dira), x-tik y-ra bitarteko zenbaki oso denak aurkezten dituen funtzioa idatzi nahi dugula.

Bi bertsio idatziko ditugu:

- Lehenengoan, funtzioaren barruan x-en balioa aldatuz joango gara. Bertsio hau ez da egokia.
- Bigarrengoan, funtzioaren barruan x ez da aldatuko. Bertsio hau egokia izango da.

Egokia ez den bertsioa:

```
void aurkeztu_tarteko_zenbakiak (int x, int y)

 /* Funtzio honek x eta y-ren arteko zenbaki oso denak aurkeztuko ditu. x
    balioa y baino handiagoa bada, ez du zenbakirik aurkeztuko. */

 { /* Funtzioaren hasierako giltza */

   while (x <= y)
    {
      printf("\n%d", x);
      x = x + 1;
    }

   /* Bertsio hau ez da zuzena datua den x parametroaren balioa aldatzen
      delako funtzioaren barruan. */
 }
```

Bertsio zuzena:

```
void aurkeztu_tarteko_zenbakiak (int x, int y)

 /* Funtzio honek x eta y-ren arteko zenbaki oso denak aurkeztuko ditu. x
    balioa y baino handiagoa bada, ez du zenbakirik aurkeztuko. */

 { /* Funtzioaren hasierako giltza */
   int zenb;

   zenb = x; /* x-tik y-ra bitarteko zenbakiak banan-banan pasatzeko */

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

   /* Bertsio hau zuzena da, datua den x parametroaren balioa ez delako
      aldatzen funtzioaren barruan. Hori lortzeko beste aldagai bat definitu
      da: zenb. */
 }
```

2. adibidea:

Jarraian beste adibide bat ikusiko dugu. Demagun datu bezala x eta y argumentuak emanda (x eta y zenbaki osoak izango dira), x zenbakia y zenbakiaz zenbat aldiz zati daitekeen kalkulatzen duen funtzioa idatzi nahi dugula.

Kasu honetan ere bi bertsio idatziko ditugu:

- Lehenengoan, funtzioaren barruan x-en balioa aldatuz joango gara. Bertsio hau ez da egokia izango.
- Bigarrengoan, funtzioaren barruan x ez da aldatuko. Bertsio hau egokia izango da.

Egokia ez den bertsioa:

```
int zatiketa_kopurua (int x, int y)

 /* Funtzio honek x zenbakia y zenbakiaz zenbat aldiz zati daitekeen kalkulatu
    eta kopuru hori itzuliko du emaitza bezala. */

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

```
  kont = 0; /* Aldagai hau x zenbakia y zenbakiaz zenbat aldiz zati daitekeen
               kontatuz joateko erabiliko da. */

  while (x % y == 0)
   {
     kont = kont + 1;;
     x = x / y;
   }

  return(kont);

  /* Bertsio hau ez da zuzena, datua den x parametroaren balioa aldatzen
     delako funtzioaren barruan. */
} /* Funtzioaren bukaerako giltza */
```

Bertsio zuzena:

```
int zatiketa_kopurua (int x, int y)

 /* Funtzio honek x zenbakia y zenbakiaz zenbat aldiz zati daitekeen kalkulatu
    eta kopuru hori itzuliko du emaitza bezala. */

 { /* Funtzioaren hasierako giltza */
   int kont, xlag;

   kont = 0; /* Aldagai hau x zenbakia y zenbakiaz zenbat aldiz zati daitekeen
                kontatuz joateko erabiliko da. */

   xlag = x; /* Aldaketak xlag-en gainean burutuko dira eta ez x-en gainean.
             */

   while (xlag % y == 0)
    {
      kont = kont + 1;;
      xlag = xlag / y;
    }

   return(kont);

   /* Bertsio hau zuzena da, datua den x parametroaren balioa ez delako
      aldatzen funtzioaren barruan. Hori lortzeko beste aldagai bat definitu
      da: xlag */

 } /* Funtzioaren bukaerako giltza */
```