Oinarrizko Programazioa

Klaseetako materiala

Programazio lengoaiaren erabilera: Ada lengoaia

(5. gaia)

Arantza Díaz de Ilarraza Sánchez
Kepa Sarasola Gabiola

Lengoaiak eta Sistema Informatikoak Saila
Euskal Herriko Unibertsitatea

Oinarrizko Programazioa

Gai zerrenda:

- **Sarrera.**
- **Programazioko oinarrizko kontzeptuak.**
- **Programen beheranzko diseinua. Azpiprogramak: funtzioak eta prozedurak.**
- **Oinarrizko datu-egiturak.**
- ***Programazio-lengoaien erabilera.***
- **Aplikazio-adibideak.**

5. Programazio-lengoaien erabilera

5.1 Programazio-lengoaien ezaugarriak:

- Sintaxia eta semantika

5.2 ADA programazio lengoaia

5.3 Kanpo-memoriako datu-egiturak:

- Testu-fitxategiak.
- Testuzkoak ez diren fitxategiak. (eduki osagarriak)

Sintaxia eta Semantika

- SINTAXIA:
  sententziak idazteko arau-multzoa
  - sintaxi-diagramen bidez
  - EBNF erregelen bidez

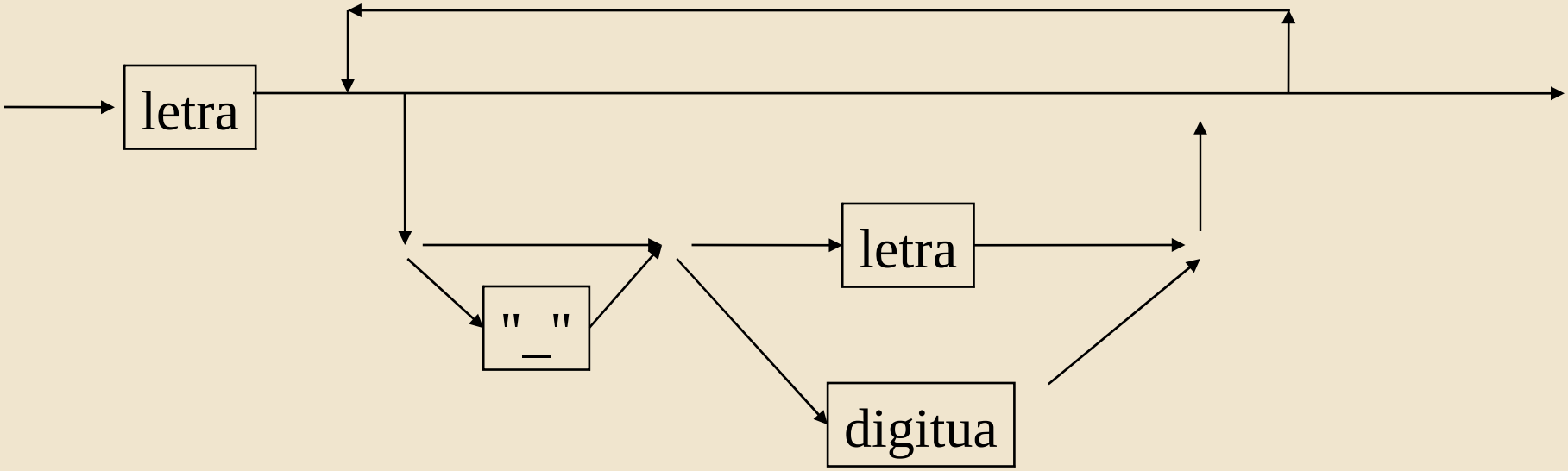
- SEMANTIKA:
  sententzi horien esangura

Sintaxi-diagramak

- Sintaxi-diagrama bakoitzak eraikuntza bakun baten sintaxia definitzen du.
- Aukera ezazu diagramaren sarreratik bere irteeraraino edozein bide gezien norantza jarraituz eta bidean topatzen duzun guztia kontutan har ezazu
- Ada lengoaia osorako sintaxi-diagramak daude.

Identifikadorea: sintaxi-diagrama

Identifikadorea



Identifikadorea: EBNF definizioa

identifikadorea ::= letra { ["_"] (letra | digitua) }

letra ::= **"A" | "B" | "C" | ... | "Z"**
| "a" | "b" | "c" | ... | "z"

digitua::= **"0" | "1" | "2" | "3" | "4" | "5"**
| "6" | "7" | "8" | "9"

Ikus Ada lengoaiaren sintaxiaren definizio osoa :

- http://cui.unige.ch/db-research/Enseignement/analyseinfo/Ada95/BNFindex.html
- Adagiden: Help/languageRM Erreferentzia eskuliburua

Adaren sintaxia

- Ada-programa batean idatzi daitezkeen ikurrak (~hitzak) honela sailka daitezke:
  - literalak
  - identifikadoreak
  - hitz erreserbatuak
  - banatzaileak.

Literalak

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

- 0 1 60 1_000 (osoko literalak)
- 0.0 3.14158 (literal errealak)
- 'H' ':' ' ' (karaktereak)
- "Ordua: " "???" (kateak)

Identifikadoreak

- Konstante, aldagai, mota, azpiprograma, pakete eta Ada programetako beste entitate batzuei ematen zaizkien izenak dira.
- Letra beraren maiuskulak eta minuskulak baliokideak dira
- seg_minutuko = Seg_Minutuko = SEG_MINUTUKO

  seg_minutuko /= SegMinutuko

Hitz erreserbatuak

- Ada lengoaian xede jakinetarako erabiltzen diren ingelesezko hitzak dira.
- Ez dira aukeratu behar identifikadore bezala
- Horregatixe esaten zaie ‘hitz erreserbatuak’.
- Esate baterako :

  if then else while loop

  procedure begin end constant

Hitz erreserbatuak

abort	**declare**	**generic**	**of**	**select**
abs	**delay**	**goto**	**or**	**separate**
accept	**delta**		**others**	**subtype**
access	**digits**	**if**	**out**	
all	**do**	**in**		**task**
and		**is**	**package**	**terminate**
array			**pragma**	**then**
at	**else**		**private**	**type**
	elsif	**limited**	**procedure**	
	end	**loop**		
begin	**entry**		**raise**	**use**
body	**exception**		**range**	
	exit	**mod**	**record**	**when**
			rem	**while**
		new	**renames**	**with**
case	**for**	**not**	**return**	
constant	**function**	**null**	**reverse**	**xor**

Banatzaileak

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

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

  baita zuriunea ere

Programazio-estiloa

- Iruzkinak lasai erabil itzazu programaren funtzioa adierazi eta zati bakoitzaren funtzionamendua esplikatzeko.
  - Iruzkinak adan: "--" ondoren lerro bukaeraraino
- Identifikadore ahalik eta deskriptiboenak aukera itzazu.
  - Erabil maiuskulak edo minuskulak
  - (baina beti era berean!).

Datuen adierazpidea Adan

Oinarrizko DATU-MOTAK

- Balio bakar bat dute:

Datu-mota	Balioak	ADAn
Osokoa	osoko zenbakiak	Integer
Erreala	zenbaki errealak	Float
Karakterea	karaktereak	Character
Boolearra	true eta false	Boolean
Katea	karaktere-kateak	String

Datuen adierazpidea Adan
Konstanteak eta aldagaiak

- Erazagutu behar dira prozedura hasieran
- Konstanteei balioa ezartzen zaie:

  Pi : **constant** float := 3.14159 ;

  Zuriunea : **constant** character := ´ ´ ;

- Aldagaiei hasierako balioa ezar dakieke:

  Zabalera : float ;

  Kontagailua : integer := 0 ;

Oinarrizko ekintzak Adan

Datu-irakurketa (teklatutik)

- **Irakurri _Osokoa (** *ald1***) ;**
- Baina programa duen fitxategi-hasieran hau jarri behar da:

  with Irakurri _Osokoa **;**
- Antzekoak

  Irakurri_Erreala (ald2**) ;**

  Irakurri _Karakterea (ald3**) ;**

Oharra:

Hau ez da Ada estandarrekoa, ikastaro honetarako asmatua baizik

Oinarrizko ekintzak Adan

Datu- idazketa (pantailan)

- **Idatzi _Osokoa (***adierazpen***) ;**
- Baina programa duen fitxategi-hasieran hau jarri behar da:

  with Idatzi _Osokoa **;**
- Antzekoak

  Idatzi_Erreala (ald2**) ;**

  Idatzi _Karakterea (ald3**) ;**

Oharra:

Hau ez da Ada estandarrekoa, ikastaro honetarako asmatua baizik

Oinarrizko ekintzak Adan

Asignazioa

- *Algoritmoetan bezalaxe*

  aldagaia := *adierazpena ;*

- *adierazpena* ebaluatuz lortzen den balioa aldagaiaren balio berri bezala ezartzen du.
- 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

Baldintzazko egitura

baldin baldintza **orduan** ekintza$_1$... ekintza$_n$ **ambaldin**	**if** baldintza **then** ekintza$_1$... ekintza$_n$ **end if ;**

baldin baldintza **orduan**
ekintza$_1$... ekintza$_n$
bestela
ekintza$_{21}$... ekintza$_{2n}$
ambaldin
	if baldintza **then**
	ekintza$_1$... ekintza$_n$
	else
	ekintza$_{21}$... ekintza$_{2n}$
	end if ;

Kontrol egiturak Adan

Baldintzazko egitura

baldin b1 **orduan**
ekintza$_{11}$... ekintza$_{1n}$
bestela baldin b2 **orduan**
ekintza$_{21}$... ekintza$_{2n}$
bestela baldin b3 **orduan**
ekintza$_{31}$... ekintza$_{3n}$
bestela
ekintza$_{41}$... ekintza$_{4n}$
ambaldin
ambaldin

ambaldin

if b1 **then**
ekintza$_{11}$... ekintza$_{1n}$
elsif b2 **then**
ekintza$_{21}$... ekintza$_{2n}$
elsif b3 **then**
ekintza$_{31}$... ekintza$_{3n}$
else ekintza$_{41}$... ekintza$_{4n}$
end if ;

Kontrol egiturak Adan Iterazioa

bitartean baldintza **egin**

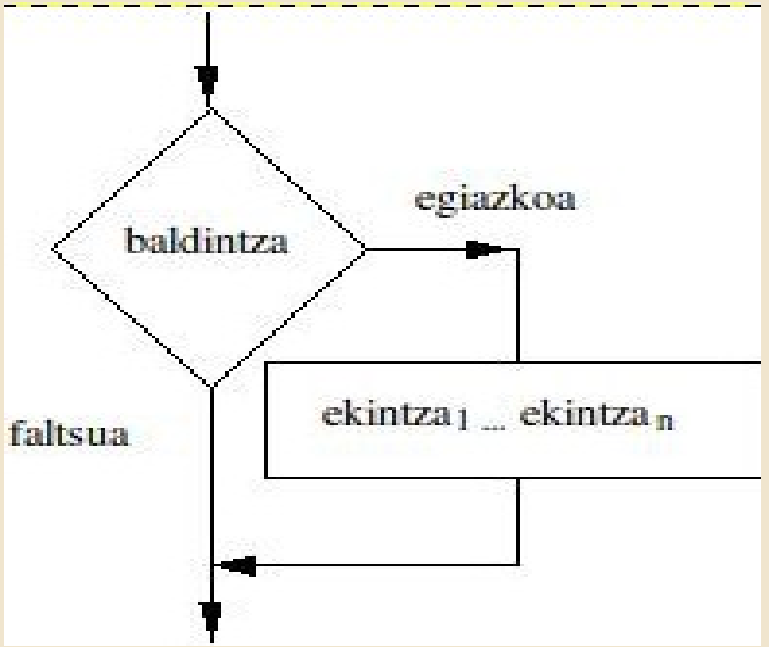
ekintza$_1$ ... ekintza$_n$

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**

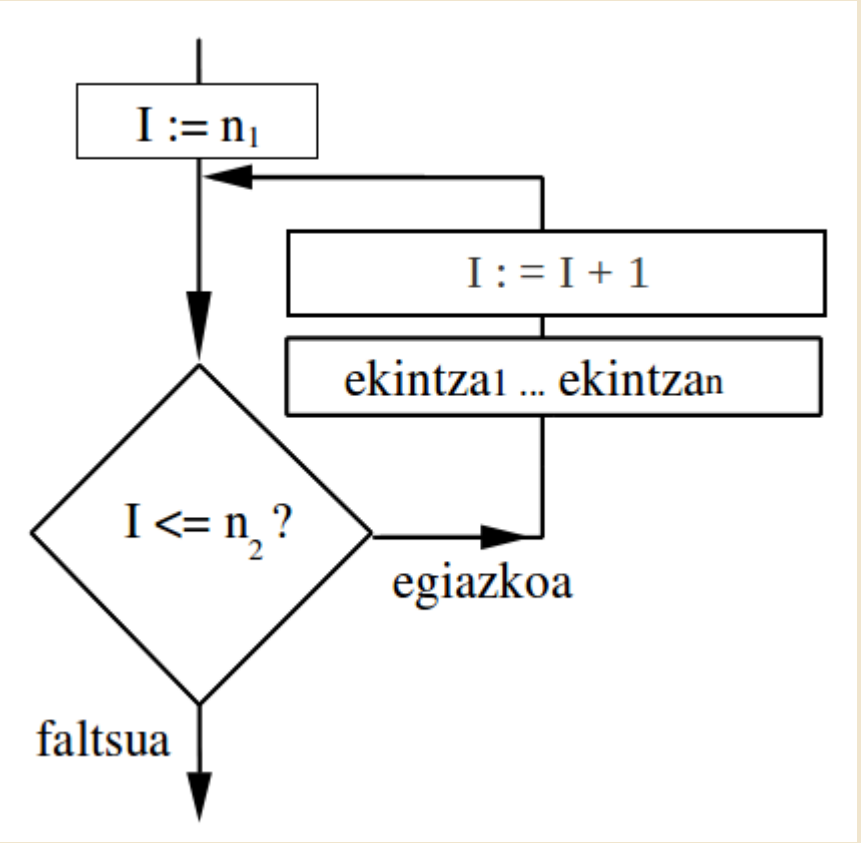
ekintza$_1$ ... ekintza$_n$

end loop ;

Beste aukerak:

for I **in reverse** 1.. 14 **loop**

for I **in 'a'..'z' loop**

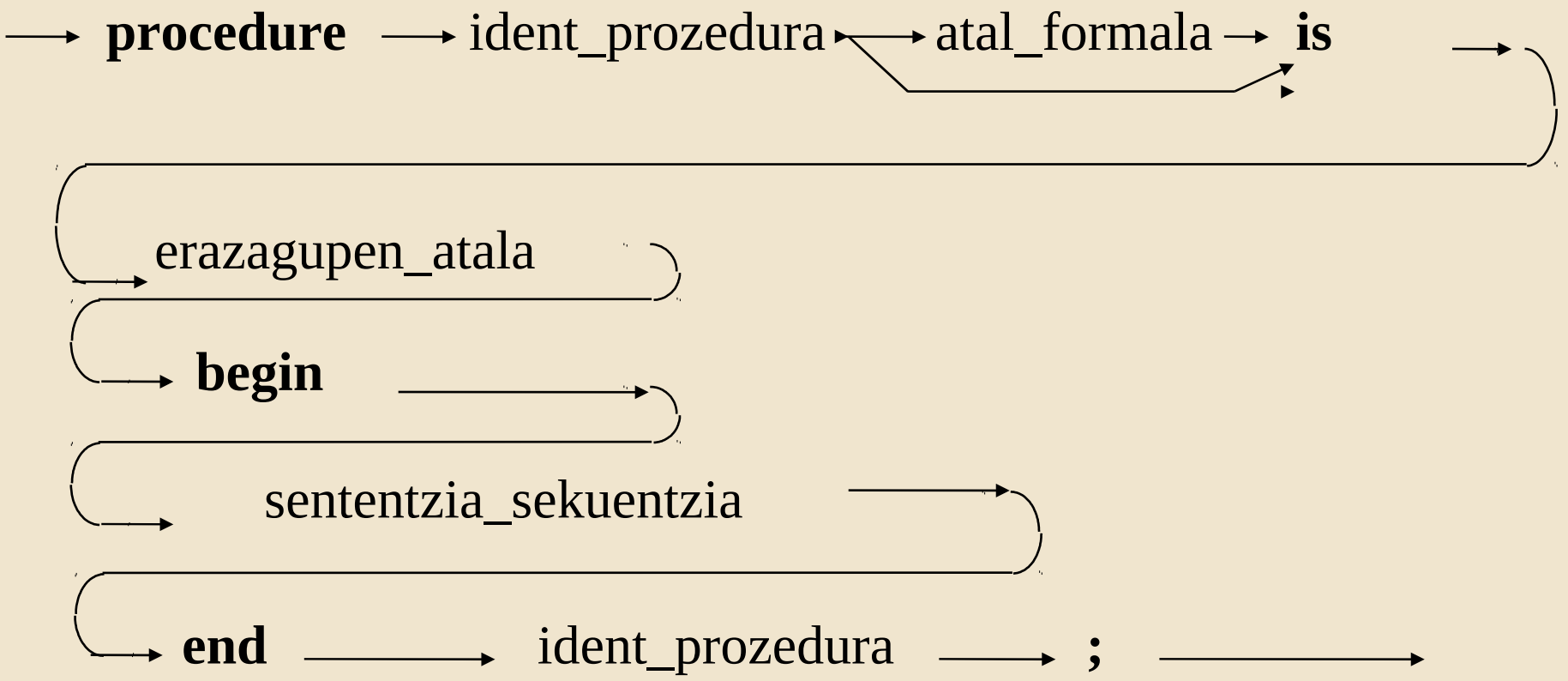


Azpiprograma (sinplifikatua)

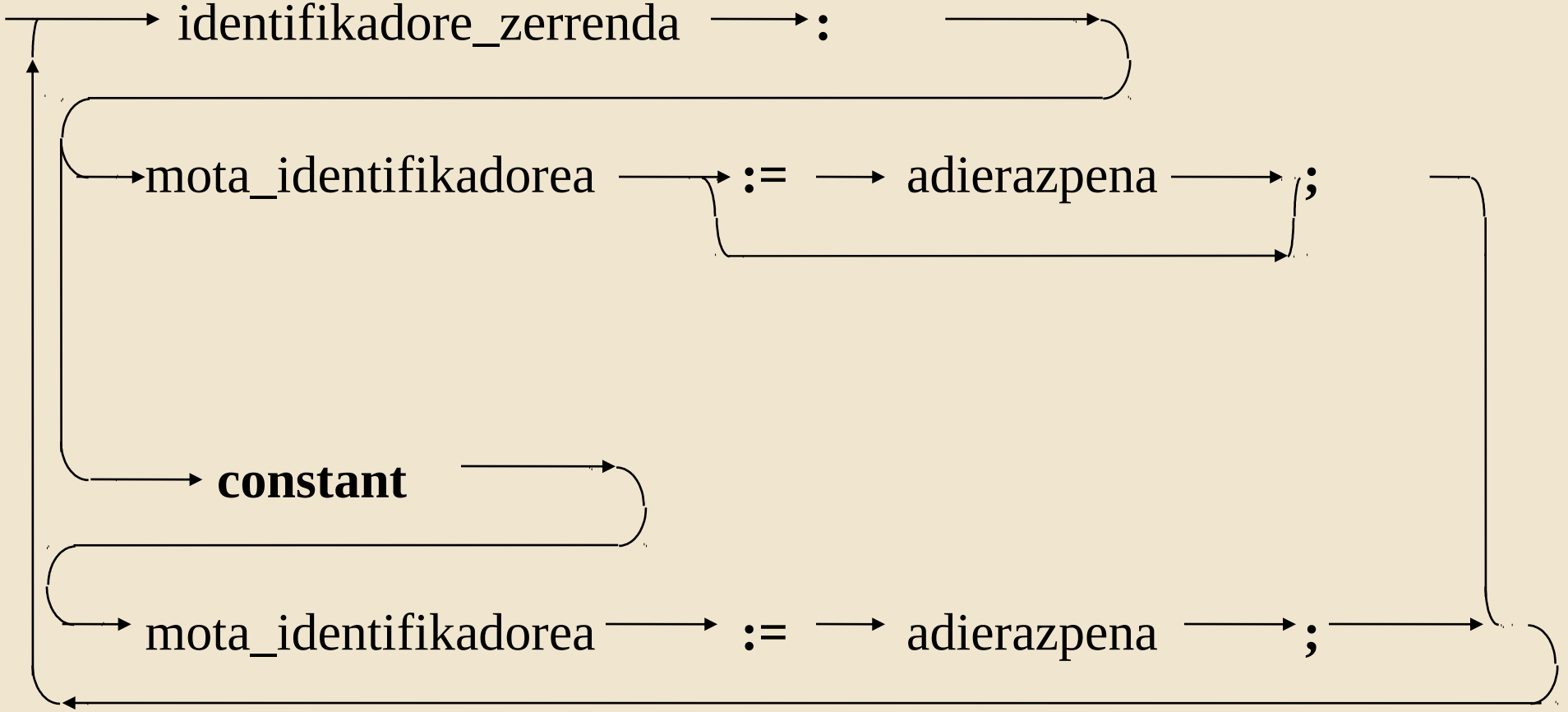
```
procedure ident_prozedura   atal_formala is
   erazagupen_atala
begin
      sententzia_sekuentzia
end ident_prozedura ;
```

Atal formalean parametroak eta beren motak definitzen dira, eta horiek datu (*in*), emaitza (*out*) edo datu-emaitza (*in out*) erakoak diren adierazi behar da.

Azpiprograma (sinplifikatua)



erazagupen_atala (sinplifikatua)



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_Osokoa (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_Osokoa (O); Idatzi_Katea (":");
   Idatzi_Osokoa (M); Idatzi_Katea (":");
   Idatzi_Osokoa (S);
end Ordua_Erakutsi;
```

Azpiprograma adibidea (III)

```
function Faktoriala (N : in Integer) return Integer is
-- Aurrebaldintza: N osoko zenbakia, N > 0
-- Postbaldintza: Fakt zenbaki osokoa, Fakt N zenbakiaren faktoriala da
    Fakt : Integer := 1;
    N, I : Integer;
begin
   I := 1;
   while  I <= N  loop
    Fakt := Fakt * I ;
    I := I + 1 ;
   end loop ;
   return Fakt ;
end Faktoriala ;
```

Azpiprograma bat erabili beste azpiprograma batetik (I)

Azpiprograma lagungarria beste fitxategi batean definitzea, prozeduraren izen berarekin, eta hura erabili nahi duen azpiprogramaren definizioa baino lehen *with* erako sententzia bat erabiltzea beste prozeduraren izenarekin. Kasu honetan, azpiprograma lagungarria beste edozein programatatik ere erabili ahal izango da

b.adb fitxategia:

```
procedure B … is

begin
   …
end B;
```

a.adb fitxategia:

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

Azpiprograma bat erabili beste azpiprograma batetik (II)

Azpiprograma lagungarria barruan definitzea. 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 sarrera eta irteera estandarretik (teklatutik eta pantailan) egin dugu. 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

Kanpo-fitxategiak eta fitxategi-objektuak
Fitxategien izen fisikoa eta izen logikoa

Bi ikuspuntu desberdin:

- Sistema eragiletik: Kanpo-fitxategia.

  Sistema eragileak erabiltzen duen identifikazioa: Izen fisikoa

  Adibidez: "/users/alumnos/soft/jiaa/eusk/lab3/datuak.txt"

- Ada azpiprogrametatik: aldagaia

  Fitxategi-objektua (*File-type motako aldagaia*)

  Ada azpiprogramek erabiltzen duten identifikazioa: Izen logikoa

  Adibidez F1

  (honela erazagutua: *F1: Ada.Text_IO.File_Type)*

Kanpo-fitxategiak eta fitxategi-objektuak
Fitxategien izen fisikoa eta izen logikoa

Irekitze-aginduaren bitartez lotzen dira bi ikuspuntuak

- Fitxategiaren izen logikoa fiskoarekin lotzeko, irekitze agindua erabiltzen da:

Adibidez:

```
Ada.Text_IO.Open( F1, Ada.Text_IO.In_File,
  "/users/alumnos/soft/jiaa/eusk/lab3/datuak.txt")
```

Testu-fitxategiak

Egunero hasten delako

Gizona

bere bakardadean

'E'	'g'	'u'	'n'	'e'	'r'	'o'	' '	'h'	'a'	's'	't'	'e'	'n'	' '	'd'	'e'	'l'	'a'	'k'

'o'	LB	'G'	'i'	'z'	'o'	'n'	'a'	LB	‘b’	'e'	'r'	'e'	' '	'b	'a'	'k'	'a'

'r'	'd'	'a'	'd'	'e'	'a'	'n'	LB	FB

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

Sarrera-irteerarako eragiketa guztiak paketetan daude.

Pakete bat erabili ahal izateko, “**with**” klausula batean aipatu behar n izendatuz dei dakioke.

Sarrera-irteera eragiketak ohiko azpiprograma-deien bidez lortzen 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

Bi fitxategi mota ikusiko ditugu:

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

- **Erregistrozko fitxategiak**: Fitxategi sekuentzial baten elementu guztiak mota berekoak dira. Bata bestearen atzetik irakurri eta idatzi behar dira lehenengotik hasita.

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

Irteera-fitxategi bat sortu

```
procedure Create  (File : in out File_Type;
                   Mode : in     File_Mode;
                         Name : in     String);
```

Non … `type File_Mode is (In_File, Out_File);`

Adibidez:

```
   Ada.Text_IO.Create(F1,
                      Ada.Text_IO.Out_File,
                      "datuak.txt")
```

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

Fitxategi bat itxi

```
procedure Close  (File  : in out File_Type);
```

Erabilera

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

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

Adibidea

```
   Ada.Text_IO.Close(F1);
```

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

Fitxategi bat zabaldu

```
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 bezala erazagutu behar da

Adibidea

```
Ada.Text_IO.Open( F1,Ada.Text_IO.In_File,
                     "users/alumnos/jiaa/datuak.txt")
```

Testu-fitxategiak (Ada.Text_IO paketea).

Bukaera 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).

Bukaera eta Lerroaren kontrola

Irteerako fitxategi batean hurrengo lerrora pasatzeko

```
procedure New_Line (File : in File_Type);
```

Erabilera

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

-- Lerro-bukaerako karakterea sartzen du fitxategian

Testu-fitxategiak (Ada.Text_IO paketea).

Bukaera eta Lerroaren kontrola

Sarrerako fitxategi batean hurrengo lerrora saltatzeko

```
procedure Skip_Line (File : in File_Type);
```

Erabilera

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

-- Lerro-bukaerako karakterearen hurrengo karaktereraino

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 osokoak 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 (F, Kar)
Ada.Integer_Text_IO.Get (F1, N)
Ada.Float_Text_IO.Get (F, X))
```

Testu-fitxategiak (Ada.Text_IO paketea) sarrera

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 kopuru irrakurri den gordetzen da
```

Testu-fitxategiak (Ada.Text_IO paketea) irteera

Sarrerako fitxategi batean idatzi

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

Erabilera

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

Testu-fitxategiak (Ada.Text_IO paketea) irteera

Sarrerako fitxategi batean lerro bat idatzi eta hurrengo lerrora pasa

Ada.Text_IO -- Testu-fitxategiak tratatzeko azpiprogramak dituen paketea

```
procedure Put_Line (File : in File_Type; Item : in String);
```

Erabilera

```
Ada.Text_IO.Put_Line (File => Barne-izena,
Item => String motako aldagaia)
Barne-izena erazagutu behar da
Barne-izena: Ada.Text_IO.File_Type
```

Testu-fitxategiak (Ada.Text_IO paketea) irteera

Sarrerako fitxategia teklatua baldin bada ***Standard_Input da fitxategiaren izena***

Irteerako fitxategia pantaila baldin bada, ***Standard_Output da fitxategiaren izena***

1 eta 2 kasuan, fitxategia sortu eta irekitzea ez da beharrezkoa. Gainera, kasu horietan ere, ***fitxategien izena 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 => Osoko 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 =>Osoko motako ald);
Ada.Float_Text_IO.Get(Item => Erreal motako ald)
```

Adibidea: *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;
```

Adibidea: *kontatu_a_letrak* azpiprograma

```
with Ada.Text_Io, Ada.Integer_Text_Io ;
procedure Kontatu_A_Letrak is
   -- Aurre: S karaktere-sekuentzia bat idatziko da teklatutik,
   --        azken karakterea '.' karakterea da.
   -- Post : kontagailua= zenbat aldiz azaldu da a letra sekuentzia horretan
  Kontagailua : Integer;
  Kar : Character;
begin
   Ada.Text_Io.Put_Line("Sekuentzia batean a letra zenbatetan azaltzen den kontatu nahi dugu.");
   Ada.Text_Io.Put_Line ("Idatzi hainbat karakterea eta bukaeran puntua");
   Kontagailua := 0 ;
   Ada.Text_Io.Get (Kar);
   while Kar /= '.' loop
      if Kar = 'A' or Kar = 'a'
      then Kontagailua := Kontagailua + 1 ;
      end if;
      Ada.Text_Io.Get (Kar);
   end loop ;
   Ada.Integer_Text_Io.Put (Kontagailua);
end Kontatu_A_Letrak;
```

Adibidea: *kontatu_a_letrak_fitx* azpiprograma

```
with Ada.Text_Io, Ada.Integer_Text_Io ;
procedure Kontatu_A_Letrak_Fitx  is
   -- Aurre: S karaktere-sekuentzia bat dago testua.txt fitxategian
   -- Post : Kontagailua = zenbat aldiz azaldu den a letra testua.txt fitxategian
  Kontagailua : Integer;
  Kar : Character;
  F : Ada.Text_IO.File_Type;
begin
   Ada.Text_IO.Open (F, Ada.Text_Io.In_File, "testua.txt");
   Kontagailua := 0 ;
   while not Ada.Text_IO.End_Of_File(F) loop
      Ada.Text_Io.Get (F, Kar);
      if Kar = 'A' or Kar = 'a'
      then Kontagailua := Kontagailua + 1 ;
      end if;
   end loop ;
   Ada.Text_IO.Close (F);
   Ada.Text_Io.Put_Line ("testua.txt fitxategian  a letra zenbatetan azaltzen den:");
   Ada.Integer_Text_Io.Put (Kontagailua);
end Kontatu_A_Letrak_Fitx ;
```

Adibidea:

kontatu_a_letrak_fitx_zelataria azpiprograma

```
with Ada.Text_Io, Ada.Integer_Text_Io ;
procedure Kontatu_A_Letrak_Fitx_Zelataria  is
   -- Aurre: S karaktere-sekuentzia bat dago testua.txt fitxategian.
   --        azken karakterea '.' karakterea da.
   -- Post : kontagailua= zenbat aldiz azaldu da a letra puntuaz amaitzen den sekuentzia horretan
  Kontagailua : Integer;
  Kar : Character;
  F : Ada.Text_IO.File_Type;
begin
   Ada.Text_IO.Open (F, Ada.Text_Io.In_File, "testua.txt");
   Ada.Text_Io.Get (F, Kar);
   Kontagailua := 0 ;
   while Kar /= '.' loop
      if Kar = 'A' or Kar = 'a'
      then Kontagailua := Kontagailua + 1 ;
      end if;
      Ada.Text_Io.Get (F, Kar);
   end loop ;
   Ada.Text_IO.Close (F);
   Ada.Text_Io.Put_Line ("testua.txt fitxategian  a letra zenbatetan azaltzen den:");
   Ada.Integer_Text_Io.Put (Kontagailua);
end Kontatu_A_Letrak_Fitx_Zelataria ;
```

Idatzi_Karakterea azpiprograma

idatzi_karakterea.adb

```
with Ada.Text_IO;
procedure Idatzi_Karakterea (K : in Character) is
begin
   Ada.Text_IO.Put (K);
end Idatzi_Karakterea;
```

Idatzi_Katea azpiprograma

idatzi_katea.adb

```
with Ada.Text_IO;
procedure Idatzi_Katea (K : in String) is
begin
   Ada.Text_IO.Put (K);
end Idatzi_Katea;
```

Idatzi_Osokoa azpiprograma

idatzi_osokoa.adb

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

Testuzkoak ez diren fitxategiak

(eduki osagarriak dira hemendik aurrerakoak)

- elementu sekuentzia bat
- elementu guztiak mota berberekoak dira
- adierazpide bitarrean gordeta
- eraginkorragoa testu-fitxategia baino (memorian eta denboran)
- baina begi bistan ulergaitza (ezin dira testu gisa editatu)
- bi eratakoak:
  - atzipen sekuentzialekoak
  - atzipen zuzenekoak

Testuzkoak ez diren fitxategiak

Fitxategiko elementuak *Elementu* motakoak badira,

fitxategia erabiltzeko azpiprogramak pakete batean sortu behar dira honela:

```
package Elem_SI is new Sequential_IO (Elementu);
```

Elem_SI izena eman diogu paketeari

Testuzkoak ez diren fitxategiak

Fitxategien gestioa (ireki eta itxi)

```
procedure Create  (File : in out File_Type;
          Mode : in File_Mode := Out_File;
          Name : in String );
procedure Open    (File : in out File_Type;
          Mode : in File_Mode;
          Name : in String);
procedure Close   (File : in out File_Type);
```

Testuzkoak ez diren fitxategiak

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 Adibidea

Erabilera (ikusitako procedura/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: in Motak.Elementu);
function    Elementu_SI.End_of_File    (File:    Elementu_SI.File_Type)
                                 return Boolean;
```

Testuzkoak ez diren fitxategiak Adibidea

Erabilera (ikusitako procedura/funtzio batzuk) (2)

```
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.Create (File:in out Elementu_SI.File_Type;
                              Mode: in Elementu_SI.File_Mode := Out_File;
                              Name: in String );
procedure Elementu_SI.Open (File: in out Elementu_SI.File_Type;
                            Mode: in Elementu_SI.File_Mode := In_File;
                            Name: in String );
procedure Elementu_SI.Close (File: in out Elementu_SI.File_Type);
```