



Oinarrizko Programazioa

5. Programazio-lengoaiei erabilera: ADA lengoaia



Oinarrizko Programazioa

Gai zerrenda:

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

Eduki osagarriak:

Testuzkoak ez diren fitxategiak



Sintaxia eta Semantika

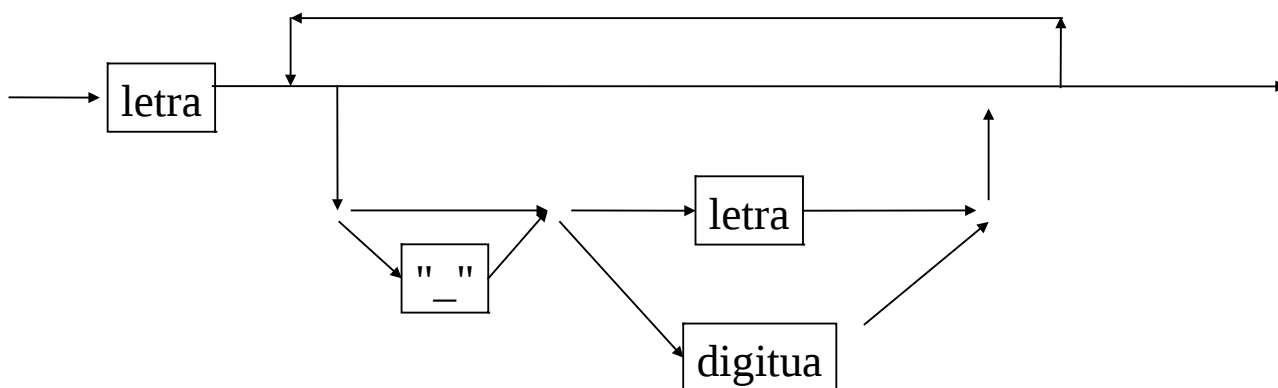
- SINTAXIA:
sententziak idazteko arau-multzoa
 - sintaxi-diagramen bidez
 - EBNF erregelen bidez
- SEMANTIKA:
sententzia horien esangura

Sintaxi-diagramak

- Sintaxi-diagramen bidez lengoaien idazkera definitzen da.
- Sintaxi-diagrama bakoitzak eraikuntza bakun baten sintaxia definitzen du.
- Diagramaren sarreratik bere irteerarainoko edozein bidek, gezen norantza jarraituz, markatzen du eraikuntza sintaktiko egokia.
- 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 lengoiaren sintaxiaren definizio osoa :

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



Adaren sintaxia

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

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

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

11

Oinarrizko Programazioa

2014/09/26

Adako hitz erreserbatuen zerrenda

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

12

Oinarrizko Programazioa

2014/09/26

Banatzaileak

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

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

13

Oinarrizko Programazioa

2014/09/26

Programazio-estiloa

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

14

Oinarrizko Programazioa

2014/09/26

Datuen adierazpidea Adan

Oinarrizko datu-motak

- Oinarrizkotzat hartzen ditugu balio bakar bat hartzen dutenak:

<u>Datu-mota</u>	<u>Balioak</u>	<u>ADAn</u>
<i>Osokoa</i>	osoko zenbakiak	Integer
<i>Errealak</i>	zenbaki errealak	Float
<i>Karakterea</i>	karakterek	Character
<i>Boolearra</i>	true eta false	Boolean
<i>Katea</i>	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_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

Baldintzazko egitura

```
baldin baldintza orduan  
    ekintza1... ekintzan
```

```
ambaldin
```

```
baldin baldintza orduan  
    ekintza1... ekintzan
```

```
bestela
```

```
    ekintza21... ekintza2n
```

```
ambaldin
```

```
if baldintza then  
    ekintza1... ekintzan
```

```
end if ;
```

```
if baldintza then  
    ekintza1... ekintzan
```

```
else
```

```
    ekintza21... ekintza2n
```

```
end if ;
```

Kontrol-egiturak Adan

Baldintzazko egitura (II)

```
baldin b1 orduan  
    ekintza11... ekintza1n
```

```
bestela baldin b2 orduan  
    ekintza21... ekintza2n
```

```
bestela baldin b3 orduan  
    ekintza31... ekintza3n
```

```
bestela  
    ekintza41... ekintza4n
```

```
ambaldin
```

```
if b1 then  
    ekintza11... ekintza1n
```

```
elsif b2 then  
    ekintza21... ekintza2n
```

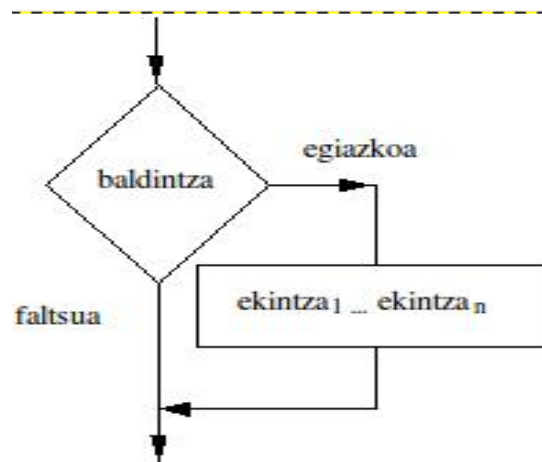
```
elsif b3 then  
    ekintza31... ekintza3n
```

```
else  
    ekintza41... ekintza4n  
end if ;
```

Kontrol-egiturak Adan Iterazioa

bitartean baldintza **egin**
 ekintza₁ ... ekintza_n
ambitartean

```
while baldintza loop
    ekintza1 ... ekintzan
end loop ;
```



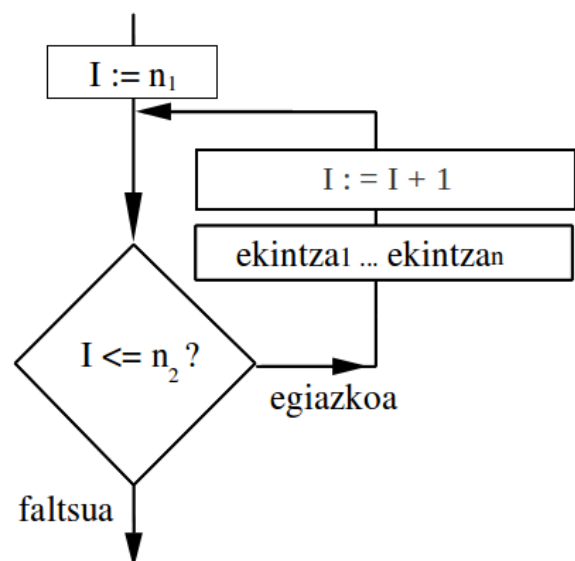
Kontrol-egiturak Adan Aldi kopuru jakineko iterazioa

egin I guztietarako n_1 tik n_2 raino
 ekintza₁ ... ekintza_n
amguztietarako

```
for I in  $n_1..n_2$  loop
    ekintza1 ... ekintzan
end loop ;
```

Beste aukerak:

```
for I in reverse 1..14 loop
for I in 'a'..'z' loop
```

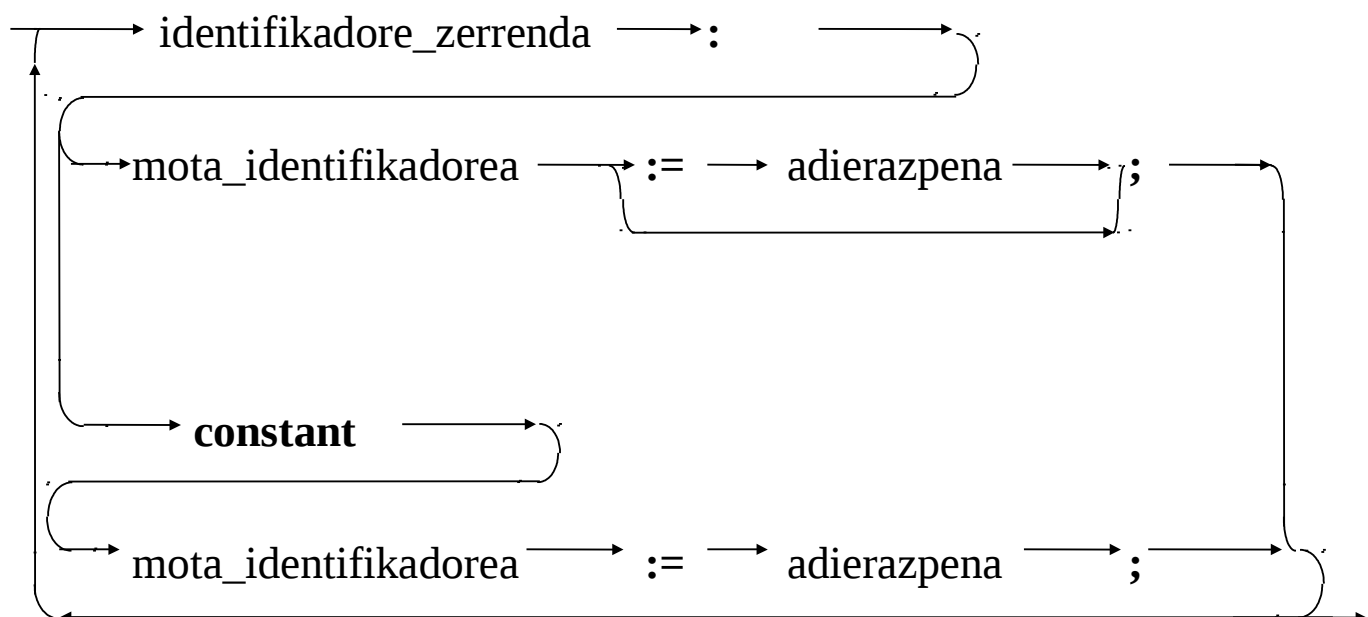


Azpiprogramak (sinplifikatua)

```
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.

Azpiprogramak 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-datuekin 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 adibidea (III)

```
function Faktoriala (N : in Integer) return Integer is  
-- Aurrebaldintza: N (osoa), N > 0  
-- Postbaldintza: Eraitza (osoa), Eraitza = N zenbakiaren faktoriala  
    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 ;
```

27

Oinarrizko Programazioa

2014/09/26

Azpiprograma bat erabili beste azpiprograma batetik (I)

- Aukera1:
 - Azpiprograma lagungarria (prozedura edo funtzioa) fitxategi batean definitzen da
 - Fitxategiaren izena = azpiprogramaren izena
 - Azpiprogramak beste edozein kanpoko programatan erabil daitezke
 - Azpiprograma lagungarria erabili nahi duen azpiprograma nagusiaren fitxategian, definizioa baino lehen, *with* erako sententzia bat erabiltzen da.

b.adb fitxategia:

```
procedure B ... is  
  
begin  
    ...  
end B;
```

a.adb fitxategia:

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

28

Oinarrizko Programazioa

2014/09/26

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
begin
    procedure B ... is
    begin
    end B;
end A;
```

Kanpo-memoriako datu-egiturak

- Orain arte, datuen irakurketa eta idazketa sarrera eta irteera estandarretik (teklaturatik 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

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`)

31

Oinarrizko Programazioa

2014/09/26

Kanpo-fitxategiak eta fitxategi-objektuak.

Fitxategien izen fisikoa eta izen logikoa (II)

Irekitze-aginduaren bitartez lotzen dira bi ikuspuntuak

- Irekitze-aginduak lotzen ditu fitxategiaren izen logikoa eta fisikoa:

Adibidez:

```
Ada.Text_IO.Open(F1,  
                  Ada.Text_IO.In_File,  
                  "users/jipsogaklab3/datuak.txt")  
Ada.Text_IO.Open(F2,  
                  Ada.Text_IO.In_File,  
                  "jarduerak0.txt")
```

32

Oinarrizko Programazioa

2014/09/26

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 >
```

'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										

33


Oinarrizko Programazioa

2014/09/26

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

Irteera-fitxategi bat sortu

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

Erabilera:

```
Ada.Text_IO.Create  
(File => Barne-izena,  
 Mode => Ada.Text_IO.Out_File,  
 Name => Kanpo-izena)  
-- Barne-izena: programa barruan fitxategia izendatzeko aldagaia.  
-- Kanpo-izena, fitxategiaren izen fisikoa (benetako izena).  
-- Barne-izena aldagaia erazagutu behar da  
-- eta Ada.Text_IO.File_Type motakoa izan beharko da
```



Testu-fitxategiak (*Ada.Text_IO* paketea)

Ireki eta itxi fitxategiak (II)

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 gisa erazagutu behar da

Adibidea

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

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) Bukaeraren eta lerroaren kontrola (II)


Irteerako fitxategi batean hurrengo lerroa 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) Bukaeraren eta lerroaren kontrola (III)

Sarrerako fitxategi batean hurrengo lerroa 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 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 lerroa 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

Irteerako 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 osoak 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 => Oso motako aldagaia)
Ada.Float_Text_IO.Put (File => Barne-izena, Item => Float motako aldagaia)
```



Testu-fitxategiak (*Ada.Text_IO* paketea)

Irteera (II)

Irteerako fitxategi batean lerro bat idatzi eta hurrengo lerroa 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 izan da
Barne-izena: Ada.Text_IO.File_Type
```

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)
```

45

Oinarrizko Programazioa

2014/09/26

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;
```

46

Oinarrizko Programazioa

2014/09/26



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_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 (II)

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 (III)

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 (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;
```



Testuzkoak ez diren fitxategiak (VI)

Erabilera (ikusitako prozedura/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);
    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);
```