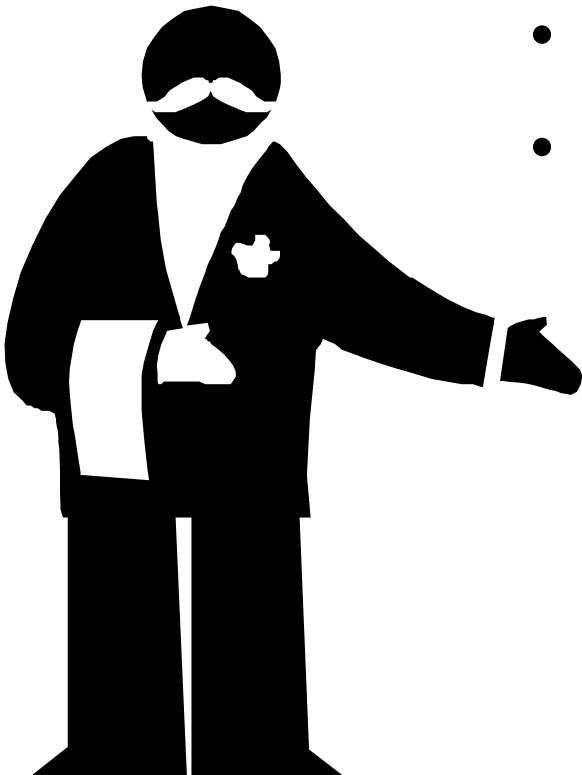


2.4 Gaia. Ada eta Python erabiltzeko oinarrizko gida

Aurkibidea

- Gaiaren helburuak
- Garapen inguruneak
- **Adaren oinarrizko gida**
- Pythonen oinarrizko gida



Ezaugarri orokorrak

- Adak moten kudeaketa oso zorrotza egiten du
- Ez ditu letra larri eta xeheak bereizten aldagaien izenetan
- Ekintzak puntu eta komaz bukatzen dira (;)
 - Ohitura ona: lerro bat = ekintza bat

Programa baten oinarrizko egitura

```
with Package_Name;  
use Package_Name;  
procedure Program_Name is  
    ---Specification  
    Variable : Some_Type;  
begin  
    Statement_1;  
    ...  
    Statement_n;  
end Program_Name;
```

Adibidea: Kaixo mundua

```
with Ada.Text_IO;  
use Ada.Text_IO;  
procedure kaixo_mundua is  
    ---Sarrera:-  
    ---Aurre: -  
    ---Irteera: mezu bat  
    ---Post: "kaixo mundua!"idatzi  
begin  
    put("kaixo mundua!");  
end kaixo_mundua;
```

Oharrak

- Gidoietatik (--), lerro bukaerara arte

```
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
    -- Goiko bi lerroak testuinguaren klausulak dira, testua eta balioosokoak
    -- irakurri eta idatzi ahal izateko.
procedure proba is
Zen1, Zen2: Integer ; --Aldagaien erazagupena
begin
put("Zenbaki osoko bat idatzi: "); -- idatzi
get(Zen1);                          -- irakurri
put("Beste zenbaki osoko bat idatzi:");
get(Zen2);
new_line; -- Lerro saltoa (pantailan)
put("Gehiketaren balioa: ");
put(Zen1 + Zen2);
end proba;
```

Oinarrizko datu motak Adaz (I)

- Integer (zenbaki osokoak)
 - Tarteak [Integer'First, Integer'Last]
 - Eragiketak =, /=, >, >=, <, <=, +, -, *, /, rem, **
- Float (zenbaki errealak)
 - Tarteak [Float'First, Float'Last]
 - Eragiketak =, /=, >, >=, <, <=, +, -, *, /, **
(float** integer)

Oinarrizko datu motak Adaz (II)

- Character (karakterearak)
 - Eragiketak =, /=, >, >=, <, <=
- Boolean (boolearrak)
 - Eragiketak and, or, xor, not
- String (karaktere kateak)
 - Eragiketak =, /=, >, >=, <, <=

Aldagaien erazagupena

- Adibideak
 - Zen1, Zen2 : Integer;
 - Erro_karratua : Float;
 - Kar, Letra : Character;
 - Lerroa : String(1..80);
- Hasieraketarekin
 - Salataria : Boolean:= False;
 - Kont : Integer := 0;

Konstanteen erazagupena

- **constant** hitz erreserbatuarekin erazagutzen dira eta hasierako balioa ematen zaie
 - Izena : constant String(1..4) := “ACME”;
 - Pi : constant Float := 3.1416;
 - Max : constant Integer := 100;

Adi, galdera

- Zergatik da interesgarria konstante baten erazagupena eta erabilera, bere balioa zuzenean aldagai batean erabili beharrean?



Esleipena

- `:=` bidez adierazten da
- Moten baliokidetza egotea ezinbestekoa da
 - Aldagaiaren mota, adierazpenaren motarekin bat egin behar du

```
Batazbestekoa : Float ;  
Notak, Ikasle_kop := Integer ;
```

```
Batazbestekoa := Notak / Ikasle_kop ;  
-- ERRORRA!
```

```
Batabestekoa      :=      float(Notak)      /  
float(Ikasle_kop) ;
```

Datuen sarrera eta irteera

- get (irakurri) eta put (idatzi) eragileek egiten dute
 - get/put bikote bat dago osokoentzat, beste bat errealentzat, eta abar.
 - with Ada.Text_IO; use Ada.Text_IO;
 - with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
 - with Ada.Float_Text_IO; use Ada.Float_Text_IO;



Modu honetan, put edo get eragileakerabiltzean, ez dugu mota zehatza adierazi beharko, ADAk testuinguruagatik ebatziko baitu. Horri, eragileen gainkarga deritzo.

Baldintzazko egiturak

- if *baldintza* then
 ekintza(k);
end if;

- if *baldintza* then
 ekintza(k);
else
 ekintza(k);
end if;

- if *baldintza* then
 ekintza(k);
elseif *baldintza* then
 ekintza(k);
...
[else
 ekintza(k);
end if;

Adibideak

...

```
if Zen < 0 then
    put("Zenbakia negatiboa da");
end if;
```

...

```
if Zen rem 2 = 0 then
    put("Zenbakia bikoitia da");
else
    put("Zenbakia bakoitia da");
end if;
```

...

...

```
if Zen < 0 then
    put("Zenbakia negatiboa da");
elsif Zen = 0 then
    put("Zenbakia zero da");
else
    put("Zenbakia positiboa da");
end if;
```

...

Iteraziozko egiturak

- loop
 $ekintza(k);$
exit when $baldintza;$
end loop;
- loop exit when $baldintza;$
 $ekintza(k);$
end loop;
- while $baldintza$ loop
 $ekintza(k);$
end loop;
- for $aldagaia$ in $n1 .. n2$ loop
 $ekintza(k);$
end loop;
- for $aldagaia$ in reverse $n2..n1$
loop
 $ekintza(k);$
end loop;

Adibideak

...

```
Kont := 1;
loop exit when Kont > 0;
  put(Kont);
  Kont := Kont + 1;
end loop;
```

...

```
Kont := 1;
while Kont <= Zenb loop
  put(Kont);
  Kont := Kont + 1;
end loop;
```

...

...

```
for Kont in 1 .. Zenb loop
  put(Kont);
end loop;
```

...

```
for Kont in reverse Zenb .. 1 loop
  put(Kont);
end loop;
```

...

```
loop
  put("Zenbaki positibo bat idatzi");
  get(Zenb);
  exit when Zenb > 0;
end loop;
```

...

Azken adibidea: berreketa kalkulatu

Programa

```
with Ada.Text_IO,Ada.Integer_Text_IO;
use Ada.Text_IO,Ada.Integer_Text_IO;
procedure berreketa_kalkulatu is
    Zen1,Zen2,Akum,Kont:Integer;
begin
    get(Zen1);--datuak jaso
    get(Zen2);
    if Zen1=0 and Zen2=0 then -- kasu kritikoa
        put("Balio zehaztugabea");
    else -- kasu orokorrak
        Akum := 1;
        Kont := 0;
        loop exit when Kont=Zen2;
        Akum := Akum * Zen1;
        Kont := Kont + 1;
    end loop;
    put("Emaitza: ");
    put(Akum);

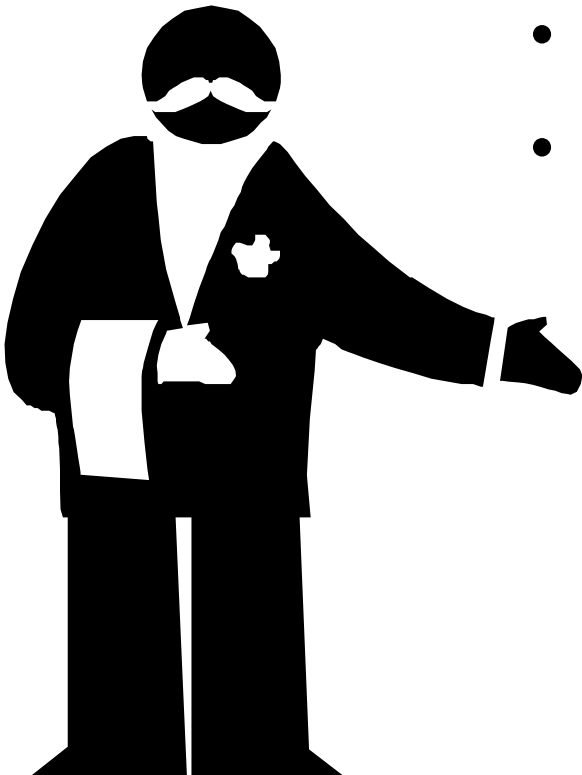
    end if;
end potentzia_kalkulatu ;
```

Algoritmoa

```
zen1,zen2,akum,kont: Integer;
irakurri(zen1);
irakurri(zen2);
baldin zen1=0 eta zen2=0 orduan
    idatzi("Balio zehaztugabea");
bestela
    akum <-- 1;
    kont <-- 0;
    errepikatu kont=zen2 bete arte;
        akum <-- akum * zen1;
        kont <-- kont + 1;
    amerrepikatu;
    idatzi(akum);
ambaldin;
```

Aurkibidea

- Gaiaren helburuak
- Garapen inguruneak
- Adaren oinarrizko gida
- **Pythonen oinarrizko gida**



Ezaugarri orokorrak

- Pythonek ez ditu aldagaien motak ez erazagutzen, ez kontrolatzen ere
- Letra larri eta xeheen artean bereizten du
- Kodearen tabulazioa sintaxiaren parte da
- Lerro bakoitzean ekintza bakarra egon daiteke

Programa baten oinarrizko egitura

```
def program_name () :  
    #Specification  
    Statement_1  
    ...  
    Statement_n;
```

Adibidea: Kaixo mundua

```
def kaixo_mundua()  
    #Sarrera:-  
    #Aurre: -  
    #Irteera: mezu bat  
    #Post: "kaixo mundua!" idatzi  
    print("kaixo mundua!")  
  
#erabiltzeko:  
kaixo_mundua()
```

Oharrak

- Almohadilatik (#) lerro bukaerara

```
# Proba programa
def proba():
    zen1 = 3 # Aldagaiak ez dira erazagutzen!
    zen2 = 2
    print("Gehiketaren balioa:")
    print(zen1 + zen2)

#Probari deia, exekutatu ahal izateko
proba()
```

Oinarrizko datu motak Pythonez

- int, long (zenbaki osokoak)
 - Eragiketak ==, !=, >, >=, <, <=, +, -, *, /, //, %, **
 - / zatiketa errealak, // zatiketa osokoa
- float (zenbaki errealak)
 - Eragiketak ==, !=, >, >=, <, <=, +, -, *, /, **
- str (ez da karaktere mota existitzen, kateak dira)
 - Eragiketak ==, !=, >, >=, <, <=, + (kateaketa)
- bool (boolearrak)
 - Eragiketak and, or, not

Aldagaien erazagupena

- Pythonen ez dira aldagaien motak kontrolatzen
 - Ez dira aldagaiak erazagutzen
 - Ez zaie datu mota bat esplizituki esleitzen
 - Exekuzioan zehar, datu mota alda daiteke
 - Ez dago konstanteak adierazteko modu espliziturik



Esleipena

- = bidez adierazten da
 - aldagaia = 1
 - aldagaia = 3.0
 - aldagaia = 'a'
 - aldagaia = “a”
 - aldagaia = True

Komila bikoitzak edo soilak erabil daitezke, alderik egon gabe. Koherentzia mantendu behar da ireki eta ixteko. Horrela komilak habiara daitezke, adibidez:

```
print("Lehenik 'Kaixo mundua!' esango dut.")
```



Datuen sarrera

- input() eragiketak karaktereak irakurtzen ditu, eta zenbaitetan zenbaki bilakatu beharko ditugu
 - `adina = int(input("Sartu zure adina: "))`
 - `nota = float(input("Sartu lortutako nota: "))`
 - `izena = input("Sartu zure izena: ")`

Datuen irteera

- `print()` eragiketak pantailatik idazten du, eta defektuz lerro saltoa jartzen du
 - Lerro saltoa ekiditeko *end* parametroa erabili behar da → `print('Kaixo',end=' ')`
 - `print("Kaixo mundua")`
 - `print('honako izena dauka: ' + izena)`
 - `print("ta adina: " + str(adina))`
 - `print(adina)`

Baldintzazko egiturak

- *if baldintza:*
 ekintza(k)
- *if baldintza :*
 ekintza(k)
else:
 ekintza(k)
- *if baldintza:*
 ekintza(k)
elif baldintza:
 ekintza(k)
 ...
 [*else:*
 ekintza(k)]

Adibideak

...

```
if zen < 0:
    print("Zenbakia negatiboa da")
```

...

```
if zen rem 2 == 0:
    print("Zenbakia bikoitia da")
else:
    print("Zenbakia bakoitia da")
```

...

...

```
if zen < 0:
    print("Zenbakia negatiboa da")
elif zen == 0:
    print("Zenbakia zero da")
else:
    print("Zenbakia positiboa da")
```

...

```
if zen != 0:
    print("Zenbakia ez da zero")
    if zen 0:
        print("Zenbakia positiboa da")
    else:
        print("Zenbakia negatiboa da")
else:
    print("Zenbakia zero da")
```

Iteraziozko egiturak

- while *baldintza* :
 ekintzak(k)
- for *aldagaia* in range(n1,n2):
 ekintza(k)
- for *aldagaia* in range(n2,n1,-1):
 ekintza(k)

Tarteak (range)

- Pythonen tarteak funtzionamendu berezia du
 - `range(hasiera, bukaera, saltoa)`
 - Hasiera barne dago, baino ez bukaera
 - saltoa hautazkoa da, eta defektuz 1 balio du
 - hasiera ere hautazkoa da, eta defektuz 0 balio du

Adi, galdera

- Zer inprimatuko du hurrengo adibideetako bakoitzak?



```
zen = 20
...
i = 1
while i < zen :
    print(i)
    i = i + 1
...
for i in range (1, zen) :
    print(i)
...
for i in range (zen, 0, -1) :
    print(i)
...
for i in range (num) :
    print(i)
```

Azken adibidea: berreketa kalkulatu

Algoritmoa

```
zen1,zen2,akum,kont: Integer;
irakurri(zen1);
irakurri(zen2);
baldin zen1=0 eta zen2=0 orduan
    idatzi("Balio zehaztugabea");
bestela
    akum <-- 1;
    kont <-- 0;
    errepikatu kont=zen2 bete arte;
        akum <-- akum * zen1;
        kont <-- kont + 1;
    amerrepikatu;
    idatzi(akum);
ambaldin;
```

Programa

```
def berreketa_kalkulatu
    # datuak jaso
    zen1 = int(input())
    zen2 = int(input())
    if zen1 == 0 and zen2 == 0:
        # kasu kritikoa
        print("Balio zehaztugabea")
    else:
        # kasu orokorrak
        akum = 1
        kont = 0
        while kont < zen2:
            akum = akum * zen1
            kont = kont + 1
        print("Emaitza: "+str(akum))
```


3.1 Gaia. Beheranzko diseinua: azpiprogramak

Aurkibidea

- Gaiaren helburuak
- Motibazioa
- **Kontzeptu orokorrak**
- Azpiprogramak Adaz
- Azpiprogramak Pythonez
- Ohitura onak

Datuen elkartrukaketa

- Azpiprogramek datuak jaso ditzakete beraien ataza egin ahal izateko edota emaitzak itzuli ahal izateko.
 - Parametroen bitartez eta *return* ekintzaren bitartez

```
...
```

```
n <-- ... ;
```

```
fakt_n <-- faktoriala(n);
```

```
idatzi(n "-ren faktoriala: " fakt_n);
```

```
...
```

Berrerabilpena

- Azpiprograma baten definizioa orokorra da
 - Ondorio berdina lor daiteke datuen balio ezberdinetarako (aldagai edo adierazpenak)

```
...  
n <-- ... ;  
k <-- ... ;  
fakt_n <-- faktoriala(n) ;  
fakt_k <-- faktoriala(k) ;  
fakt_nk <-- faktoriala(n - k) ;  
konb <-- fakt_n / (fakt_k * fakt_nk) ;  
idatzi("Eraitza: " konb) ;  
...
```

Parametro formalak

- Azpiprograma baten parametroen izenak, bere burukoan adierazten diren moduan
 - Azpiprograma barruan bakarrik existitzen dira

Parametro errealak

- Parametro formalei esleitzen zaien balioak (aldagaiak, adierazpenak...) azpiprogramari deitzen zaionean
 - Dei ezberdinak == parametro erreal ezberdinak
 - Dei bakoitzean parametro errealen kopurua eta horien motak bat egin behar dute

Azpiprograma bati deia

- A (azpi)programa batek, beste B azpiprograma bati deitzen badio
 - Aren egikaritzea gelditzen da, eta Bri ematen dio kontrola
 - Parametro errealen eta formalen arteko harremana ezartzen da
 - Bren aldagaienezat memoria erreserbatzen da
 - Bren egikaritzea bukatzen denean, Ak emaitza jasotzen du eta egikaritzea jarraitzen du

return ekintza

- Azpiprograma baten emaitzak itzultzeko erabiltzen da
 - Emaitza gisa zein adierazpen ebatziko den adierazten du
- *return* egikaritzen denean azpiprograma bukatzen da

```
azpiprograma konbinatoria(n, k : Integer)  
return Integer
```

```
(azpi)programa nagusia
```

```
...
```

```
a1 <-- ... ;
```

```
a2 <-- ... ;
```

```
ema <-- konbinatoria(a1,a2);
```

```
...
```

```
    fakt_n, fakt_k, fakt_nk : Integer;  
hasiera
```

```
    fakt_n <-- faktoriala(n);
```

```
    fakt_k <-- faktoriala(k);
```

```
    fakt_nk <-- faktoriala(n-k);
```

```
    return fakt_n / (fakt_k * fakt_nk);
```

```
amaiera konbinatoria;
```


Aldagaien ikusgarritasuna

- Aldagaiak, definituta dauden azpiprogramaren gorputzean baino ez dira ikusgarriak
 - Ondorioz, ez dira existitzen eta ezin dira beste batzuetan erabili
 - Salbuespenak (aldagai globalak...) ez ditugu oraindik ikusiko

```
azpiprograma konbinatoria(n, k : Integer) return  
Integer
```

```
(azpi)programa nagusia  
  a1,a2:Integer;  
hasiera  
  ...  
  a1 <-- ... ;  
  a2 <-- ... ;  
  fakt_n <-- ...; -- ERROREA  
  ...  
amaiera nagusia;
```

```
      fakt_n, fakt_k, fakt_nk : Integer;  
hasiera  
      fakt_n <-- faktoriala(a1); -- ERROREA  
      ...  
amaiera konbinatoria;
```

Aldagaien ikusgarritasuna

- Azpiprogramaren egikaritzea bukatzean, definitutako aldagai guztiak desagertzen dira (existitzeari uzten diote, memoria hori askatzen da)
 - Azpiprogramari berriz deitzen bazaio, aldagai berriak sortuko dira

Azpiprograma bati deia

- Oso garrantzitsua da azpiprogramen espezifikazioa
 - Espezifikazioaren irakurketarekin soilik, beste programatzaile batek azpiprogramak egiten duena ulertu beharko du
- Espezifikazioan ez dira parametroak edota euren motak adierazten, soilik euren ezaugarriak

Adibideak

```
azpiprograma asteriskoak_idatzi (Zenbat : Integer)
```

```
    ---Aurrebaldintza: Zenbat >=0
```

```
    ---Postbaldintza: Zenbat asterisko sinbolo idatzi dira
```

```
azpiprograma letra_da (Kar : Character) return Boolean
```

```
    ---Aurrebaldintza:
```

```
    ---Postbaldintza: Kar letra bat bada, True itzultzen du.
```

```
False bestela
```

Aurkibidea

- Gaiaren helburuak
- Motibazioa
- Kontzeptu orokorrak
- **Azpiprogramak Adaz**
- Azpiprogramak Pythonez
- Ohitura onak

Parametro motak Adaz

- Hiru parametro mota definitzen dira, azpiprogramari datuak emateko edota bertatik jasotzeko erabiliko denaren arabera
 - Sarrera parametroak
 - Irteera parametroak
 - Sarrera-irteera parametroak

Sarrera parametroak

- Azpiprograma bati datuak emateko erabiltzen dira
 - Sarrerako parametro formalek parametro errealetatik jasotzen dituzte balioak
 - Azpiprograma egikaritzen den bitartean euren balioak ezin dira aldatu (konstante lokalak bailira)

Sarrera parametroak

- Adaz, **in** hitz erreserbatua erabiltzen da (hautazkoa) parametro bat edo batzuk sarrerakoak direla adierazteko
 - Baliokideak dira
 - azpiprograma idatzi_asteriskoak(Zenbat : Integer)
 - azpiprograma idatzi_asteriskoak(Zenbat : in Integer)

Parámetro errealak (*in*):
Parametro formaleen
mota berdineko
adierazpenak

```
azpiprograma asteriskoak_idatzi(Zenbat : in Integer)
...
begin
  for I in 1..Zenbat loop
    put("*");
  end loop;
end asteriskoak_idatzi;
```

Deien adibideak:

```
asteriskoak_idatzi(7);
asteriskoak_idatzi(2*N+4);
```


Irteera parametroak

- Azpiprograma batetatik balioak itzultzeko erabiltzen dira
 - Irteerako parametro formal bakoitzak, azpiprogramako aldagai gisa erabiltzen da
 - *aldagai* honek azpiprogramaren exekuzioa bukatzen denean duen balioa esleituko zaio parametro errealari
 - Irteerako parametro erreal bat beti izango da aldagai bat

Irteera parametroak

- Adaz, **out** hitz erreserbatua erabiltzen da parametro bat edo batzuk irteerakoak direla adierazteko

Parametro errealak (*in*):
Parametro formalen
mota berdineko
adierazpenak

```
azpiprograma zatiketa_moztua  
    (Zatikizuna, Zatitzailea: in Integer;  
     Zatiketa, Hondarra: out Integer)
```

```
-- Azpiprograma honek bi datu jasotzen  
-- ditu eta bi itzultzen ditu  
...
```

Deien adibideak:

```
zatiketa_moztua (17, 7, N1, N2);  
zatiketa_moztua (X, Y,  Zat, Ema);
```

```
begin  
    Zatiketa := Zatikizuna / Zatitzailea;  
    Hondarra := Zatikizuna rem Zatitzailea;  
end zatiketa_moztua;
```

Parametro errealak
(*out*): Parametro
formalen mota berdineko
aldagaiak

Sarrera-irteera parametroak

- Aldagaien balioak eguneratzeko erabiltzen dira
 - Parametro errealaen balioa hartzen da eta aldatzen da
 - Irteera parametroen moduan funtzionatzen dute, baina kasu honetan hasierako balio bat izango dute (sarrerakoa)

Sarrera-irteera parametroak

- Adaz, **in out** hitz erreserbatuak erabiltzen dira parametro bat edo batzuk sarrera-irteerakoak direla adierazteko

```
azpiprograma ainkrementatu(Kont : in out Integer)
...
begin
    Kont := Kont + 1;
end inkrementatu;
```

Parámetro errealak (*in out*):
Parametro formalen mota
berdineko **aldagaiak**

Deien adibideak:

```
inkrementatu(N);
inkrementatu(Kont);
```

Bigarren adibidean, parametro errealaren izena parametro formalaren berdina da (Kont).

Arazo bat izan daiteke?

Azpiprograma motak Adaz

- Adak bi azpiprograma moten arteko bereizketa esplizitua egiten du
 - Funtzioak
 - Prozedurak

Funtzioak

- Adierazpen baten edo eragiketa baten parekoak dira
 - Parametro guztiak sarrerakoak dira
 - Eraitza bakar bat kalkulatu dute eta *return* ekintzaren bidez itzultzen dute
 - Exekuzioan, itzultzen duten balioetatik ordezkatuak dira

Adibideak (burukoak)

```
function letra_da (Kar : in Character) return Boolean is ...
```

Izena

Parametro
formalak

Emitza
mota

```
function balio_absolutua (N : in Integer) return Integer  
is ...
```

Adibideak (gorputzak)

```
function letra_da (Kar : in Character) return Boolean is
```

```
    Emaizta : Boolean := False;
```

```
begin
```

```
    if (Kar >= 'A' and Kar <= 'Z') or (Kar >= 'a' and Kar <= 'z') then
```

```
        Emaizta := True;
```

```
    end if;
```

```
    return Emaizta;
```

```
end letra_da;
```

Dei adibidea:

```
    get(Iniziala);
```

```
    if letra_da(Iniziala) then
```

```
        ...
```

```
function balio_absolutua (N : in Integer) return Integer is
```

```
    Emaizta : Integer := N;
```

```
begin
```

```
    if (Emaizta < 0) then
```

```
        Emaizta := Emaizta * -1;
```

```
    end if;
```

```
    return Emaizta;
```

```
end balio_absolutua;
```

Dei adibidea:

```
    N := ...;
```

```
    B := balio_absolutua(N);
```

```
    put(balio_absolutua(N-1));
```


Garrantzitsua

- Funtzioen parametroak sarrerakoak dira (*in*)
 - Beraz, konstanteak bezelakoak dira, eta ezin zaie balioa aldatu

```
function proba(Zenbakia: in Integer) return Boolean is
```

```
    Lag : Integer := Zenbakia;
```

```
begin
```

```
    Lag := Lag - 1;
```

```
    ...
```

```
    Zenbakia := Zenbakia - 1; -- ERROREA!!!
```

```
    ...
```

Prozedurak

- Ekintza oso baten parekoak dira
 - Eraitza bat, batzuk edo bat ere ez itzul ditzake
 - Sarrerako, irteerako edota sarrera-irteerako parametroak izan ditzake
 - Ez dute inongo *return* ekintzarik gauzatzen
 - Zerbait itzuliz gero, irteerako parametroen bidez egiten dute

Adibideak (burukoak)

```
procedure agurra_inprimatu () is ...
```

Izena

Parametro
formalak

```
procedure zatiketa_moztua (Zatikizuna, Zatitzailea : in Integer;  
                           Zatiketa, Hondarra: out Integer) is ...
```

Dei adibideak:

```
agurra_inprimatu();  
N1 : = .....;  
zatiketa_moztua(N1,2,Ema,Hondarra);  
asteriskoak_idatzi(Ema);
```

Adibidea: bi zenbaki ordenatu

```
procedure ordenatu() is
    Zenbaki1, Zenbaki2: Integer;
begin
    put("Zenbaki osoko bat idatzi: ");
    get(Zenbaki1);
    put("Beste zenbaki osoko bat idatzi: ");
    get(Zenbaki2);
    ordenatu1(Zenbaki1,Zenbaki2);
    if balioztatu(Zenbaki1,Zenbaki2) then
        put(Zenbaki1);
        put(Zenbaki2);
    else
        ordenatu2(Zenbaki1,Zenbaki2);
        if balioztatu(Zenbaki1,Zenbaki2) then
            put(Zenbaki1);
            put(Zenbaki2);
        else
            put("Ezin izan dira zenbakiak ordenatu");
        end if;
    end if;
end ordenatu;
```

Adibidea: bi zenbaki ordenatu

```
procedure ordenatu1(N1,N2: in out Integer) is
    -- Aurre: 2 zenbaki osoko
    -- Post: zenbaki osokoak ordenatzen ditu (mota 1)
begin
    put("ORDENATU1");
    new_line;
    if N2 < N1 then
        N1 := N1 + N2;
        N2 := N1 - N2;
        N1 := N1 - N2;
    end if;
end ordenatu1;
```

Adibidea: bi zenbaki ordenatu

```
procedure ordenatu2(N1,N2: in out Integer) is
    -- Aurre: 2 zenbaki osoko
    -- Post: zenbaki osokoak ordenatzen ditu (mota 2)
    Lag : Integer;
begin
    put("ORDENATU2");
    new_line;
    if N2 < N1 then
        Lag := N1;
        N1 := N2;
        N2 := Lag;
    end if;
end ordenatu2;
```

Adibidea: bi zenbaki ordenatu

```
function balioztatu(N1,N2: in Integer) return Boolean is  
    -- Aurre: 2 zenbaki osoko  
    -- Post: True itzultzen du N1 <= N2 bada, bestela False  
  
    Emaidza : Boolean := True;  
begin  
    if N1 > N2 then  
        Emaidza := False;  
    end if;  
    return Emaidza;  
end ordenatu1;
```

Aurkibidea

- Gaiaren helburuak
- Motibazioa
- Kontzeptu orokorrak
- Azpiprogramak Adaz
- **Aziprogramak Pythonez**
- Ohitura onak

Parametro motak Pythonez

- Datu motak parametro mota zehazten du
 - Aldaezina bada (osokoa, errealak, karakterea, string,...), orduan parametroa sarrerakoa izango da
 - Aldakorria bada (zerrendak,...), orduan sarrerakoa (ez bada aldatzen) edo sarrera-irteerakoa (balioa aldatzen bazaio) izango da

Azpiprogramak Pythonez

- Pythonen ez dira funtzioak eta prozedurak esplizituki ezberdintzen
- Burukoan ez da parametroen mota adierazten, ez eta itzulitako datuarena
 - Azpiprograma batek ez du zertan ***return*** ekintza izan
 - *return* batek emaitza bat baino gehiago itzul dezake

Adibidea: bi zenbaki ordenatu

```
def ordenatu1(n1,n2):  
    print("ORDENATU1:")  
    if n2 < n1:  
        n1 = n1 + n2  
        n2 = n1 - n2  
        n1 = n1 - n2  
    return n1,n2
```

```
def ordenatu2(n1,n2):  
    print("ORDENATU2:")  
    if n2 < n1:  
        lag = n1  
        n1 = n2  
        n2 = lag  
    return n1,n2
```

Adibidea: bi zenbaki ordenatu

```
def balioztatu(n1,n2):  
    ## Aurre: 2 zenbaki osoko  
    ## Post: True itzultzen du  $n1 \leq n2$  bada, bestela False  
    emaitza = True  
    if n1 > n2:  
        emaitza = False  
    return emaitza
```

Adibidea: bi zenbaki ordenatu

```
def ordenatu():
    zenbaki1 = int(input("Zenbaki osoko bat idatzi: "))
    zenbaki2 = int(input("Beste zenbaki osoko bat idatzi: "))
    zenbaki1, zenbaki2 = ordenatu1(zenbaki1, zenbaki2)
    if balioztatu(zenbaki1, zenbaki2):
        print(Zenbaki1)
        print(Zenbaki2)
    else:
        zenbaki1, zenbaki2 = ordenatu2(zenbaki1, zenbaki2)
        if balioztatu(zenbaki1, zenbaki2):
            print(zenbaki1)
            print(zenbaki2)
        else:
            print("Ezin izan dira zenbakiak ordenatu")
```

Aurkibidea

- Gaiaren helburuak
- Motibazioa
- Kontzeptu orokorrak
- Azpiprogramak Adaz
- Azpiprogramak Pythonez
- **Ohitura onak**

Izendatzea

- Aldagaiek eta azpiprogramek izen egokiak badituzte, inplementazioa *testu moduan* irakurriko da
 - Aldagaien erabilera deskribatzen duten izenak erabili
 - Azpiprograma baten helburua sinplea bada, izen egokia ematea erraza izango da

```
...  
zenbaki_positiboa_eskatu(adina);  
if adina > 18 then  
...
```

Espezifikazioa

- Garrantzitsua da azpiprogramak espezifikatzea eta anbiguotasuna ekiditea
 - Gainera, gomendagarria da oharrak erabiltzea
- Erraza da azpiprograma baten bertsioa beste batengatik ordezkatzeko, espezifikazioa aldatzen ez den bitartean
 - Bestela, beste programen kodea aldatu beharko litzateke

return ekintza

- Adako funtzioetan eta Pythoneko azpiprograma batzuk, momenturen batean *return* egikaritzea eskatzen dute
 - *return* ekintza bat baino gehiago egon daiteke, baina bide guztiek *return* ekintza batera eraman behar dute
 - Adierazpena emaitzaren mota berdinekoa izan behar da (Ada)
 - *return* ekintza ta geroko kodea ez da inoiz egikarituko

return ekintza

- Ohitura ona: ***return* bakarra** erabiliko da, eta azpiprogramaren azken ekintza izango da
 - Askoz argiagoa da, eta arazketa errazten du. Are gehiago, azpiprogramen kopurua eta tamaina handia denean

Parametroak aldatzea

- Adak argi uzten du ze parametro alda daitezkeen (*out*, *in out*) eta zeintzuk ez (*in*)
- Pythonen, parametroak aldatzeak espero ez diren emaitzak sor ditzake
 - Ezagutza *aurreratuak* behar dira (punteroak, ...)
 - Ondorioz, ikasgai honetan, aldatu beharrean, hainbat elementu itzuliko ditugu

```
zenbaki1, zenbaki2=ordenatu(zenbaki1, zenbaki2)
```

Begizta habiaratuak

- Orokorrean, ez da ohitura ona begizta bat bestearen barruan jartzea
 - Gehienetan, barruko begizta azpiprograma bati deia bezala ikus liteke
 - Salbuespenen artean, matrizeen korritzea dago

3.2 Gaia. Azpiprogrammen ariketak

1. Ariketa (ADA eta Python)

- Azpiprograma bat egin, erabiltzaileari zenbaki bat eskatzen diona eta zenbaki hori positiboa dela ziurtatzen duena. Ez bada, berriz eskatuko lioke, behin eta berriz, positiboa izan arte
 - Espezifikazioa

Sarrera: -

Aurre: -

Irteera: zenbaki osoko bat: zen

Post: zen: balioa $l > 0$

Soluzioa

```
with Ada.Text_IO, Ada.Integer_Text_IO;
use Ada.Text_IO, Ada.Integer_Text_IO;
procedure positiboa_eskatu(Zen: out Integer) is
begin
    loop
        put_line("0 baino handiagoko osoko bat sartu: ");
        get(Zen);
        if (Zen<=0) then
            put_line("Sartutako balioa ez da egokia");
        end if;
        exit when Zen >0;
    end loop;
end positiboa_eskatu;
```

Programa nagusia

```
with positiboa_eskatu, Ada.Text_IO, Ada.Integer_Text_IO;  
use Ada.Text_IO, Ada.Integer_Text_IO;
```

```
procedure positibo_nagusia is
```

```
    Zen: Integer;
```

```
begin
```

```
    put_line("Azpiprogramari deituko diogu...");
```

```
    positiboa_eskatu(Zen);
```

```
    new_line;
```

```
    put("Azpiprogramak itzulitako azpiprograma: ");
```

```
    put(Zen);
```

```
    new_line;
```

```
end positibo_nagusia;
```


Soluzioa Python

OHARRA: azpipograma honek sarrerako parametroa du, baina ez luke eduki beharko espezifikazioaren arabera.

```
def positiboa_eskatu(Zen):  
    print(str(Zen) + " baino handiagoko osoko bat sartu: ");  
    zenbakia=int(input());  
    while(zenbakia<=Zen):  
        print(str(Zen) + " baino handiagoko osoko bat sartu: ");  
        zenbakia=int(input());  
        if (zenbakia<=Zen):  
            print("Sartutako balioa ez da egokia, zenbaki  
positiboa izan behar du");  
    return zenbakia
```

Programa nagusia Python

```
def positibo_nagusia():  
    print("Azpiprogramari deituko diogu...");  
    zenb=positiboa_eskatu(0);  
    print("Azpiprogramak itzulitako azpiprograma: ");  
    print(zenb);
```

```
positibo_nagusia()
```

2. Ariketa (ADA eta Python)

- Azpiprograma bat egin, zenbaki bat eta posizio bat emanda, posizio horretako digitua itzuliko duena
 - Espezifikazioa:

Sarrera: Bi zenbaki osoko: zen eta pos

Aurre: zen > 0 eta pos >= 1 eta zen ez du 0 digiturik.

Irteera: zenbaki osoko bat: dig

Post: $0 \leq \text{dig} \leq 9$, non dig zen zenbakiaren pos posizioan dagoen digitua da, eskuinetik hasita.

Soluzioa

```
function digitua_posizioan (Zen: Integer; pos:
Integer) return Integer is
    Auxiliarra: Integer := Zen;
    Kontagailua: Integer := 1;
begin
    while (Kontagailua<pos) loop
        Auxiliarra:=Auxiliarra/10;
        Kontagailua:=Kontagailua+1;
    end loop;
    return(Auxiliarra rem 10);
end digitua_posizioan;
```

Proba programa

```
with digitua_posizioan, Ada.Text_Io, Ada.Integer_Text_Io;
use Ada.Text_Io, Ada.Integer_Text_Io;

procedure probak_digitua_posizioan is
    Zenbakia, Eraitza, Posizioa: Integer;
begin
    Zenbakia:= 1234;
    Posizioa:=2;
    Eraitza:= digitua_posizioan(Zenbakia, Posizioa);
    if (Eraitza = 0) then
        put("Posizioa digitu kopurua baina handiagoa da");
    else
        put(Zenbakia);
        put(" ren ");
        put(Posizioa);
        put(" posizioan dagoen digitua ");
        put(Eraitza);
        put(" da.");
    end if;
    new_line;

    -- Kasu gehiago frogatu beharko lirateke
end probak_digitua_posizioan;
```

Python soluzione

```
def digitua_posizioan(zen, pos):  
    posizioa=1  
    while(posizioa<pos):  
        zen=zen//10  
        posizioa=posizioa+1  
    return zen % 10
```

Python nagusia

```
def posizioan_nagusia():  
    print("Osoko bat sartu:");  
    zenbakia=int(input());  
    print("Osoko bat sartu posizioa adierazteko:");  
    posizioa=int(input());  
    print("Azpiprogramari deituko diogu...");  
    digitua=digitua_posizioan(zenbakia, posizioa);  
    print("Azpiprogramak itzultako azpiprograma:");  
    print(digitua);
```

posizioan_nagusia()

3. Ariketa (ADA eta Python)

- Azpiprograma bat egin erabiltzaileari bi zenbaki osoko eskatzen dizkiona eta horien zeinua aldatzen duena
 - Espezifikazioa

Sarrera: 2 zenbaki osoko: zenb1 eta zenb2

Aurre: zenb1: bal1 eta zenb2: bal2

Irteera: 2 zenbaki osoko: zenb1 eta zenb2

Post: zenb1: bal1*-1 eta zenb2: bal2*-1

1. bertsioa

```
with Ada.Text_IO, Ada.Integer_Text_IO;
use Ada.Text_IO, Ada.Integer_Text_IO;

procedure zeinu_aldaketa_nagusia_v1 is
  Zen1, Zen2: Integer;
  procedure zeinu_aldaketa(Zen1, Zen2: out Integer) is begin
    put_line("Sartu lehenengo zenbakia: ");
    get(Zen1);
    put_line("Sartu bigarren zenbakia: ");
    get(Zen2);
    put_line ("Sartutako zenbakiak: ");
    put(Zen1);
    put(Zen2);
    Zen1:= Zen1*(-1);
    Zen2:= Zen2*(-1);
  end zeinu_aldaketa;
begin -- zeinu_aldaketa_nagusia_v1 programa hasten da
  zeinu_aldaketa(Zen1,Zen2);
  put_line ("Zeinua aldatu ostean, zenbakiak dira: ");
  put(Zen1);
  put(Zen2);
end zeinu_aldaketa_nagusia_v1;
```

2. bertsioa

```
with Ada.Text_IO, Ada.Integer_Text_IO;
use Ada.Text_IO, Ada.Integer_Text_IO;

procedure zeinu_aldaketa_nagusia_v2 is
  Zen1, Zen2: Integer;
  procedure zeinu_aldaketa(Zen1, Zen2: in out Integer) is begin
    Zen1:= Zen1*(-1);
    Zen2:= Zen2*(-1);
  end zeinu_aldaketa;
begin  -- zeinu_aldaketa_nagusia_v1 programa hasten da
  put_line("Sartu lehenengo zenbakia: ");
  get(Zen1);
  put_line("Sartu bigarren zenbakia: ");
  get(Zen2);
  put_line ("Sartutako zenbakiak: ");
  put(Zen1);
  put(Zen2);
  zeinu_aldaketa(Zen1,Zen2);
  put_line ("Zeinua aldatu ostean, zenbakiak dira: ");
  put(Zen1);
  put(Zen2);
end zeinu_aldaketa_nagusia_v2;
```

Python soluzioa

```
def zeinua_aldatu(zenb1, zenb2):  
    return zenb1*-1, zenb2*-1
```

```
def zeinua_nagusia():  
    print("Osoko bat sartu: ");  
    zenbakia1=int(input());  
    print("Beste osoko bat sartu: ");  
    zenbakia2=int(input());  
    print("Osoko bat sartu posizioa adierazteko: ");  
    print("Azpiprogramari deituko diogu...");  
    zenb1, zenb2=zeinua_aldatu(zenbakia1,zenbakia2);  
    print("Azpiprogramak itzulitako azpiprograma: ");  
    print(str(zenb1) + " eta " + str(zenb2));
```

```
zeinua_nagusia()
```

4. Ariketa (ADA eta Python)

- Azpiprograma bat egin, zeinak 2. mailako ekuazio baten bi erroen balioa itzultzen duen ($ax^2 + bx + c = 0$), a, b eta c balioak emanda
 - Espezifikazioa

Sarrera: 3 zenbaki erreal: a, b eta c

Aurre: a-ren balioa ez da 0

Irteera: 2 zenbaki erreal: x1 eta x2

Post: x1: $(-b + \sqrt{b^2 - 4ac})/2a$, x2: $(-b - \sqrt{b^2 - 4ac})/2a$

4.1 Gaia. Bektoreak (edo arrayak) eta matrizeak

Aurkibidea

- Gaiaren helburuak
- Motibazioa
- **Sekuentziak Adaz**
- Sekuentziak Pythonez
- Matriziak Adaz
- Matriziak Pythonez

Sekuentziak Adaz

- *Array* datu motaren bidez inplementatzen dira
 - Memorian tokia erreserbatzen da, sekuentziako ondoz ondoko elementuentzat, eta elementu guztiak mota berdinekoak dira

```
NA: constant Integer := ...;
```

```
type T_adinak is array (1..NA) of Integer;
```

Sekuentziak Adaz

- Zenbat aldagai behar dira OP ikasgaian matrikulatutako ikasle guztien adinak gordetzeko?

```
Adinak: T_adinak;
```


Sekuentziak Adaz

- Zenbat kode-lerro behar dira adin guztiak 0 balioarekin hasieratzeko?
 - Soluzio posibleak:

```
for Ind in 1..NA loop  
    Adinak(Ind) := 0;  
end loop;
```

```
for Ind in Adinak'first..Adinak'last loop  
    Adinak(Ind) := 0;  
end loop;
```

```
Adinak := (others => 0);
```

Sekuentziak Adaz

- Zenbat parametro jasotzen ditu, matrikulatutako ikasle gazteenaren adina itzultzen duen azpiprogramak?
 - Soluzioa:

```
function gazteena(Adinen_Bektorea: in T_adinak) return Integer is  
    ...
```

Beste adibide bat

```
package datuak is
    N: constant Integer := 12;
    type T_sekuentzia is array(1..N) of Integer
end datuak;
```

Algoritmoa

```
Elem: Integer;
Seku: 12 Integer;
hasieran_kokatu(Seku);
errepikatu kanpoan(Seku) bete arte;
    Elem <-- uneko_elementua(Seku);
    ...
    aurreratu(Seku);
amerrepikatu;
```

ADA

```
Elem, Ind: Integer;
Seku: T_sekuentzia;
Ind := 1;
loop exit when Ind > Seku'last;
    Elem := Seku(Ind);
    ...
    Ind := Ind+1;
end loop;
```

Array bat definitzeko bi modu

- Tamaina definitzitik mugatzen
 - type T_adinak is array(1..100) of Integer;
 - Adinak: T_adinak;
- Mugatu gabe (erazagupenean mugatzen da)
 - type T_adinak is array(Integer range <>) of Integer;
 - Adinak1: T_adinak(1..100);
 - Adinak2: T_adinak(1..50);

Hemendik aurrera

- Pakete batean lau datu mota definitu ditzakegu eta horrela gure programetatik erabili
 - with bektoreak; use bektoreak;

```
package bektoreak is

  type Osokoen_Bektorea is array (Integer range <>) of Integer;

  type Errealen_Bektorea is array (Integer range <>) of Float;

  type Boolearren_Bektorea is array (Integer range <>) of Boolean;

  type Karaktereen_Bektorea is array (Integer range <>) of Character;

end bektoreak;
```

Ariketak

- Datu mota bat emanda
 - type T_ikasleen_NANak is array (1..100) of Integer;
- Azpiprogramak idatzi
 - function/procedure?? irakurri_sekuentzia (.....)..... is
 - function/procedure?? sekuentziako_handiena (.....) is
 - function/procedure?? sekuentzia_inprimatu (.....) is

Ariketak

- Datu motaren definizioa emanda
 - type T_hilabeteak is array (1..12) of string(1..10);
 - edo bere baliokidea:
 - subtype T_hilabete_izena is string(1..10);
 - type T_hilabeteak is array(1..12) of T_hilabete_izena;
- Idatzi hilabete_izena_lortu azpiprograma

```
sarrera: osoko bat
Aurre: 1 <= hilabetea:balioa1 <=12
Irteera: izen bat
Post: hilabeteari dagokion izena, 1: Urtarrila, 2: Otsaila, etab.
```

Ariketak

- Datu mota baten definizioa emanda
 - type Osokoen_Bektorea is array (1..100) of Integer;
- ezkerrera_mugitu azpiprograma idatzi, zeinak Osokoen_Bektorea jasotzen duen, eta bere elementuak posizio bat ezkerrera mugitzen dituen

13 7 1 3 9 12 23 5 8 3

7 1 3 9 12 23 5 8 3 13

Aurkibidea

- Gaiaren helburuak
- Motibazioa
- Sekuentziak Adaz
- **Sekuentziak Pythonez**
- Matriziak Adaz
- Matriziak Pythonez

Sekuentziak Pythonez

- Hainbat modu daude inplementatzeko. Adibidez, *list* datu mota erabiliz
 - Ez dira erazagutzen, zuzenean balioen sekuentzia bat esleitzen zaie

```
bektore1 = [1, 7, 11, 23, 4]
```

```
bektore2 = ['a', 'e', 'i', 'o', 'u']
```

Sekuentziak Pythonez

- Posizioak beti 0tik *len()-1*-era doaz
 - Aurredefinitutako *len()* azpiprogramak zerrenda baten elementu kopurua itzultzen du

```
for i in range(0,len(bektore1)):  
    print (bektore1[i])
```

**Gogoratu, range
funtzioak hasierako
elementua jasotzen
duela, baina ez
azkenekoa**

0	1	2	3	4
1	7	11	23	4

Adibidea

- Sekuentzia baten balio maximoa

```
def maximoa(bek):  
    emaitza = bek[0]  
    for i in range(1, len(bek)):  
        if emaitza < bek[i]:  
            emaitza = bek[i]  
    return emaitza
```

```
def nagusia():  
    zerrenda = [1, 7, 11, 23, 4]  
    print(maximoa(zerrenda))
```

Aurkibidea

- Gaiaren helburuak
- Motibazioa
- Sekuentziak Adaz
- Sekuentziak Pythonez
- **Matrizeak Adaz**
- Matrizeak Pythonez

Matrizeak Adaz

- Bi dimentsiotako array moduan defini daitezke
 - Dimentsio gehiagotara orokortu daiteke
- Array-en array moduan ere defini daitezke

Bi dimentsiotako arraya

- Definizioetik bere tamaina zehazten
 - type M_adinak is array(1..4, 1..7) of Integer;
 - Adinak: M_adinak;

	1	2	3	4	5	6	7
1	17	19	18	18	19	17	19
2	21	18	17	19	18	18	17
3	18	20	18	21	18	22	18
4	20	18	18	19	19	20	21

Bi dimentsiotako arraya

- Tamaina mugatu gabe (erazagupenean mugatzen da)
 - type M_adinak is array(Integer range <>, Integer range <>) of Integer;
 - Adinak1: M_adinak(1..100, 1..30);
 - Adinak2: M_adinak(1..4, 1..7);

Arrayen arraya

- type T_adinen_errenkada is array (1..7) of Integer;
- type M_adinak is array(1..4) of T_adinen_errenkada;
 - Adinak: M_adinak;

	1	2	3	4	5	6	7
1	17	19	18	18	19	17	19
2	21	18	17	19	18	18	17
3	18	20	18	21	18	22	18
4	20	18	18	19	19	20	21

Lehen bezala

- Pakete batean lau datu mota berri defini ditzakegu, gure programetan erabili ahal izateko

```
package matriziak is

  type Osokoen_Matrizea is array (Integer range <>, Integer range <>) of Integer;

  type Errealen_Matrizea is array (Integer range <>, Integer range <>) of Float;

  type Boolearren_Matrizea is array (Integer range <>, Integer range <>) of Boolean;

  type Karakterren_Matrizea is array (Integer range <>, Integer range <>) of Character;

end matriziak;
```

Elementuen atzipena

- Bi indize erabiliko dira, bata errenkadari erreferentzia egiteko eta bestea zutabeari

```
Pantaila: Karaktereen_Matrizea(1..24, 1..80);  
...  
Pantaila(6,25) := 'F';  
-- 6. errenkadako eta 25 zutabeko karakterea F izango da
```

```
type T_errenkada_boolak is array (1..10) of Boolean;  
type M_boolearrak is array (1..10) of T_errenkada_boolak;  
...  
Itsasontziak: M_boolearrak;  
...  
Itsasontziak(I) (J+1) := True;  
-- i errenkada eta j+1 zutabeko elementua True izango da
```

Tarteak: bi dimentsiotako arraya

- Esplizituki adierazi behar da zein dimentsiori egiten dioten erreferentzia 'first eta 'last adierazpenek

```
package datuak is
  type Osokoen_Matrizea is array(Integer range <>, Integer range <>) of Integer;
end datuak;
```

```
procedure pantailan_idatzi (M: in Osokoen_Matrizea) is
begin
  for errenkada in M'first(1)..M'last(1) loop
    for zutabea in M'first(2)..M'last(2) loop
      put(M(errenkada, zutabea));
    end loop
  end loop;
end pantailan_idatzi;
```

Tarteak: arrayen arraya

- Kasu honetan, 'first eta 'last dimentsio bakarreko arrayei aplikatzen zaie, beraz ez da bereizketarik egin behar

```
package datuak is
  type T_osokoen_errenkada is array (1..10) of Integer;
  type M_osokoak is array (1..10) of T_osokoen_errenkada;
end datuak;
```

```
procedure pantailan_idatzi (M: in M_osokoak) is
begin
  for errenkada in M'first..M'last loop
    for zutabe_elem in M(errenkada)'first..M(errenkada)'last loop
      put(M(errenkada)(zutabe_elem));
    end loop
  end loop;
end pantailan_idatzi;
```

Aurkibidea

- Gaiaren helburuak
- Motibazioa
- Sekuentziak Adaz
- Sekuentziak Pythonez
- Matriziak Adaz
- **Matriziak Pythonez**

Matrizeak Pythonez

- Zerrenden zerrenda moduan definitzen ditugu

```
zutabeak = [None]*7  
M_adinak = [zutabeak]*4  
...  
M_adinak[0]=[17,19,18,18,19,17,19]  
M_adinak[1]=[21,18,17,19,18,18,17]  
M_adinak[2]=[18,20,18,21,18,22,18]  
M_adinak[3]=[20,18,18,19,19,20,21]
```

	0	1	2	3	4	5	6
0	17	19	18	18	19	17	19
1	21	18	17	19	18	18	17
2	18	20	18	21	18	22	18
3	20	18	18	19	19	20	21

Adibidea

- Matrizen baten elementuak inprimatu

```
def inprimatu(Matrizea):  
  
    errenkadaKop = len(Matrizea)  
    zutabeKop = len(Matrizea[0])  
  
    for posE in range(0,errenkadaKop):  
        for posK in range(0,zutabeKop):  
            print(Matrizea[posE][posK], end=' ')  
        print() #lerro saltoa
```

```
def principal():  
    zutabeak=[None]*9  
    M=[zutabeak]*5  
    M[0]=[1,2,3,4,5,6,7,8,9]  
    M[1]=[11,12,13,14,15,16,17,18,19]  
    M[2]=[21,22,23,24,25,26,27,28,29]  
    M[3]=[31,32,33,34,35,36,37,38,39]  
    M[4]=[41,42,43,44,45,46,47,48,49]  
    inprimatu(M)
```


4.1 gaiko ariketak

Ariketak

- Datu mota bat emanda
 - type T_ikasleen_NANak is array (1..100) of Integer;
- Azpiprogramak idatzi
 - function/procedure?? irakurri_sekuentzia (.....)..... is
 - function/procedure?? sekuentziako_handiena (.....) is
 - function/procedure?? sekuentzia_inprimatu (.....) is

Motak.ads

```
package motak is
  type T_ikasleen_NANak is array (1..100) of Integer;
end motak;
```

Irakurri_sekuentzia.adb

```
with motak, ada.text_io, ada.integer_text_io;
use motak, ada.text_io, ada.integer_text_io;

procedure irakurri_sekuentzia(sek: in out T_ikaskleen_NANak) is
    zenb:integer:=0;
    ind:integer:=1;
begin
    put("sartu zenbakiak, -1 amaitzeko");
    new_line;
    while(zenb/=-1) loop
        put("sartu &integer'image(ind)&" zenbakia:");
        get(zenb);
        sek(ind):=zenb;
        ind:=ind+1;
    end loop;
end irakurri_sekuentzia;
```

Sekuentzia_inprimatu.adb

```
with motak;  
use motak;  
with ada.integer_text_io; use ada.integer_text_io;  
  
procedure sekuentzia_inprimatu (sek: in T_ikasteen_NANak) is  
  i:integer:=1;  
begin  
  while (sek(i)/=-1) loop  
    put(sek(i),4);  
    i:=i+1;  
  end loop;  
end sekuentzia_inprimatu;
```

Sekuentzia_handiagoa.adb

```
with motak;
```

```
use motak;
```

```
function sekuentziako_handiena (Bek: in T_ikasteen_NANak) return integer is  
    handi:integer:=integer'first;  
    i:integer:=1;
```

```
begin
```

```
    while (Bek(i)/=-1) loop  
        if (handi<Bek(i)) then  
            handi:=Bek(i);  
        end if;
```

```
    i:=i+1;
```

```
    end loop;
```

```
    return (handi);
```

```
end sekuentziako_handiena;
```

Nagusia.adb

```
with motak;  
use motak;  
with irakurri_sekuentzia, sekuentzia_inprimatu, sekuentziako_handiena;  
with ada.integer_text_io; use ada.integer_text_io;  
with ada.text_io; use ada.text_io;  
  
procedure nagusia is  
    sekuentzia: T_ikasleen_NANak;  
    zenb_handi:integer;  
  
begin  
    irakurri_sekuentzia(sekuentzia);  
    zenb_handi:=sekuentziako_handiena(sekuentzia);  
    put("Zenbaki handiena: "&integer'Image(zenb_handi));  
    new_line;  
    sekuentzia_inprimatu(sekuentzia);  
end nagusia;
```

Ariketak

- Datu motaren definizioa emanda
 - type T_hilabeteak is array (1..12) of string(1..10);
 - edo bere baliokidea:
 - subtype T_hilabete_izena is string(1..10);
 - type T_hilabeteak is array(1..12) of T_hilabete_izena;
- Idatzi hilabete_izena_lortu azpiprograma

```
sarrera: osoko bat
Aurre: 1 <= hilabetea:balioa1 <=12
Irteera: izen bat
Post: hilabeteari dagokion izena, 1: Urtarrila, 2: Otsaila, etab.
```


Ariketak

- Datu mota baten definizioa emanda
 - type Osokoen_Bektorea is array (1..100) of Integer;
- ezkerrera_mugitu azpiprograma idatzi, zeinak Osokoen_Bektorea jasotzen duen, eta bere elementuak posizio bat ezkerrera mugitzen dituen

1	2	3	4	5	6	7	8	9	10
13	7	1	3	9	12	23	5	8	3

1	2	3	4	5	6	7	8	9	10
7	1	3	9	12	23	5	8	3	13



4.2 Gaia. Datu egituren diseinua erregistroekin

Aurkibidea

- Gaiaren helburuak
- Motibazioa
- **Erregistroak**
- Habiaratutako egiturak

Erregistroa

- Datu mota egituratua da, zeinak mota berdineko zein ezberdineko datu multzo bat biltegiratzen duen
 - Motaren definizioan, erregistroa osatzen duen eremu edo balio bakoitzari datu mota bat esleitzen zaio.

```
type Ikaslea is record
    Esp_zen: Integer;
    Izena, Abizena: String(1..30);
    Ikasturtea: Integer;
    Taldea: Character;
end record;
```

Erregistro motako aldagaiak

- Gainerako aldagaietan ikusitakoaren arabera erazagutzen dira

```
Ikas1, Ikas2: Ikaslea;
```

	Ikas1
Esp_Zen	??
Izena	??
Abizena	??
Kurtsoa	??
Taldea	??

	Ikas2
Esp_Zen	??
Izena	??
Abizena	??
Kurtsoa	??
Taldea	??

Erregistroko eremuetara atzipena

```
get(Ikas1.Esp_Zen);  
get(Ikas1.Izena);  
Ikas1.Abizena := "Cousteau" -- 30 karaktere behar ditu!!!  
Ikas1.Talde := 'A';  
Ikas2.Kurtsoa := 1;  
Ikas1.Kurtsoa := Ikas2.Kurtsoa;
```

Ikas1	
Esp_Zen	1569
Izena	Iker
Abizena	Cousteau
Kurtsoa	1
Talde	'A'

Ikas2	
Esp_Zen	??
Izena	??
Abizena	??
Kurtsoa	1
Talde	??

Esleipena

```
Ikas2:= Ikas1;
```

	Ikas1
Esp_Zen	1569
Izena	Iker
Abizena	Cousteau
Kurtsoa	1
Taldea	'A'

	Ikas2
Esp_Zen	1569
Izena	Iker
Abizena	Cousteau
Kurtsoa	1
Taldea	'A'

Esleipena

```
Ikas2.Izena := "Jacques";  
Ikas2.Kurtsoa := 5;  
Ikas1 := Ikas2;
```

Ikas1	
Esp_Zen	1569
Izena	Jacques
Abizena	Cousteau
Kurtsoa	5
Talde	'A'

Ikas2	
Esp_Zen	1569
Izena	Jacques
Abizena	Cousteau
Kurtsoa	5
Talde	'A'

Erregistroekin eragiketak

- Erregistro motako aldagai baten eremu baten atzipenera eragiketa:
 - `Ikas1.Izena := Ikas2.Izena;`
 - `if (Ikas1.Kurtsoa = 2) then ...`
- Erregistro oso baten balioa esleitu mota berdineko aldagai bati:
 - `Ikas1 := Ikas2;`

Erregistroekin eragiketak

- Erregistro mota bereko balioen arteko konparaketa
 - $l_{kas1} = l_{kas2}$
 - $l_{kas1} \neq l_{kas2}$
- $<$, $>$, \leq eta \geq konparatzaileek ez dute zentzurik

Moten arteko komuztadura

```
type Ikaslea is record
  Esp_Zen: Integer;
  Izena, Abizena: String(1..30);
  Kurtsoa: Integer;
  Taldea: Character;
end record;
```

```
type Produktua is record
  Izena: Character;
  Prezioa: Integer;
  Izakinak: Integer;
end record;
```

```
-- Aldagaien erazagupena
Ikas1, Ikas2: Ikaslea;
ProdA, ProdB, ProdB: Produktua;

-- Esleipen posibleak
Ikas1 := Ikas2;
ProdA := ProdB;
Ikas1.Kurtsoa := ProdB.Izakinak;

-- Esleipen okerrak
Ikas1 := ProdB;
ProdA.Izakinak := Ikas1.Izena;
Ikas1.Izena := ProdB.Izena;
```

Erregistroen erabileraren abantailak

- Askoz aldagai eta parametro gutxiago definitzen dira
 - Ikasle1, Ikasle2, Ikasle3 vs
 - Zen_Esp1, Zen_Esp2, Zen_Esp3, Izena1, Izena2, Izena3, Abizena1, Abizena2, Abizena3, Kurtso1, Kurtso2, Kurtso3, Taldea1, Taldea2, Taldea3
- Gainera, diseinua eta inplementazioa errazten du
 - Irakurgarritasuna, eskalagarritasuna eta malgutasuna

Aurkibidea

- Gaiaren helburuak
- Motibazioa
- Erregistroak
- **Habiaratutako egiturak**

Erregistroak erregistroekin

- Erregistro baten eremuak erregistro motakoak izan daitezke

```
type T_pertsona is record  
  Identif: Integer;  
  Izena, Abizena: String(1..20);  
end record;
```

```
type T_bikotea is record  
  Pertsona1, Pertsona2: T_pertsona;  
  Helbidea: String(1..30);  
end record;
```

Erregistroak erregistroekin

```
Bikotea: T_bikotea;
```

Hedapena

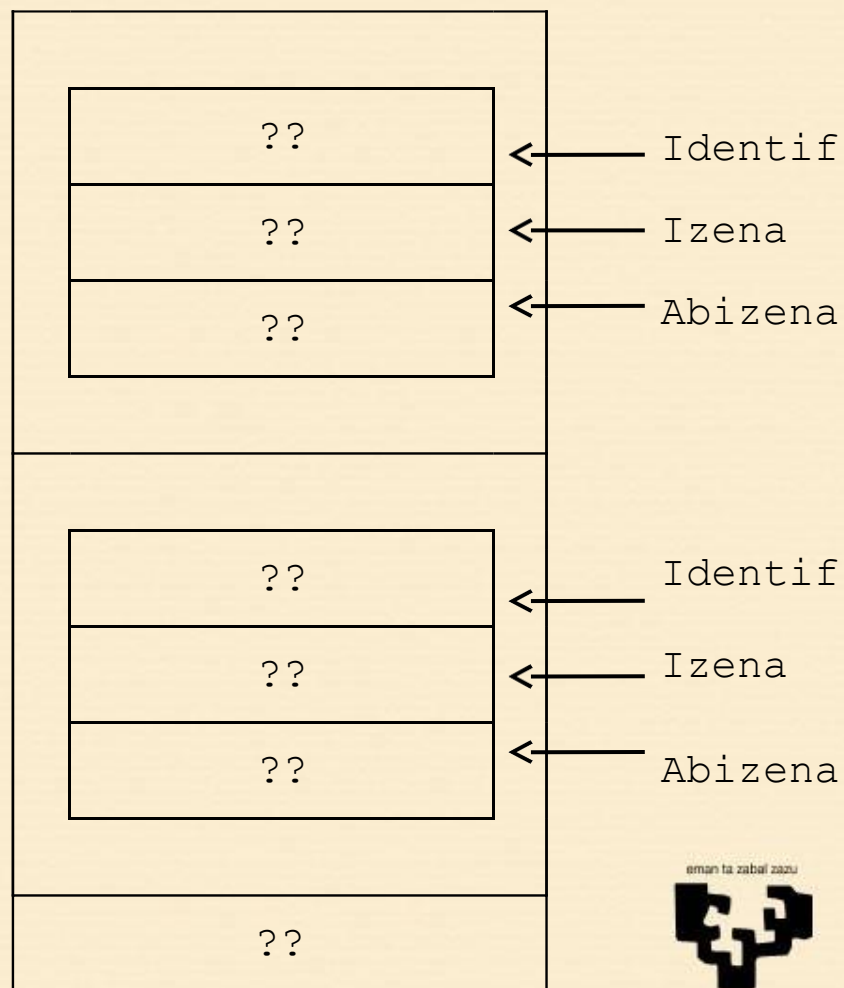
```
Bikotea.Pertsona1.Identif  
      .Izena  
      .Abizena  
      .Pertsona2.Identif  
      .Izena  
      .Abizena  
      .Helbidea
```

Pertsona1

Pertsona2

Helbidea

Bikotea



Eremuen esleipena

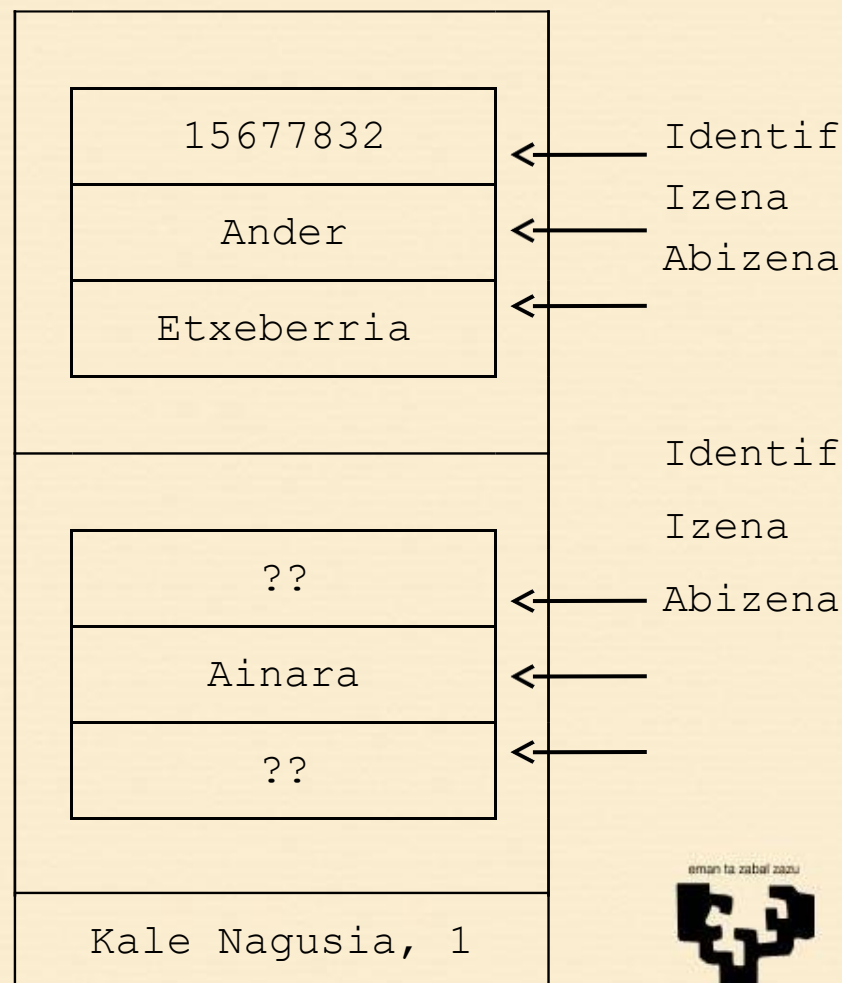
```
Bikotea.Domicilio := "Kale Nagusia, 1";  
Bikotea.Pertsonal1.Identif := 15677832;  
Bikotea.Pertsonal1.Izena := "Ander";  
Bikotea.Pertsonal1.Abizena := "Etxeberria";  
Bikotea.Pertsona2.Izena := "Ainhua";
```

Pertsonal1

Pertsona2

Helbidea

Bikotea



Erregistro motako parametroak

```
procedure irakurri (P: out T_pertsona) is  
-- Aurre: Sarrerako sekuentzian (teklatutik) zenbaki bat eta  
bi karaktere kate (20 karakteretakoak) pertsona  
identifikatzen dutenak  
-- Post: datuak P-n gorde dira  
begin  
    get(P.Identif);  
    get(P.Izena);  
    get(P.Abizena);  
end irakurri;
```

```
procedure idatzi (P: in T_pertsona) is  
-- Post: P-ren datuak irteera estandarrean idatzi dira  
begin  
    put(P.Identif);  
    put(P.Izena);  
    put(P.Abizena);  
end idatzi;
```

Erregistro motako parametroak

```
procedure bikotea_esleitu (Bikotea: in out T_bikotea; Pos: in Integer;  
                           Pertsona: in T_pertsona) is  
  -- Aurre: Pos 1 edo 2 da  
  -- Post: Pertsona1 eremuak Pertsona-ren balioa dauka Pos-en balioa 1  
  bada, edo Pertsona 2 eramuak dauka Po-en balioa 2 bada.  
  
  begin  
  
    if (Pos=1) then  
      Bikotea.Pertsona1 := Pertsona;  
    else  
      Bikotea.Pertsona2 := Pertsona;  
    end if;  
  
  end bikotea_esleitu;
```

Erregistroak eta bektoreak konbinatuz

- Erregistroak eta bektoreak konbinatu daitezke hamaika datu ezberdinetako egiturak ahalbidetuz
 - Geroz eta egitura konplexuak, orduan eta errazagoa izango da hedapenarekin lan egitea

Erregistroen arraya

```
type T_pertsona is record
  Identif: Integer;
  Izena, Abizena: String(1..20);
end record;
type T_Taula_pertsonak is array (1..5) of T_pertsona;
```

Hedapena

Pertsonak(1..5).Identif
 .Izena
 .Abizena

```
Pertsonak: T_Taula_pertsonak;
Pertsonak(3).Izena := "Jon";
```

1	2	3	4	5
??	??	??	??	??
??	??	Jon	??	??
??	??	??	??	??

eman ta zabal zazu



Erregistroa array motako eremu batekin

```
type T_Taula_km is array (1..12) of Integer;  
type T_Korrikalari record  
  Identif: Integer;  
  Izena, Abizena: String(1..20);  
  Egindako_km: T_Taula_km;  
end record;
```

```
Korrikalari: T_Korrikalari;  
Korrikalari.Izena := "Ane";  
Korrikalari.Egindako_km(5) := 580;
```

Hedapena

Korrikalari.Identif
 .Izena
 .Abizena
 .Egindako_km(1..12)

??											
Ane											
??											
??	??	??	??	580	??	??	??	??	??	??	??




Ariketa ebatzia

- Datu egitura bat definituko da klaseko afarira joango direnen izen eta abizenak gordetzeko
- Zerrendan hutsetik abiatu beharrean, zerrenda betearekin hasiko gara, klasekide guztiak dituelarik, eta afarira etorriko ez direnak zerrendatik kenduko ditugu

Datu egitura

120 matrikulatu daude OP
ikasgaien, 01 eta 31 taldeak
kontuan hartuta

```
type Ikasle is record  
  Izena: String(1..30);  
  Abizena: String(1..30);  
end record;  
type Ikasle_Bektorea is array (1 .. 120) of Ikasle;
```



Hedapena

```
Klaseko_ikasleak(1...120).Izena  
                      .Abizena
```



Soluzioaren inplementazioa

```
procedure ikasleak_kudeatu is
  Klaseko_ikasleak: Ikasle_Bektorea;
  Izena: String(1..30);
begin
  egitura_bete(Klaseko_ikasleak);
  get(Izena);
  loop exit when Izena = " ";
    ikaslea_ezabatu(Klaseko_ikasleak, Izena);
    get(Izena);
  end loop;
end ikasleak_kudeatu;
```

```
procedure egitura_bete
  (Ikasleak: out Ikasle_Bektorea) is
  Ind: Integer;
begin
    for Ind in 1..120 loop
      get(Ikasleak(Ind).Izena);
      get(Ikasleak(Ind).Abizena);
    end loop;
end egitura_bete;
```

```
procedure ikaslea_ezabatu(Eguneratzeiko_ikasleak: in out Ikasle_Bektorea;
                          Ezabatzeko_izena: in String(1..30)) is
  Ind: Integer;
begin
    Ind := posizioa(Eguneratzeiko_ikasleak, Ezabatzeko_izena);
    ezabatu(Eguneratzeiko_ikasleak, Ind);
end ikaslea_ezabatu;
```



posizioa

```
function posizioa (Ikasleak: in Ikasle_Bektorea;  
                  Ezabatzeko_izena: in String(1..30)) return Integer is  
  Indizea: Integer;  
  Aurkitua: Boolean := False;  
begin  
    Indizea:=1;  
    loop exit when Indizea > Ikasleak'last or Aurkitua;  
      if Ikasleak(Indizea).Izena = Ezabatzeko_izena then  
        Aurkitua:= True;  
      else  
        Indizea:= Indizea+1;  
      end if;  
    end loop;  
    return Indizea;  
end posizioa;
```

ezabatu

```
procedure ezabatu(Ikasleak: in out Ikasle_Bektorea;Pos: in Integer) is  
  begin
```

```
    ???????
```

```
end ezabatu;
```

**Nola ezabatuko dugu
elementua bektoretik?**

Diseinuko arazoa

- 120 ikasleren informazioa bektore batean gordetzen ari gara, eta honen tamaina beti 120koa izango da
 - Elementu bat ezabatuz gero, jada ez zaigu 120 ikasleren informazioa gordetzea interesatzen, 119rena baizik
 - Programa exekutatuz doan bitartean, geroz eta ikasle gutxiagoren informazioa gorde nahiko dugu

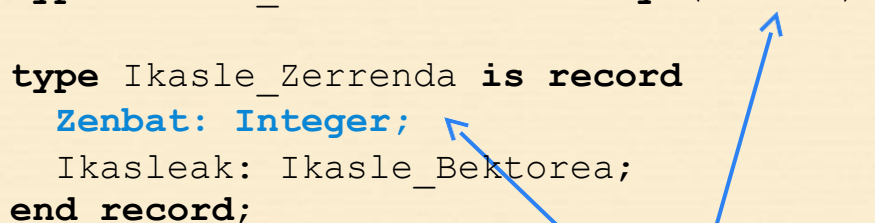
Nola konpon dezakegu
hori?

Datu egitura egokia

- Arraya erregistro baten barruan kapsulatu beharko dugu

```
type Ikasle is record
  Izena: String(1..30);
  Abizena: String(1..30);
end record;
type Ikasle_Bektorea is array (1..120) of Ikasle;

type Ikasle_Zerrenda is record
  Zenbat: Integer;
  Ikasleak: Ikasle_Bektorea;
end record;
```



Bektorean beti egongo dira 120 “ikasle”.
Baina lehenengo *Zenbat* ikasleen
informazioa bakarrik interesatzen zaigu
(gainerakoak zaborra izango dira)

Hedapena

Klasea.Zenbat
 .Ikasleak(1...120).Izena
 .Abizena

Inplementazioa

```
procedure ikasleak_kudeatu is  
    Klaseko_ikasleak: Ikasle_zerrenda;  
    Izena: String(1..30);  
begin  
    Klaseko_ikasleak.Zenbat:= 0;  
    egitura_bete(Klaseko_ikasleak);  
    get(Izena);  
    loop exit when Nombre = "          ";  
        ikaslea_ezabatu(Klaseko_ikasleak, Izena);  
        get(Izena);  
    end loop;  
end ikasleak_kudeatu;
```

Inplementazioa

```
procedure egitura_bete(IZ: out Ikasle_Zerrenda) is
  Ize, Abi: String(1..30);
begin
  get(Ize);
  get(Abi);
  loop exit when Ize = "          ";
    IZ.Zenbat := IZ.Zenbat + 1;
    IZ.Ikasleak(IZ.Zenbat).Izena:= Ize;
    IZ.Ikasleak(IZ.Zenbat).Abizena:= Abi;
  end loop;
end egitura_bete;
```

```
procedure ikaslea_ezabatu(Eguneratzeko_ikasleak: in out Ikasle_Zerrenda;
                          Ezabatzeko_Izena: in String(1..30)) is
  Ind: Integer;
begin
  Ind := posizioa(Eguneratzeko_ikasleak, Ezabatzeko_izena);
  ezabatu(Eguneratzeko_ikasleak, Ind);
end ikaslea_ezabatu;
```

Inplementazioa

```
function posicion (IZ: in Ikasle_Zerrenda;  
                  Izena: in String(1..30)) return Integer is  
  Indizea: Integer;  
  Aurkitua: Boolean := False;  
begin  
  Indizea:=1;  
  loop exit when Indizea > IZ.Zenbat or Aurkitua;  
    if IZ.Ikasleak(Indizea).Izena = Izena then  
      Aurkitua:= True;  
    else  
      Indizea:= Indizea+1;  
    end if;  
  end loop;  
  return Indizea;  
end posizioa;
```

Inplementazioa

```
procedure ezabatu(IZ: in out Ikasle_zerrenda; Pos: in Integer) is  
  begin  
    ezkerrea_mugitu(IZ, Pos);  
    IZ.Zenbat := IZ.Zenbat - 1;  
  end eliminar;
```

```
procedure ezkerrera_mugitu(IZ: in out Ikasle_Zerrenda; Ind: in Integer) is  
  Indizea: Integer := Ind;  
  begin  
    loop exit when Indizea = IZ.Zenbat;  
    IZ.Ikasleak(Indizea) := IZ.Ikasleak(Indizea+1);  
    Indizea := Indizea + 1;  
  end loop;  
end ezkerrera_mugitu;
```


Proposatutako ariketa 1

- Sarrera estandarrean 10 saltzaileren datuak daude. Saltzaile bakoitzeko bere identifikatzailea, izena, abizeta eta 5 zenbaki, azken 5 hilabetetan ibilitako kilometroak adierazten dituztenak

```
123 Jorge Pastor 0 48 100 500230
600 Iñigo Balda 800 1000 0 900
2500
...
```

Proposatutako ariketa 1

- Eskatzen dena:
 - Datu horiek gordetzeko datu egiturak erazagutu (motak eta aldagaiak)
 - Funtzio bat idatzi, zeinak sarrera parametro gisa saltzaileen zerrenda jasotzen duen eta ibilitako kilometroetan altuena eta saltzailearen izena itzuliko duen
 - Zein motatakoa izango da funtzioak itzuliko duen balioa?

Proposatutako ariketa 2

- Ikasleen informazioa (izena eta notak) gordeko duen datu egitura definitu
 - Gehienez 100 ikasle egongo dira, eta ikasleko 15 nota, baina baliteke horrenbeste ikasleren informazioa ez izatea, edota horrenbeste nota ikasle bakoitzarentzat
 - Hau da, zabor elementuak egon daitezke

Proposatutako ariketa 2

- Eskatzen dena:
 - Azpiprograma bat idatzi zeinak egituraren amaieran ikasle bat txertatuko duen
 - Azpiprograma bat idatzi zeinak sarrera-parametro gisa egitura hori jasoko duen eta batazbesteko nota orokorra itzuliko duen

Teoriako ariketa

```
package datuak is
  type Ikasle is record
    Izena: String(1..10);
    Abizena: String(1..10);
  end record;
  type Ikasle_Bektorea is array (1..120) of Ikasle;
  type Ikasle_Zerrenda is record
    Zenbat: Integer;
    Ikasleak: Ikasle_Bektorea;
  end record;

end datuak;
```

Teoriako ariketa

```
with datuak, ada.text_io;  
use datuak, ada.text_io;  
with egitura_bete, ikaslea_ezabatu;
```

```
procedure ikasleak_kudeatu is
```

```
    klaseko_ikasleak: Ikasle_zerrenda;  
    Izena: String(1..10);
```

```
begin
```

```
    klaseko_ikasleak.Zenbat:= 0;  
    egitura_bete(klaseko_ikasleak);  
    put_line("ezabatzeko ikaslearen izena:(" return amaitzeko)");  
    get(Izena);
```

```
    loop exit when Izena = "          ";  
        ikaslea_ezabatu(klaseko_ikasleak, Izena);  
        put_line("ezabatzeko ikaslearen izena:(" return amaitzeko)");  
        get(Izena);
```

```
    end loop;
```

```
end ikasleak_kudeatu;
```

Teoriako ariketa

```
with datuak, ada.text_io;
use datuak, ada.text_io;
procedure egitura_bete(IZ: out Ikasle_Zerrenda) is
    Ize, Abi: String(1..10);
begin
    put_line("sartu izena/return amaitzeko");
    get(Ize);
    put_line("sartu abizena");
    get(Abi);
    loop exit when Ize = "          " ; - - 10 hutsune
        IZ.Zenbat := IZ.Zenbat + 1;
        IZ.Ikasleak(IZ.Zenbat).Izena:= Ize;
        IZ.Ikasleak(IZ.Zenbat).Abizena:= Abi;
        put_line("sartu izena/return amaitzeko");
        get(Ize);
        put_line("sartu abizena");
        get(Abi);
    end loop;
    put_line("egitura beteta");
end egitura_bete;
```

Teoriako ariketa

```
with datuak;
```

```
use datuak;
```

```
with posizioa, ezabatu;
```

```
procedure ikaslea_ezabatu(Eguneratzeko_ikasleak: in out Ikasle_Zerrenda;  
Ezabatzeko_Izena: in String) is
```

```
    Ind: Integer;
```

```
begin
```

```
    Ind := posizioa(Eguneratzeko_ikasleak, Ezabatzeko_izena);
```

```
    ezabatu(Eguneratzeko_ikasleak, Ind);
```

```
end ikaslea_ezabatu;
```


Teoriako ariketa

```
with datuak;
```

```
use datuak;
```

```
function posizioa (IZ: in Ikasle_Zerrenda; Izena: in String) return Integer is
```

```
    Indizea: Integer;
```

```
    Aurkitua: Boolean := False;
```

```
begin
```

```
    Indizea:=1;
```

```
    loop exit when Indizea > IZ.Zenbat or Aurkitua;
```

```
        if IZ.Ikasleak(Indizea).Izena = Izena then
```

```
            Aurkitua:= True;
```

```
        else
```

```
            Indizea:= Indizea+1;
```

```
        end if;
```

```
    end loop;
```

```
    return Indizea;
```

```
end posizioa;
```

Teoriako ariketa

```
with datuak;
```

```
use datuak;
```

```
with ezkerrera_mugitu;
```

```
procedure ezabatu(IZ: in out Ikasle_zerrenda; Pos: in Integer) is
```

```
begin
```

```
    ezkerrera_mugitu(IZ, Pos);
```

```
    IZ.Zenbat:= IZ.Zenbat - 1;
```

```
end ezabatu;
```

Teoriako ariketa

with datuak;

use datuak;

procedure ezkerrera_mugitu(IZ: **in out** Ikasle_Zerrenda; Ind: **in** Integer) **is**

Indizea: Integer:= Ind;

begin

loop exit when Indizea = IZ.Zenbat;

 IZ.Ikasleak(Indizea):= IZ.Ikasleak(Indizea+1);

 Indizea:= Indizea + 1;

end loop;

end ezkerrera_mugitu;

Proposatutako ariketa 1

```
package datuak is
  type km_bektorea is array(1..5) of integer;

  type saltzaile is record
    identifikazioa: integer;
    izena: string(1..6);
    abizena:string(1..6);
    kmetroak: km_bektorea;
  end record;

  type saltzaileen_bektorea is array(1..10) of saltzaile;

  type T_emaizta is record --emaitza itzultzeko
    izena: String(1..6);
    km_max: integer;
  end record;
end datuak;
```

Proposatutako ariketa 1

```
with ada.text_io, ada.integer_text_io, datuak;  
use ada.text_io, ada.integer_text_io, datuak;
```

```
procedure lista_bete (l: out saltzaileen_bektorea) is
```

```
    id: integer;  
    izena, abizena: string(1..6);  
    kilom: integer;
```

```
begin
```

```
    put("lista betetzera goaz:");
```

```
    new_line;
```

```
    for i in 1..l'last loop
```

```
        put ("Sartu identifikadorea saltzaile("&integer'image(i)&"): ");
```

```
        new_line;
```

```
        get(id);
```

```
        put ("Sartu izena saltzaile("&integer'image(i)&"): (6 KARAKTERE SARTU BEHAR DITUGU)");
```

```
        new_line;
```

```
        get(izena);
```

```
        put ("introduzca el abizena del saltzaile("&integer'image(i)&"): (6 KARAKTERE SARTU BEHAR DITUGU)");
```

```
        new_line;
```

```
        get(abizena);
```

```
        l(i).identifikazioa:=id;
```

```
        l(i).izena:=izena;
```

```
        l(i).abizena:=abizena;
```

```
        for j in 1..l(i).kmetroak'last loop
```

```
            put("sartu km( "&integer'image(j)&"):");
```

```
            new_line;
```

```
            get(kilom);
```

```
            l(i).kmetroak(j):=kilom;
```

```
        end loop;
```

```
    end loop;
```

```
    new_line;
```

```
end lista_bete;
```

Proposatutako ariketa 1

with datuak;

use datuak;

function **km_handi_eta_izena** (l: in saltzaileen_bektorea) return T_emaiza is

 ema:T_emaiza;

begin

 ema.km_max:=0;

 for i in 1..l'last loop

 for j in 1..l(i).kmetroak'last loop

 if (ema.km_max<l(i).kmetroak(j)) then

 ema.km_max:=l(i).kmetroak(j);

 ema.izena:=l(i).izena;

 end if;

 end loop;

 end loop;

 return ema;

end **km_handi_eta_izena**;

Proposatutako ariketa 1

```
with datuak, ada.text_io; use ada.text_io, lista_bete, km_handi_eta_izena;
```

```
use datuak, ada.text_io; use ada.text_io;
```

procedure nagusia is

```
    lista: saltzaileen_bektorea;
```

```
-- lista: datuak.saltzaileen_bektorea; -- "USE DATUAK" IPINI EZ DUGUNEZ, HELBIDEA IPINI BEHAR DUGU
```

```
    ema: T_emaiza;
```

begin

```
--lista_bete(lista);
```

```
--DATUAK MANUALKI EDO AUTOMATIKOKI SARTU NAHI DUZUEN AUKERA DEZAKEZUE.
```

```
    lista(1).identifikazioa:=1; lista(2).identifikazioa:=2; lista(3).identifikazioa:=3; lista(4).identifikazioa:=4; lista(5).identifikazioa:=5;
```

```
    lista(6).identifikazioa:=6; lista(7).identifikazioa:=7; lista(8).identifikazioa:=8; lista(9).identifikazioa:=9; lista(10).identifikazioa:=10;
```

```
    lista(1).izena:="pepe1 "; lista(2).izena:="pepe2 "; lista(3).izena:="pepe3 "; lista(4).izena:="pepe4 "; lista(5).izena:="pepe5 ";
```

```
    lista(6).izena:="pepe6 "; lista(7).izena:="pepe7 "; lista(8).izena:="pepe8 "; lista(9).izena:="pepe9 "; lista(10).izena:="pepe10";
```

```
    lista(1).abizena:="garcia"; lista(2).abizena:="garcia"; lista(3).abizena:="garcia"; lista(4).abizena:="garcia"; lista(5).abizena:="garcia";
```

```
    lista(6).abizena:="garcia"; lista(7).abizena:="garcia"; lista(8).abizena:="garcia"; lista(9).abizena:="garcia"; lista(10).abizena:="garcia";
```

```
    lista(1).kmetroak:= (1111,2,3,4,5);    lista(2).kmetroak:= (11,12,13,14,15);    lista(3).kmetroak:= (21,22,23,24,25);    lista(4).kmetroak:= (31,32,33,34,35);
```

```
    lista(5).kmetroak:= (41,42,143,44,45);
```

```
    lista(6).kmetroak:= (51,52,53,54,55);    lista(7).kmetroak:= (61,62,63,64,65);    lista(8).kmetroak:= (71,72,73,74,75);    lista(9).kmetroak:=  
(81,82,83,84,85);
```

```
    lista(10).kmetroak:= (91,92,93,94,95);
```

```
    ema:=km_handi_eta_izena(lista);
```

```
    put("1) Saltzailearen izena "& ema.izena &" da eta "&integer'image(ema.km_max)&" kilometro ditu");
```

end nagusia;