

# Informe Compiladors

José Ruiz Bravo, 123456789 <joseruizbravo@gmail.com>,  
Biel Moyà Alcover, 43142617E <bilibiel@gmail.com>,  
Álvaro Medina Ballester, 43176576X <alvaro@comiendolimones.com>

20 de novembre de 2009

## Resum

Compilador *compilemon* creat amb el llenguatge Ada. Està compost per un subconjunt bàsic d'instruccions en Ada.

## 1 Anàlisi Sintàctica

### 1.1 Gramàtica del nostre llenguatge

*programa* → *procediment*

*procediment* → **PC\_PROCEDURE** *encap* **PC\_IS**  
                  *declaracions*  
                  **PC\_BEGIN**  
                  *bloc*  
                  **PC\_END** *identificador*;

*encap* → *identificador* *args*

*args* → (*lparam*)  
      | λ

*lparam* → *lparam* ; *param*  
      | *param*

*param* → *identificador* : *mode* *identificador*

*mode* → **PC\_IN**

$$\begin{array}{l} | \text{PC\_OUT} \\ | \text{PC\_IN PC\_OUT} \end{array}$$

$$\begin{array}{l} \text{declaracions} \rightarrow \text{declaracions declaracio} \\ | \lambda \end{array}$$

$$\begin{array}{l} \text{declaracio} \rightarrow \text{dec\_var} \\ | \text{dec\_constant} \\ | \text{dec\_tipus} \\ | \text{programa} \end{array}$$

– Manual d’usuari variables

$$\text{dec\_var} \rightarrow \text{lid} : \text{identificador};$$

$$\begin{array}{l} \text{lid} \rightarrow \text{lid, identificador} \\ | \text{identificador} \end{array}$$

– Manual d’usuari constant

$$\text{dec\_constant} \rightarrow \text{identificador} : \text{PC\_CONSTANT identificador} := \text{valor};$$

$$\begin{array}{l} \text{valor} \rightarrow \text{lit} \\ | - \text{lit} \end{array}$$

– Manual d’usuari tipus

$$\begin{array}{l} \text{dec\_tipus} \rightarrow \text{dec\_subrang} \\ | \text{dec\_registre} \\ | \text{dec\_coleccio} \end{array}$$

$$\begin{array}{l} \text{dec\_subrang} \rightarrow \text{PC\_TYPE identificador PC\_IS PC\_NEW identificador} \\ \text{PC\_RANGE valor .. valor}; \end{array}$$

$$\begin{array}{l} \text{dec\_registre} \rightarrow \text{PC\_TYPE identificador PC\_IS PC\_RECORD} \\ \text{ldc} \\ \text{PC\_END PC\_RECORD}; \end{array}$$

$$\begin{array}{l} \text{ldc} \rightarrow \text{ldc dc} \\ | \text{dc} \end{array}$$

$$\text{dc} \rightarrow \text{identificador} : \text{identificador};$$

– Tipus colecció (*array*)

$\overline{dec\_coleccio} \rightarrow \mathbf{PC\_TYPE}$  identificador  $\mathbf{PC\_IS\ PC\_ARRAY}$   
                   (*lid*)  $\mathbf{PC\_OF}$  identificador;

*lid*             $\rightarrow$  *lid*, identificador  
                   |    identificador

– Bloc d'instruccions

*bloc*            $\rightarrow$  *bloc sent*  
                   |    *sent*

*sent*            $\rightarrow$  *sassig*  
                   |    *scond*  
                   |    *srep*  
                   |    *crida\_proc*  
                   |     $\lambda$

*sassig*         $\rightarrow$  *referencia := expressio*;

*scond*          $\rightarrow$   $\mathbf{PC\_IF}$  *expressio*  $\mathbf{PC\_THEN}$   
                    $\overline{bloc}$   
                    $\mathbf{PC\_END\ PC\_IF}$ ;  
                   |    $\mathbf{PC\_IF}$  *expressio*  $\mathbf{PC\_THEN}$   
                    $\overline{bloc}$   
                    $\mathbf{PC\_ELSE}$   
                    $\overline{bloc}$   
                    $\mathbf{PC\_END\ PC\_IF}$ ;

*srep*           $\rightarrow$   $\mathbf{PC\_WHILE}$  *expressio*  $\mathbf{PC\_LOOP}$   
                    $\overline{bloc}$   
                    $\mathbf{PC\_END\ PC\_WHILE}$ ;

*crida\_proc*    $\rightarrow$  *referencia*;

*referencia*    $\rightarrow$  identificador  
                   |    *referencia*.identificador  
                   |    *referencia* (*prparam*)

*prparam*       $\rightarrow$  *expressio*  
                   |    *expressio*, *prparam*

*expressio* → *expressio* + *expressio*  
| *expressio* − *expressio*  
| *expressio* \* *expressio*  
| *expressio* / *expressio*  
| *expressio* **PC\_MOD** *expressio*  
| *expressio* > *expressio*  
| *expressio* < *expressio*  
| *expressio* ≥ *expressio*  
| *expressio* ≤ *expressio*  
| *expressio* ≠ *expressio*  
| *expressio* = *expressio*  
| − *expressio*  
| *expressio* && *expressio*  
| *expressio* || *expressio*  
| **PC\_NOT** *expressio*  
| (*expressio*)  
| *referencia*  
| *lit*

## 2 Anàlisi Lèxica

### 2.1 Descripció del lèxic: *compilemon.l*

```
1  -- Macros
2
3  lletra          [A-Za-z]
4  digit          [0-9]
5  separadors     [\n\b\t\f]
6  caracter       \' [^\n\t] \',
7
8
9  %%
10
11
12 -- Paraules clau
13
14 procedure       {mt_atom(tok_begin_line, tok_begin_col,
15                        yylval); return pc_procedure;}
16
17 begin           {mt_atom(tok_begin_line, tok_begin_col,
18                        yylval); return pc_begin;}
19
20 while           {mt_atom(tok_begin_line, tok_begin_col,
21                        yylval); return pc_while;}
22
23 if              {mt_atom(tok_begin_line, tok_begin_col,
24                        yylval); return pc_if;}
25
26 else            {mt_atom(tok_begin_line, tok_begin_col,
27                        yylval); return pc_else;}
28
29 end             {mt_atom(tok_begin_line, tok_begin_col,
30                        yylval); return pc_end;}
31
32 loop            {mt_atom(tok_begin_line, tok_begin_col,
33                        yylval); return pc_loop;}
34
35 constant        {mt_atom(tok_begin_line, tok_begin_col,
36                        yylval); return pc_constant;}
37
38 type            {mt_atom(tok_begin_line, tok_begin_col,
39                        yylval); return pc_type;}
```

```
40
41 array      {mt_atom(tok_begin_line, tok_begin_col,
42                  yylval); return pc_array;}
43
44 record     {mt_atom(tok_begin_line, tok_begin_col,
45                  yylval); return pc_record;}
46
47 is         {mt_atom(tok_begin_line, tok_begin_col,
48                  yylval); return pc_is;}
49
50 then       {mt_atom(tok_begin_line, tok_begin_col,
51                  yylval); return pc_then;}
52
53 not        {mt_atom(tok_begin_line, tok_begin_col,
54                  yylval); return pc_not;}
55
56 in         {mt_atom(tok_begin_line, tok_begin_col,
57                  yylval); return pc_in;}
58
59 out        {mt_atom(tok_begin_line, tok_begin_col,
60                  yylval); return pc_out;}
61
62 new        {mt_atom(tok_begin_line, tok_begin_col,
63                  yylval); return pc_new;}
64
65 null       {mt_atom(tok_begin_line, tok_begin_col,
66                  yylval); return pc_null;}
67
68 of         {mt_atom(tok_begin_line, tok_begin_col,
69                  yylval); return pc_of;}
70
71 mod        {mt_atom(tok_begin_line, tok_begin_col,
72                  yylval); return pc_mod;}
73
74 range      {mt_atom(tok_begin_line, tok_begin_col,
75                  yylval); return pc_range;}
76
77 and        {mt_atom(tok_begin_line, tok_begin_col,
78                  yylval); return pc_or;}
79
80 or         {mt_atom(tok_begin_line, tok_begin_col,
81                  yylval); return pc_and;}
82
```

```
83
84 --Simbols
85
86 ":"      {mt_atom(tok_begin_line, tok_begin_col,
87                  yylval); return s_assignacio;}
88
89 ":"      {mt_atom(tok_begin_line, tok_begin_col,
90                  yylval); return s_dospunts;}
91
92 ";"      {mt_atom(tok_begin_line, tok_begin_col,
93                  yylval); return s_final;}
94
95 ","      {mt_atom(tok_begin_line, tok_begin_col,
96                  yylval); return s_coma;}
97
98 "("      {mt_atom(tok_begin_line, tok_begin_col,
99                  yylval); return s_parentesiobert;}
100
101 ")"      {mt_atom(tok_begin_line, tok_begin_col,
102                  yylval); return s_parentesitancat;}
103
104 ".."     {mt_atom(tok_begin_line, tok_begin_col,
105                  yylval); return s_puntsrang;}
106
107 "."      {mt_atom(tok_begin_line, tok_begin_col,
108                  yylval); return s_puntrec;}
109
110
111 --Operadors
112
113 "<"      {mt_atom(tok_begin_line, tok_begin_col,
114                  yylval); return op_menor;}
115
116 "<="    {mt_atom(tok_begin_line, tok_begin_col,
117                  yylval); return op_menorigual;}
118
119 ">="    {mt_atom(tok_begin_line, tok_begin_col,
120                  yylval); return op_majorigual;}
121
122 ">"      {mt_atom(tok_begin_line, tok_begin_col,
123                  yylval); return op_major;}
124
125 "="      {mt_atom(tok_begin_line, tok_begin_col,
```

```
126         yylval); return op_igual;}
127
128     "/"= "        {mt_atom(tok_begin_line, tok_begin_col,
129         yylval); return op_distint;}
130
131     "+"          {mt_atom(tok_begin_line, tok_begin_col,
132         yylval); return op_suma;}
133
134     "-"          {mt_atom(tok_begin_line, tok_begin_col,
135         yylval); return op_resta;}
136
137     "*"          {mt_atom(tok_begin_line, tok_begin_col,
138         yylval); return op_multiplicacio;}
139
140     "/"          {mt_atom(tok_begin_line, tok_begin_col,
141         yylval); return op_divisio;}
142
143
144 --Digit
145
146 {digit}+        {mt_numero(tok_begin_line, tok_begin_col,
147         yytext, yylval); return const;}
148
149
150 --Lletra
151
152 {caracter}      {mt_caracter(tok_begin_line, tok_begin_col,
153         yytext, yylval); return const;}
154
155
156 --String
157
158 \"[^\n\t]*\"     {mt_string(tok_begin_line, tok_begin_col,
159         yytext, yylval); return const;}
160
161
162 --Identificador
163
164 {lletra}({digit}|{lletra})*
165         {mt_identificador(tok_begin_line,
166         tok_begin_col, yytext, yylval);
167         return id;}
168
```



```
169
170
171 --Comentaris
172
173 "-- "[^\n]*      {null;}
174
175
176 --Separadors
177
178 " "              {null;}
179
180 {separadors}*    {null;}
181
182
183 --Error
184
185 .                {return error;}
186
187
188
189 %%
190
191
192
193 with      decls.d_taula_de_noms ,
194           d_token ,
195           decls.d_atribut;
196
197
198 use       decls.d_taula_de_noms ,
199           d_token ,
200           decls.d_atribut;
201
202
203 package u_lexica is
204
205         yylval: atribut;
206         tn : taula_de_noms;
207         function YYLex return token;
208
209 end u_lexica;
210
211
```

```
212  
213 package body u_lexica is  
214  
215 ##  
216  
217 begin  
218  
219         tbuida(tn);  
220  
221 end u_lexica;
```

## 2.2 Taula de noms

### 2.2.1 Fitxer *decls-d\_taula\_de\_noms.ads*

```
1  -- -----
2  --   Paquet de declaracions de la taula de noms
3  -- -----
4  --   Versio   :    0.2
5  --   Autors   :    Jose Ruiz Bravo
6  --               Biel Moya Alcover
7  --               Alvaro Medina Ballester
8  -- -----
9  --   Especificacio de l'estructura necessaria
10 -- per el maneig de la taula de noms i dels metodes
11 -- per tractar-la.
12 --
13 -- -----
14
15 WITH      decls.dgenerals;
16
17 USE        decls.dgenerals;
18
19
20 PACKAGE decls.d_taula_de_noms IS
21
22     PRAGMA pure;
23
24     -- Excepcions
25     E_Tids_Plana : EXCEPTION;
26     E_Tcar_Plana : EXCEPTION;
27
28     TYPE taula_de_noms IS LIMITED PRIVATE;
29
30     TYPE id_nom IS NEW integer
31         RANGE 0 .. max_id;
32
33     TYPE rang_dispersio IS NEW integer
34         RANGE 0 .. max_id;
35
36     TYPE rang_tcar IS NEW integer
37         RANGE 0 .. (long_num_ident*max_id);
38
39     -- Valor nul per al tipus id_nom
40     id_nul : CONSTANT id_nom := 0;
```

```
41
42
43     PROCEDURE tbuida
44         (tn : OUT taula_de_noms);
45
46     PROCEDURE posa_id
47         (tn : IN OUT taula_de_noms;
48          idn : OUT id_nom;
49          nom : IN string);
50
51     PROCEDURE posa_tc
52         (tn : IN OUT taula_de_noms;
53          nom : IN string);
54
55     PROCEDURE posa_str
56         (tn : IN OUT taula_de_noms;
57          ids : OUT rang_tcar;
58          s : IN string);
59
60     FUNCTION cons_nom
61         (tn : IN taula_de_noms;
62          idn : IN id_nom)
63         RETURN string;
64
65     FUNCTION cons_str
66         (tn : IN taula_de_noms;
67          ids : IN rang_tcar)
68         RETURN string;
69
70     FUNCTION fdisp_tn
71         (nom : IN string)
72         RETURN rang_dispersio;
73
74
75 PRIVATE
76
77
78     TYPE taula_dispersio IS ARRAY (rang_dispersio)
79         OF id_nom;
80
81     TYPE t_identificador IS RECORD
82         pos_tcar : rang_tcar;
83         seguent : id_nom;
```

```
84         long_paraula : Natural;
85     END RECORD;
86
87     TYPE taula_identificadors IS ARRAY
88         (1 .. id_nom'Last) OF t_identificador;
89
90     TYPE taula_caracters IS ARRAY (rang_tcar)
91         OF character;
92
93     TYPE taula_de_noms IS RECORD
94         td : taula_dispersio;
95         tid : taula_identificadors;
96         tc : taula_caracters;
97         nid : id_nom;
98         ncar : rang_tcar;
99     END RECORD;
100
101
102 END decls.d_taula_de_noms;
```

**2.2.2 Fitxer *decls-d\_taula\_de\_noms.adb***

```

1  -- -----
2  --   Paquet de declaracions de la taula de noms
3  -- -----
4  --   Versio   :    0.3
5  --   Autors   :    Jose Ruiz Bravo
6  --               Biel Moya Alcover
7  --               Alvaro Medina Ballester
8  -- -----
9  --   Implementacio dels procediments per al
10 -- tractament de la taula de noms:
11 --
12 --           - Buidat de la taula
13 --           - Insercio
14 --           - Insercio d'strings
15 --           - Consulta
16 --           - Funcio de hash
17 --
18 -- -----
19
20
21 PACKAGE BODY decls.d_taula_de_noms IS
22
23     -- Donam els valors per defecte de cada camp.
24     PROCEDURE tbuida
25         (tn : OUT taula_de_noms) IS
26
27     BEGIN
28         FOR i IN tn.td'RANGE LOOP
29             tn.td(i) := id_nul;
30         END LOOP;
31
32         tn.nid := 1;
33         tn.ncar := 1;
34         tn.tid(1).seguent := id_nul;
35     END tbuida;
36
37
38
39     PROCEDURE posa_id    (tn : IN OUT taula_de_noms;
40                          idn : OUT id_nom;
41                          nom : IN string) IS

```



```
85
86     tn.nid := tn.nid + 1;
87
88     FOR i IN 1 .. nom'Length LOOP
89         tn.tc(tn.ncar) := nom(i);
90         tn.ncar := tn.ncar + 1;
91     END LOOP;
92
93     --tn.tc(tn.ncar) := Ascii.nul;
94     --tn.ncar := tn.ncar + 1;
95
96 END posa_tc;
97
98
99
100 PROCEDURE posa_str (tn : IN OUT taula_de_noms;
101                    ids : OUT rang_tcar;
102                    s : IN string) IS
103
104     -- Index per recorre la taula de caracters.
105     jdx : rang_tcar;
106
107 BEGIN
108
109     -- Excepcio per a controlar tc plena
110     IF (tn.ncar + s'Length) > rang_tcar'Last THEN
111         RAISE E_Tcar_Plenu;
112     END IF;
113
114     -- Omplim la taula de caracters, desde la primera
115     -- posicio lliure 'ncar'.
116     jdx := tn.ncar;
117     ids := tn.ncar;
118
119     FOR i IN 1..s'Length LOOP
120         tn.tc(jdx) := s(i);
121         jdx := jdx + 1;
122     END LOOP;
123
124     tn.ncar := jdx + 1;
125     tn.tc(jdx) := Ascii.nul;
126
127
```



```
128     END posa_str;
129
130
131
132     FUNCTION cons_nom    (tn : IN taula_de_noms;
133                          idn : IN id_nom)
134         RETURN string IS
135
136         It1, It2 : Rang_Tcar;
137
138     BEGIN
139
140         It1 := Tn.Tid(Idn).Pos_Tcar;
141         It2 := Rang_Tcar(Tn.Tid(Idn).Long_Paraula);
142         It2 := It2 + It1 - 1;
143
144         RETURN String(Tn.Tc(it1 .. it2));
145
146
147     END cons_nom;
148
149
150
151     FUNCTION cons_str    (tn : IN taula_de_noms;
152                          ids : IN rang_tcar)
153         RETURN string IS
154
155         idx : rang_tcar;
156
157     BEGIN
158
159         idx := ids;
160         WHILE (tn.tc(idx) /= Ascii.nul) LOOP
161             idx := idx+1;
162         END LOOP;
163
164         RETURN string(tn.tc(ids..idx-1));
165
166     END cons_str;
167
168
169     FUNCTION fdisp_tn (nom : IN string)
170         RETURN rang_dispersio IS
```

```
171
172     a : ARRAY (nom'RANGE) OF integer;
173     r : ARRAY (1..2*nom'Last) OF integer;
174
175     k, c, m, n : integer;
176
177     base : CONSTANT Integer :=
178         Character'Pos(Character'Last)+1;
179
180 BEGIN
181
182     n := nom'Last;
183     m := nom'Length;
184
185     FOR i IN 1..n LOOP
186         a(i) := character'Pos(nom(i));
187     END LOOP;
188
189     FOR i IN 1..2*n LOOP
190         r(i) := 0;
191     END LOOP;
192
193     FOR i IN 1..n LOOP
194         c := 0; k := i - 1;
195         FOR j IN 1..n LOOP
196             c := c + r(k+j) + a(i) + a(j);
197             r(k+j) := c MOD base;
198             c := c/base;
199         END LOOP;
200         r(k+n+1) := r(k+n+1) + c;
201     END LOOP;
202
203     c := (r(n+1) * base + r(n)) MOD (max_id);
204
205     RETURN rang_dispersio(c);
206
207 END fdisp_tn;
208
209
210 END decls.d_taula_de_noms;
```

## 2.3 Tokens i atributs

### 2.3.1 Fitxer *d\_token.ads*

```
1  -- -----
2  --   Paquet de declaracions dels tokens
3  -- -----
4  --   Versio       :      0.2
5  --   Autors       :      Jose Ruiz Bravo
6  --                  Biel Moya Alcover
7  --                  Alvaro Medina Ballester
8  -- -----
9  --       Definicio del tipus token.
10 --
11 -- -----
12
13 PACKAGE d_token IS
14
15     TYPE token IS (pc_procedure ,
16                    pc_begin ,
17                    pc_while ,
18                    pc_if ,
19                    pc_else ,
20                    pc_end ,
21                    pc_do ,
22                    pc_constant ,
23                    pc_type ,
24                    pc_array ,
25                    pc_record ,
26                    pc_is ,
27                    pc_then ,
28                    pc_not ,
29                    pc_in ,
30                    pc_out ,
31                    pc_new ,
32                    pc_null ,
33                    pc_of ,
34                    pc_mod ,
35                    pc_range ,
36                    pc_and ,
37                    pc_or ,
38                    s_assignacio ,
39                    s_dospunts ,
40                    s_final ,
```

```
41         s_coma ,
42         s_parentesiobert ,
43         s_parentesitancat ,
44         s_puntsrang ,
45         s_puntrec ,
46         op_menor ,
47         op_menorigual ,
48         op_majorigual ,
49         op_major ,
50         op_igual ,
51         op_distint ,
52         op_suma ,
53         op_resta ,
54         op_multiplicacio ,
55         op_divisio ,
56         id ,
57         cadena ,
58         const ,
59         Error ,
60         End_of_Input );
61
62
63 END d_token;
```

2.3.2 Fitxer *decls-d\_atribut.ads*

```

1  -- -----
2  --   Paquet de procediments dels atributs
3  -- -----
4  --   Versio   :    0.2
5  --   Autors   :    Jose Ruiz Bravo
6  --               Biel Moya Alcover
7  --               Alvaro Medina Ballester
8  -- -----
9  --       En aquest fitxer tenim implementats les
10 -- assignacions de cada tipus de token al tipus
11 -- atribut que li correspon. Cal destacar
12 -- l'utilització de la taula de noms en els
13 -- casos d'identificadors i strings.
14 --
15 -- -----
16
17 WITH     decls.dgenerals,
18          decls.d_taula_de_noms;
19
20 USE      decls.dgenerals,
21          decls.d_taula_de_noms;
22
23
24 PACKAGE decls.d_atribut IS
25
26
27     TYPE tipus_atribut IS (atom,
28                           a_ident,
29                           a_lit);
30
31     TYPE valor IS NEW integer;
32
33     TYPE atribut (t : tipus_atribut := atom) IS RECORD
34
35         lin, col : natural;
36
37     CASE t IS
38         WHEN atom           => NULL;
39         WHEN a_ident        => idn : id_nom;
40         WHEN a_lit          => val : valor;
41     END CASE;

```

```
42
43     END RECORD;
44
45
46     PROCEDURE mt_atom
47         (l, c : IN natural;
48          a : OUT atribut);
49
50     PROCEDURE mt_identificador
51         (l, c : IN natural;
52          s : IN string;
53          a : OUT atribut);
54
55     PROCEDURE mt_string
56         (l, c : IN natural;
57          s : IN string;
58          a : OUT atribut);
59
60     PROCEDURE mt_caracter
61         (l, c : IN natural;
62          car : IN string;
63          a : OUT atribut);
64
65     PROCEDURE mt_numero
66         (l, c : IN natural;
67          s : IN string;
68          a : OUT atribut);
69
70
71 END decls.d_atribut;
```

**2.3.3 Fitxer *decls-d\_atribut.adb***

```
1  -- -----
2  --   Paquet de procediments dels atributs
3  -- -----
4  --   Versio   :    0.1
5  --   Autors   :    Jose Ruiz Bravo
6  --               Biel Moya Alcover
7  --               Alvaro Medina Ballester
8  -- -----
9  --       En aquest fitxer tenim implementats les
10 -- assignacions de cada tipus de token al tipus
11 -- atribut que li correspon. Cal destacar
12 -- l'utilització de la taula de noms en els
13 -- casos d'identificadors i strings.
14 --
15 -- -----
16
17 WITH     U_Lexica;
18
19 USE      U_Lexica;
20
21
22 PACKAGE BODY decls.d_atribut IS
23
24
25     PROCEDURE mt_atom (l, c : IN natural;
26                        a : OUT atribut) IS
27
28     BEGIN
29         a := (atom, l, c);
30     END mt_atom;
31
32
33     PROCEDURE mt_identificador (l, c : IN natural;
34                                s : IN string;
35                                a : OUT atribut) IS
36
37     BEGIN
38         id := id_nul;
39         posa_id(tn, id, s);
40         a := (a_ident, l, c, id);
41     END mt_identificador;
```

```
42
43     PROCEDURE mt_string (l, c : IN natural;
44                           s : IN string;
45                           a : OUT atribut) IS
46         id : rang_tcar;
47     BEGIN
48         posa_str(tn, id, s);
49         a := (a_lit, l, c, valor(id));
50     END mt_string;
51
52
53     PROCEDURE mt_caracter (l, c : IN natural;
54                             car : IN string;
55                             a : OUT atribut) IS
56     BEGIN
57         a := (a_lit, l, c, valor(car'First+1));
58     END mt_caracter;
59
60
61     PROCEDURE mt_numero (l, c : IN natural;
62                           s : IN string;
63                           a : OUT atribut) IS
64     BEGIN
65         a := (a_lit, l, c, valor(Integer'value(s)));
66     END mt_numero;
67
68
69     END decls.d_atribut;
```



## 3 Declaracions i altres paquets

### 3.1 Fitxer *decls.ads*

```
1  -- -----
2  --   Paquet de Declaracions
3  -- -----
4  --   Versio   :    0.1
5  --   Autors   :    Jose Ruiz Bravo
6  --               Biel Moya Alcover
7  --               Alvaro Medina Ballester
8  -- -----
9  --   Paquet de declaracions pare.
10 --
11 -- -----
12
13 PACKAGE decls IS
14
15     PRAGMA pure;
16
17
18 END decls;
```

### 3.2 Fitxer *decls-dgenerals.ads*

```
1  -- -----
2  --   Paquet de declaracions generals
3  -- -----
4  --   Versio   :    0.3
5  --   Autors   :    Jose Ruiz Bravo
6  --               Biel Moya Alcover
7  --               Alvaro Medina Ballester
8  -- -----
9  --   Declaracions generals.
10 --
11 -- -----
12
13 PACKAGE decls.dgenerals IS
14
15     PRAGMA pure;
16
17     max_id : CONSTANT integer := 1000;
18     long_num_ident : CONSTANT integer := 40;
19
20
21 END decls.dgenerals;
```

## 4 Proves i programa principal

### 4.1 Fitxer *compilemon.adb*, programa principal

```
1  -- -----
2  -- Programa de prova
3  -- -----
4  -- Versio   :   0.1
5  -- Autors   :   Jose Ruiz Bravo
6  --           Biel Moya Alcover
7  --           Alvaro Medina Ballester
8  -- -----
9  -- Programa per comprovar les funcionalitats
10 -- del lexic i la taula de noms.
11 --
12 -- -----
13
14 WITH      Ada.Text_IO ,
15           Ada.Command_Line ,
16           decls.d_taula_de_noms ,
17           decls.tn ,
18           decls.dgenerals ,
19           d_token ,
20           compilemon_io ,
21           u_lexica ;
22
23 USE       Ada.Text_IO ,
24           Ada.Command_Line ,
25           decls.d_taula_de_noms ,
26           decls.tn ,
27           decls.dgenerals ,
28           d_token ,
29           compilemon_io ,
30           u_lexica ;
31
32
33 PROCEDURE compilemon IS
34     Tk:Token;
35 BEGIN
36
37     --tbuida(tn);
38
39     Open_Input(Argument(1));
```

```
40     tk := Ylex;
41
42     WHILE tk /= end_of_input LOOP
43         Put(tok_begin_line'Img);
44         Put_Line(Token'Image(Tk));
45         tk := Ylex;
46     END LOOP;
47
48     close_Input;
49
50     -- exception
51     -- when E_Tids_Plana =>
52     --     Put_Line("ERROR: La taula d'identificadors
53     --             es plena.");
54
55     -- when E_Tcar_Plana =>
56     --     Put_Line("ERROR: La taula de caracters
57     --             es plena.");
58
59     -- when Syntax_Error =>
60     --     Put_Line("ERROR: Error a la linea
61     --             "&yy_line_number'img&" i columna
62     --             "&yy_begin_column'img");
63
64     END compilemon;
```

## 4.2 Proves

### 4.2.1 Prova 1: fitxer *prova01.lem*

```
1 procedimiento
2 PRocedimiento
3 PROCEDURE
4 PROCEDURE
5 BEGIN
6 BEGIN
7 END
8 "provastring"
9 estocastico
10 proves
11 Es
12 ES
13 procedimiento
14 prova
15 prova
16 si
17 sino
```

## Índex

<b>1</b>	<b>Anàlisi Sintàctica</b>	<b>1</b>
1.1	Gramàtica del nostre llenguatge . . . . .	1
<b>2</b>	<b>Anàlisi Lèxica</b>	<b>5</b>
2.1	Descripció del lèxic: <i>compilemon.l</i> . . . . .	5
2.2	Taula de noms . . . . .	11
2.2.1	Fitxer <i>decls-d_taula_de_noms.ads</i> . . . . .	11
2.2.2	Fitxer <i>decls-d_taula_de_noms.adb</i> . . . . .	14
2.3	Tokens i atributs . . . . .	19
2.3.1	Fitxer <i>d_token.ads</i> . . . . .	19
2.3.2	Fitxer <i>decls-d_atribut.ads</i> . . . . .	21
2.3.3	Fitxer <i>decls-d_atribut.adb</i> . . . . .	23
<b>3</b>	<b>Declaracions i altres paquets</b>	<b>25</b>
3.1	Fitxer <i>decls.ads</i> . . . . .	25
3.2	Fitxer <i>decls-dgenerals.ads</i> . . . . .	26
<b>4</b>	<b>Proves i programa principal</b>	<b>27</b>
4.1	Fitxer <i>compilemon.adb</i> , programa principal . . . . .	27
4.2	Proves . . . . .	29
4.2.1	Prova 1: fitxer <i>prova01.lem</i> . . . . .	29