Oinarrizko Programazioa

5. Programazio-lengoaien erabilera: ADA lengoaia

Oinarrizko Programazioa

Gai zerrenda:

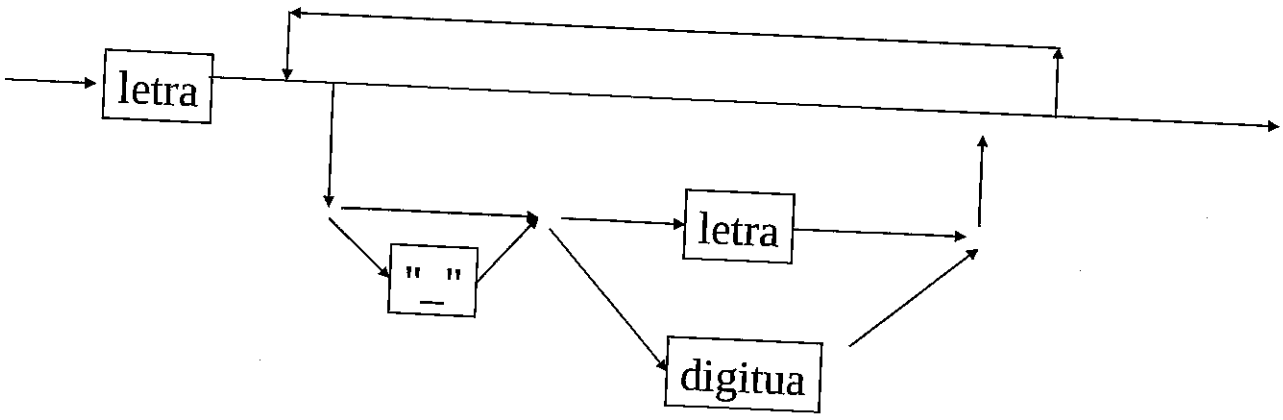
1. Sarrera.
2. Programazioko oinarrizko kontzeptuak.
3. Programen beheranzko diseinua. Azpiprogramak: funtzioak eta prozedurak.
4. Oinarrizko datu-egiturak.
5. *Programazio-lengoaien erabilera.*
6. Aplikazio-adibideak.

Sintaxi-diagramak

- Sintaxi-diagramen bidez lengoaien idazkera definitzen da.
- Sintaxi-diagrama bakoitzak eraikuntza bakun baten sintaxia definitzen du.
- Diagramaren sarreratik bere irteerarainoko edozein bidek, gezien norantza jarraituz, markatzen du eraikuntza sintaktiko egokia.
- Ada lengoaia osorako sintaxi-diagramak daude.

Identifikadorea: sintaxi-diagrama

Identifikadorea



Literalak

Balio konstanteak adierazteko erabiltzen dira (mota desberdinetakoak), adibidez:

- Osoko literalak: `0  1  60  1_000`
- Literal errealak: `0.0  3.14158`
- Karaktereak: `'H'  ':'  ' '`
- Kateak: `"Ordua:"  "???"`

Identifikadoreak

- Konstante, aldagai, mota, azpiprograma, pakete eta Ada programetako beste entitate batzuei ematen zaizkien izenak dira
- Letra beraren maiuskulak eta minuskulak baliokideak dira
  - Adibidez:

```
seg_minutuko = Seg_Minutuko = SEG_MINUTUKO
seg_minutuko /=  SegMinutuko
```

Banatzaileak

- Ondoz ondoko zenbaki-literal, identifikadore edo hitz erreserbatuen artean, gutxienez banatzaile bat jarri behar da:

```
,  ;  :  .  '
(  )
** *  /  +  -  &
=     /= <  <= >= >
:=    ..  |  => <>
```

baita zuriunea ere

Programazio-estiloa

- Komeni da iruzkinak erabiltzea programaren funtzioa adierazi eta zati bakoitzaren funtzionamendua esplikatzeko
  - Iruzkinak Adan: "--" ondoren lerro bukaeraraino
- Identifikadore ahalik eta deskriptiboenak aukeratu behar dira
  - Erabili maiuskulak edo minuskulak (baina beti era berean!).

Oinarrizko ekintzak Adan
Datu-irakurketa (teklatutik)

- **Irakurri_Osoa (*ald1*) ;**
  - Ez da aipatzen fitxategiaren izena, teklatutik irakurtzen denez, sarrera estandarra delako
- Baina programa duen fitxategi-hasieran hau jarri behar da:

  with Irakurri _Osoa ;
- Antzekoak

  Irakurri_Erreala (ald2) ;

  Irakurri_Karakterea (ald3) ;

Oharra:
Ekintza hauek ez dira Ada estandarrekoak, ikastaro honetarako asmatuak baizik

Oinarrizko ekintzak Adan
Asignazioa

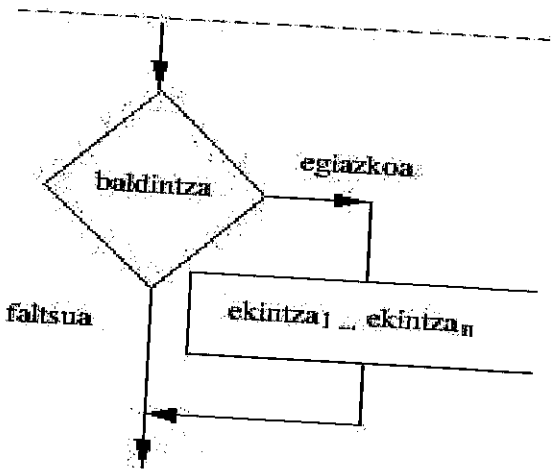
- *Algoritmoetan bezalaxe*

  aldagaia **:=** *adierazpena ;*
- *adierazpena* ebaluatuz lortzen den balioa aldagaiaren balio berri bezala ezartzen du.
- ezkerreko aldagaiak galtzen du lehengo balioa.
- adierazpenean aldagairik azaltzen bada, berau ebaluatzen denean daukan balioa lortzen da, baina balio hori ez da aldatzen.

Kontrol-egiturak Adan Iterazioa

bitartean baldintza **egin**
    ekintza$_1$ ... ekintza$_n$
ambitartean

while baldintza **loop**
    ekintza$_1$ ... ekintza$_n$
end loop ;

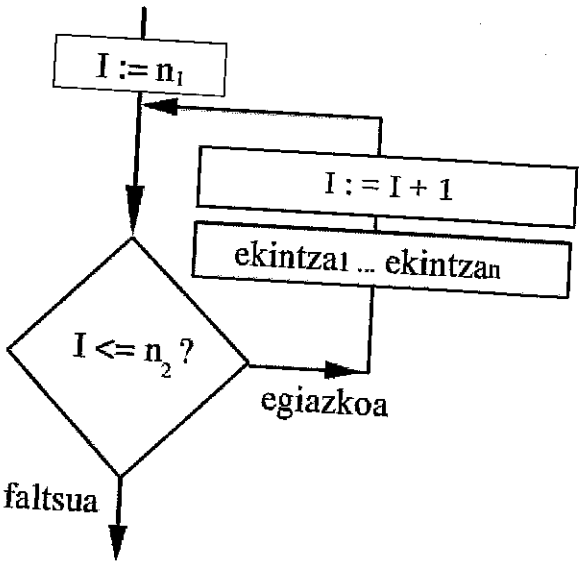


Kontrol-egiturak Adan Aldi kopuru jakineko iterazioa

egin I **guztietarako** n_1**tik** n_2 **raino**
    ekintza$_1$ ... ekintza$_n$
amguztietarako

for I **in** n_1..n_2 **loop**
    ekintza$_1$ ... ekintza$_n$
end loop ;

Beste aukerak:
for I **in reverse** 1..14 **loop**
for I **in** 'a'..'z' **loop**



Azpiprograma adibidea (I)

Ordua_Erakutsi programak, gauerdiaz gero pasatu diren segundoak irakurtzen ditu eta 24 orduko adierazpidea erabiliz idazten du O:M:S formatuan.
Adibidez: 54450 sarrera-datuarekin programaren irteera hau da:
< Ordua: 15: 7: 30 >

```
with Irakurri_Osokoa, Idatzi_Osokoa, Idatzi_Katea ;
procedure Ordua_Erakutsi is
   Seg_Minutuko: constant Integer := 60;
   Min_Ordubeteko: constant Integer := 60;
   Seg_Ordubeteko: constant Integer := Seg_Minutuko * Min_Ordubeteko;
   Ordua  : Integer;
   O, M, S : Integer;
```


Azpiprograma adibidea (II)

```
begin
    Irakurri_Osoa (Ordua);-- gauerdiaz geroko segundoak
    H := Ordua / Seg_Ordubeteko;      -- gauerdiaz geroko orduak
    Ordua := Ordua  rem  Seg_Ordubeteko; -- azken orduaz geroko segundoak
    M := Ordua / Seg_Minutuko;    -- azken orduaz geroko minutuak
    S := Ordua  rem  Seg_Minutuko;-- azken minutuaz geroko segundoak
    Idatzi_Katea ("Ordua:   ");
    Idatzi_Osoa (O); Idatzi_Katea (":");
    Idatzi_Osoa (M); Idatzi_Katea (":");
    Idatzi_Osoa (S);
end Ordua_Erakutsi;
```

Azpiprograma bat erabili beste azpiprograma batetik (II)

- Aukera2:
  - Azpiprograma lagungarria azpiprograma nagusiaren **barruan** definitzen da.
    - Kasu honetan azpiprograma lagungarri hori ezin da erabili azpiprograma nagusi horretatik kanpo, azpiprograma **lokala** izango baita.

a.adb fitxategia:

```
procedure  A … is        …
   procedure  B … is
   begin
      …
   end B;
begin
      …
      B(…)
      …
end A;
```


Kanpo-memoriako datu-egiturak

- Orain arte, datuen irakurketa eta idazketa sarrera eta irteera estandarretik (teklatutik eta pantailan) egin ditugu. Beti testu gisa.

Atal honetako kontzeptu berriak:

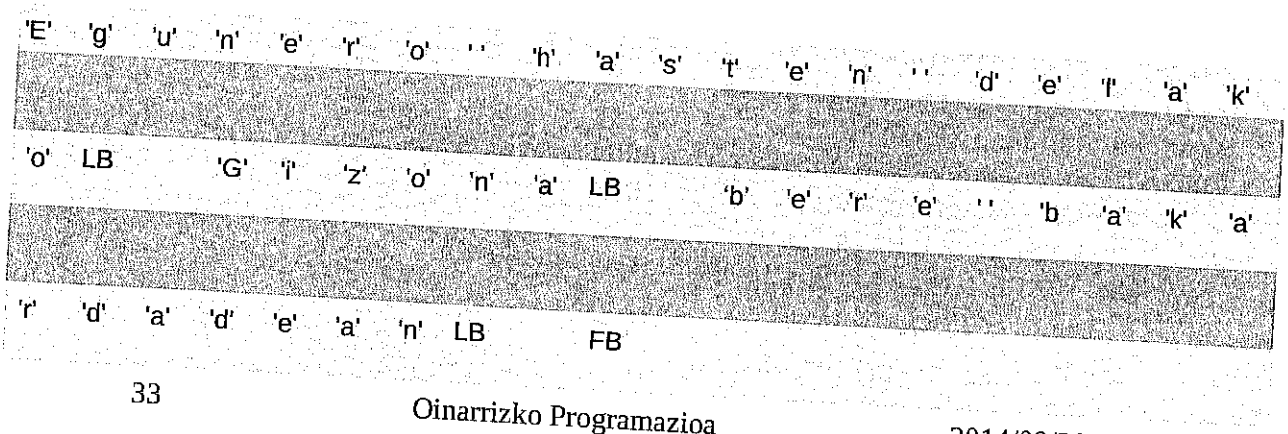
- Sarrera eta irteera ez-estandarra: fitxategiak
- Testu-fitxategiak eta fitxategi bitarrak
- Fitxategi sekuentzialak eta atzipen zuzenekoak
- Fitxategiak erabiltzeko azpiprograma-paketeak

Testu-fitxategiak

Testu-fitxategiko elementuak, lerroak eta karaktereak dira, eta hauek bata bestearen atzetik irakurri eta idatzi behar dira.

Adibidez:

```
< Egunero hasten delako
Gizona
bere bakardadean >
```



Testu-fitxategiak (*Ada.Text_IO* paketea)

- Sarrera-irteerarako eragiketa guztiak paketetan daude.
  - Paketeak erabiltzeko, "*with*" klausulan izendatzen dira.
- Sarrera-irteerako eragiketak ohiko azpiprograma-deien bidez exekutatzen dira.
- Hauek dira gehien erabiltzen diren paketeak:

```
Ada.Text_IO;
Ada.Integer_Text_IO;
Ada.Float_Text_IO;
```

Testu-fitxategiak (*Ada.Text_IO* paketea)
Ireki eta itxi fitxategiak (III)

Fitxategi bat ireki

```
procedure Open  (File : in out File_Type;
        Mode : in File_Mode;
        Name : in String);
type File_Mode is (In_File, Out_File);
```

Erabilera

```
Ada.Text_IO.Open (File => Barne-izena,
                  Mode => Ada.Text_IO.In_File,
                  Name => Kanpo-izena)
```

Barne-izena ***Ada.Text_IO.File_Type*** motako aldagai gisa erazagutu behar da

Adibidea

```
Ada.Text_IO.Open(F1,Ada.Text_IO.In_File,
                 "/home/jiplanaz/mahaigaina/lab03/datuak.txt")
```


Testu-fitxategiak (*Ada.Text_IO* paketea)
Bukaeraren eta lerroaren kontrola

Sarrera-fitxategi bateko bukaeran edo lerro bateko bukaeran ote gauden jakiteko balio duten funtzio boolearrak

```
function End_of_Line (File: in File_Type) return Boolean;
```

Erabilera

```
Ada.Text_IO.End_of_line(File => Barne-izena);
```

```
function End_of_File (File: in File_Type) return Boolean;
```

Erabilera

```
Ada.Text_IO.End_of_File(File => Barne-izena);
```

Testu-fitxategiak (*Ada.Text_IO* paketea)
Sarrera

Sarrerako fitxategi batetik irakurri

- **Ada.Text_IO** -- Testu-fitxategiak tratatzeko azpiprogramak dituen paketea
  `procedure Get (File : in File_Type; Item : out Character);`
- **Ada.Integer_Text_IO** -- Testu-fitxategietan osoak tratatzeko azpiprogramak
  `procedure Get (File : in File_Type; Item : out Integer);`
- **Ada.Float_Text_IO**-- Testu-fitxategietan errealak tratatzeko azpiprogramak
  `procedure Get (File : in File_Type; Item : out Float);`

Erabilera

```
Ada.Text_IO.Get (File => Barne-izena, Item => Character motako aldagaia)
Ada.Integer_Text_IO.Get (File => Barne-izena, Item => Osoko motako aldagaia)
Ada.Float_Text_IO.Get (File => Barne-izena, Item => Float motako aldagaia)
```


Testu-fitxategiak (*Ada.Text_IO* paketea)
Sarrera (II)

Sarrerako fitxategi batetik lerro bat irakurri eta hurrengo lerrora pasa

Ada.Text_IO -- Testu-fitxategiak tratatzeko azpiprogramak dituen paketea
`procedure Get_Line`
`(File : in File_Type; Item : out String; Last out Integer);`

Erabilera

```
Ada.Text_IO.Get_Line (File => Barne-izena,
     Item => Character motako aldagaia,
     Last => Integer)
-- azkeneko argumentuan zenbat karaktere irakurri den gordetzen da
```

Testu-fitxategiak (*Ada.Text_IO* paketea)
Irteera (III)

- Sarrerako fitxategia teklatua baldin bada ***Standard_Input*** da fitxategiaren izena
- Irteerako fitxategia pantaila baldin bada, ***Standard_Output*** da fitxategiaren izena
  - Kasu horietan, fitxategia sortu eta irekitzea ez da beharrezkoa.
  - Gainera, kasu horietan ere, fitxategiaren parametroa jartzea ez da beharrezkoa
- Ikusi ditugun azpiprogramen deiak horrela gelditzen zaizkigu:

```
Ada.Text_IO.Put(Item => Karaktere motako aldagaia);
Ada.Integer_Text_IO.Put(Item => Oso motako ald.);
Ada.Float_Text_IO.Put(Item => Erreal motako ald);
Ada.Text_IO.New_line;
Ada.Text_IO.Skip_line;
Ada.Text_IO.Get(Item => Karaktere motako aldagaia)
Ada.Integer_Text_IO.Get(Item =>Oso motako ald);
Ada.Float_Text_IO.Get(Item => Erreal motako ald)
```


Lerroak_Kopiatu azpiprograma

```
with Ada.Text_IO;
procedure Lerroak_Kopiatu  (Iturburu_Fitxategia,
               Helburuko_Fitxategia : in String) is
     F1, F2: Ada.Text_IO.File_Type;
     Lerro_Luzera_Max    : constant Natural := 120;
     Testu_Lerroa: String (1 .. Lerro_Luzera_Max);
     Lerro_Luzera: Natural range 0 .. Lerro_Luzera_Max;
begin
     Ada.Text_IO.Open  (F1, Ada.Text_IO.In_File, Iturburu_Fitxategia);
     Ada.Text_IO.Create(F2, Ada.Text_IO.Out_File, Helburuko_Fitxategia);
     while not Ada.Text_IO.End_of_File (F1) loop
           Ada.Text_IO.Get_Line (F1 Testu_Lerroa, Lerro_Luzera);
           Ada.Text_IO.Put_Line (F2, Testu_Lerroa (1 .. Lerro_Luzera));
     end loop;
     Ada.Text_IO.Close (F1);
     Ada.Text_IO.Close (F2);
end Lerroak_Kopiatu;
```

Idatzi_Osoa azpiprograma

idatzi_osoa.adb

```
with Ada.Integer_Text_IO;
procedure Idatzi_Osoa (I : in Integer) is
begin
   Ada.Integer_Text_IO.Put (I);
end Idatzi_Osoa;
```


Testuzkoak ez diren fitxategiak

- Elementu-sekuentzia bat dira
- Elementu guztiak mota berekoak dira
- Adierazpide bitarrean gordeta daude
- Erabilera eraginkorragoa dute testu-fitxategiak baino (memoria eta espazioan)
- Begi bistan ulergaitzak dira (ezin dira testu gisa editatu)
- Bi eratakoak:
  - atzipen sekuentzialekoak
  - atzipen zuzenekoak

Testuzkoak ez diren fitxategiak (IV)
Sarrera-irteera eragiketak

```
procedure Read (File    : in File_Type;
                Element : out ELEM);
procedure Write  (File    : in File_Type;
                Element : in ELEM);
function End_of_File (File : in File_Type)
                return Boolean;
```


Testuzkoak ez diren fitxategiak (V)

Erabilera (ikusitako prozedura/funtzio batzuk) (1)

```
package motak is
type Elementu is record
      Nor: Datuak_Nor;
      Telefonoa: Telefono_Zenbaki;
   end record;
end motak;
package Elementu_SI is new Sequential_IO (Motak.Elementu);
procedure Elementu_SI.Read
   (File: in Elementu_SI.File_Type; E: out Motak.Elementu);
procedure Elementu_SI.Write
   (File: in Elementu_SI.File_Type; E: out Motak.Elementu);
function  Elementu_SI.End_of_File  (File:  in  Elementu_SI.File_Type)  return
   Boolean;
```