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 composat per un subconjunt bàsic d'instruccions en Ada.

1 Anàlisi Lèxica

1.1 Descripció del lèxic: *compilemon.l*

```
1  -- Macros
2
3  lletra  [A-Za-z]
4
5  digit   [0-9]
6
7  separadors      [\n\b\t\f]
8
9  caracter        \'[^\'\n\t]\'
10
11
12 %%
13
14
15 -- Paraules clau
16
17 procedure       {mt_atom(tok_begin_line, tok_begin_col,
18                         yylval); return pc_procedure;}
19
20 begin           {mt_atom(tok_begin_line, tok_begin_col,
```

```
21                          yylval); return pc_begin;}
22
23 while             {mt_atom(tok_begin_line, tok_begin_col,
24                          yylval); return pc_while;}
25
26 if                          {mt_atom(tok_begin_line, tok_begin_col,
27                          yylval); return pc_if;}
28
29 else              {mt_atom(tok_begin_line, tok_begin_col,
30                          yylval); return pc_else;}
31
32 end                         {mt_atom(tok_begin_line, tok_begin_col,
33                          yylval); return pc_end;}
34
35 do                          {mt_atom(tok_begin_line, tok_begin_col,
36                          yylval); return pc_do;}
37
38 constant          {mt_atom(tok_begin_line, tok_begin_col,
39                          yylval); return pc_constant;}
40
41 type              {mt_atom(tok_begin_line, tok_begin_col,
42                          yylval); return pc_type;}
43
44 array             {mt_atom(tok_begin_line, tok_begin_col,
45                          yylval); return pc_array;}
46
47 record            {mt_atom(tok_begin_line, tok_begin_col,
48                          yylval); return pc_record;}
49
50 is                          {mt_atom(tok_begin_line, tok_begin_col,
51                          yylval); return pc_is;}
52
53 then              {mt_atom(tok_begin_line, tok_begin_col,
54                          yylval); return pc_then;}
55
56 not                         {mt_atom(tok_begin_line, tok_begin_col,
57                          yylval); return pc_not;}
58
59 in                          {mt_atom(tok_begin_line, tok_begin_col,
60                          yylval); return pc_in;}
61
62 out                         {mt_atom(tok_begin_line, tok_begin_col,
63                          yylval); return pc_out;}
```

```
64
65 new                      {mt_atom(tok_begin_line, tok_begin_col,
66                         yylval); return pc_new;}
67
68 null             {mt_atom(tok_begin_line, tok_begin_col,
69                         yylval); return pc_null;}
70
71 of                       {mt_atom(tok_begin_line, tok_begin_col,
72                         yylval); return pc_of;}
73
74 mod                      {mt_atom(tok_begin_line, tok_begin_col,
75                         yylval); return pc_mod;}
76
77 range            {mt_atom(tok_begin_line, tok_begin_col,
78                         yylval); return pc_range;}
79
80 and                      {mt_atom(tok_begin_line, tok_begin_col,
81                         yylval); return pc_or;}
82
83 or                       {mt_atom(tok_begin_line, tok_begin_col,
84                         yylval); return pc_and;}
85
86
87 --Simbols
88
89 ":="             {mt_atom(tok_begin_line, tok_begin_col,
90                         yylval); return s_assignacio;}
91
92 ":"                      {mt_atom(tok_begin_line, tok_begin_col,
93                         yylval); return s_dospunts;}
94
95 ";"                      {mt_atom(tok_begin_line, tok_begin_col,
96                         yylval); return s_final;}
97
98 ","                      {mt_atom(tok_begin_line, tok_begin_col,
99                         yylval); return s_coma;}
100
101 "("                     {mt_atom(tok_begin_line, tok_begin_col,
102                         yylval); return s_parentesiobert;}
103
104 ")"                     {mt_atom(tok_begin_line, tok_begin_col,
105                         yylval); return s_parentesitancat;}
106
```

```
107 ".."            {mt_atom(tok_begin_line, tok_begin_col,
108                         yylval); return s_puntsrang;}
109
110 "."                     {mt_atom(tok_begin_line, tok_begin_col,
111                         yylval); return s_puntrec;}
112
113
114 --Operadors
115
116 "<"                     {mt_atom(tok_begin_line, tok_begin_col,
117                         yylval); return op_menor;}
118
119 "<="            {mt_atom(tok_begin_line, tok_begin_col,
120                         yylval); return op_menorigual;}
121
122 ">="            {mt_atom(tok_begin_line, tok_begin_col,
123                         yylval); return op_majorigual;}
124
125 ">"                     {mt_atom(tok_begin_line, tok_begin_col,
126                         yylval); return op_major;}
127
128 "="                     {mt_atom(tok_begin_line, tok_begin_col,
129                         yylval); return op_igual;}
130
131 "/="            {mt_atom(tok_begin_line, tok_begin_col,
132                         yylval); return op_distint;}
133
134 "+"                     {mt_atom(tok_begin_line, tok_begin_col,
135                         yylval); return op_suma;}
136
137 "-"                     {mt_atom(tok_begin_line, tok_begin_col,
138                         yylval); return op_resta;}
139
140 "*"                     {mt_atom(tok_begin_line, tok_begin_col,
141                         yylval); return op_multiplicacio;}
142
143 "/"                     {mt_atom(tok_begin_line, tok_begin_col,
144                         yylval); return op_divisio;}
145
146
147
148 --EXPRESSIONS REGULARS
149
```

```
150 --Digit
151
152 {digit}+         {mt_numero(tok_begin_line, tok_begin_col,
153                         yytext, yylval); return const;}
154
155
156 --Lletra
157
158 {caracter}       {mt_caracter(tok_begin_line, tok_begin_col,
159                         yytext, yylval); return const;}
160
161
162 --String
163
164 \"[^\"\n\t]*\"   {mt_string(tok_begin_line, tok_begin_col,
165                                 yytext, yylval); return const;}
166
167
168 --Identificador
169
170 {lletra}({digit}|{lletra})* {mt_identificador(tok_begin_line,
171                                 tok_begin_col, yytext, yylval);
172                                 return id;}
173
174
175
176 --Comentaris
177
178 "-""-"[^\n]*     {null;}
179
180
181 --Separadors
182
183 " "                             {null;}
184
185 {separadors}*    {null;}
186
187
188 --Error
189
190 .        {return error;}
191
192
```

```
193
194 %%
195
196
197
198 with     decls.d_taula_de_noms,
199                 d_token,
200                 decls.d_atribut;
201
202
203 use              decls.d_taula_de_noms,
204                 d_token,
205                 decls.d_atribut;
206
207
208 package u_lexica is
209
210         yylval: atribut;
211         tn : taula_de_noms;
212         function YYLex return token;
213         tok_begin_line : integer;
214         tok_begin_col : integer;
215
216 end u_lexica;
217
218
219
220 package body u_lexica is
221
222 ##
223
224 begin
225
226         tbuida(tn);
227
228 end u_lexica;
```

2 Taula de noms

2.1 Fitxer *decls-d_taula_de_noms.ads*

```
1  -- ------------------------------------------------
2  --  Paquet de declaracions de la taula de noms
3  -- ------------------------------------------------
4  --  Versio  :   0.1
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         decls.d_hash;
17
18 use      decls.dgenerals,
19         decls.d_hash;
20
21
22 package decls.d_taula_de_noms is
23
24     --pragma pure;
25
26     -- Excepcions
27     E_Tids_Plena : exception;
28     E_Tcar_Plena : exception;
29
30     type taula_de_noms is limited private;
31
32     procedure tbuida    (tn : out taula_de_noms);
33
34     procedure posa_id   (tn : in out taula_de_noms;
35                          idn : out id_nom;
36                          nom : in string);
37
38     procedure posa_str  (tn : in out taula_de_noms;
39                          ids : out rang_tcar;
```

```
40                              s : in string);
41
42      function cons_nom   (tn : in taula_de_noms;
43                           idn : in id_nom) return string;
44
45      function cons_str   (tn : in taula_de_noms;
46                           ids : in rang_tcar) return string;
47
48
49  private
50
51      type t_identificador is record
52              pos_tcar : rang_tcar;
53               seguent : id_nom;
54          long_paraula : Natural;
55      end record;
56
57      type taula_identificadors is array
58        (1 .. id_nom'Last) of t_identificador;
59
60      type taula_caracters is array (rang_tcar)
61        of character;
62
63      type taula_de_noms is record
64           td : taula_dispersio;
65          tid : taula_identificadors;
66           tc : taula_caracters;
67          nid : id_nom;
68         ncar : rang_tcar;
69      end record;
70
71
72  end decls.d_taula_de_noms;
```

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 --
17 -- ------------------------------------------------
18
19
20 package body decls.d_taula_de_noms is
21
22     -- Donam els valors per defecte de cada camp.
23     procedure tbuida (tn : out taula_de_noms) is
24
25     begin
26
27         for i in tn.td'range loop
28             tn.td(i) := id_nul;
29         end loop;
30
31         tn.nid := 1;
32         Tn.Ncar := 1;
33
34         tn.tid(1).seguent := id_nul;
35
36     end tbuida;
37
38
39
40     procedure posa_id   (tn : in out taula_de_noms;
41                          idn : out id_nom;
```

```
42                          nom : in string) is
43
44          -- Variable per el valor de la funcio de dispersio.
45          p_tid : rang_dispersio;
46
47          -- Index per recorrer la taula d'identificadors.
48          idx : id_nom;
49          Trobat : boolean;
50
51          p : taula_identificadors renames tn.tid;
52
53      begin
54
55          p_tid := fdisp_tn(nom);
56          Idx := Tn.Td(P_Tid);
57          Trobat := False;
58
59          while not Trobat and Idx/=Id_Nul loop
60             if (Nom = Cons_Nom(Tn, Idx)) then
61                Trobat := True;
62             else
63                Idx := p(Idx).Seguent;
64             end if;
65          end loop;
66
67          if not Trobat then
68             Idn := Tn.Nid;
69             p(idn).Pos_Tcar := Tn.Ncar;
70             p(idn).Seguent := Tn.Td(P_Tid);
71             p(idn).Long_Paraula := Nom'Length;
72
73             Tn.Td(P_Tid) := Tn.Nid;
74
75             --Posa a la taula de caracters
76             Tn.Nid := Tn.Nid + 1;
77             for I in 1 .. Nom'Length loop
78                Tn.Tc(Tn.ncar) := Nom(I);
79                Tn.Ncar := Tn.Ncar + 1;
80             end loop;
81
82             Tn.Tc(Tn.Ncar) := '$';
83             Tn.Ncar := Tn.Ncar + 1;
84          end if;
```

```
85
86      end posa_id;
87
88
89
90      procedure posa_str  (tn : in out taula_de_noms;
91                           ids : out rang_tcar;
92                             s : in string) is
93
94          -- Index per recorrer la taula de caracters.
95          jdx : rang_tcar;
96
97      begin
98
99          -- Excepcio per a controlar tc plena
100         if (tn.ncar + s'Length) > rang_tcar'Last then
101             raise E_Tcar_Plena;
102         end if;
103
104         -- Omplim la taula de caracters, desde la primera
105         -- posicio lliure 'ncar'.
106         jdx := tn.ncar;
107         ids := tn.ncar;
108
109         for i in 1..s'Length loop
110             tn.tc(jdx) := s(i);
111             jdx := jdx + 1;
112         end loop;
113
114         tn.ncar := jdx + 1;
115         tn.tc(jdx) := '$';
116
117
118     end posa_str;
119
120
121
122     function cons_nom  (tn : in taula_de_noms;
123                         idn : in id_nom)
124                         return string is
125
126        It1, It2 : Rang_Tcar;
127
```

```
128     begin
129
130        It1 := Tn.Tid(Idn).Pos_Tcar;
131        It2 := Rang_Tcar(Tn.Tid(Idn).Long_Paraula);
132        It2 := It2 + It1 - 1;
133
134        return String(Tn.Tc(it1 .. it2));
135
136
137     end cons_nom;
138
139
140
141     function cons_str   (tn : in taula_de_noms;
142                          ids : in rang_tcar)
143                          return string is
144
145         idx : rang_tcar;
146
147     begin
148
149         idx := ids;
150
151         while (tn.tc(idx) /= '$') loop
152             idx := idx+1;
153         end loop;
154
155         return string(tn.tc(ids..idx-1));
156
157     end cons_str;
158
159
160 end decls.d_taula_de_noms;
```

3 Tokens i atributs

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;
```

3.2 Fitxer *decls-d_ atribut.ads*

```
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'utilitzacio 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_num,
30                            a_lit_car,
31                            a_lit_string);
32
33
34     type atribut (t : tipus_atribut := atom) is record
35
36         lin, col : natural;
37
38         case t is
39
40             when atom           => null;
41
```

```
42              when a_ident         => idn : id_nom;
43
44              when a_lit_num       => int : integer;
45
46              when a_lit_car       => val : character;
47
48              when a_lit_string    => ids : rang_tcar;
49
50          end case;
51
52      end record;
53
54
55      procedure mt_atom       (l, c : in natural;
56                                   a : out atribut);
57
58      procedure mt_identificador  (l, c : in natural;
59                                       s : in string;
60                                       a : out atribut);
61
62      procedure mt_string (l, c : in natural;
63                              s : in string;
64                              a : out atribut);
65
66      procedure mt_caracter   (l, c : in natural;
67                                 car : in string;
68                                   a : out atribut);
69
70      procedure mt_numero (l, c : in natural;
71                              i : in string;
72                              a : out atribut);
73
74
75  end decls.d_atribut;
```

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'utilitzacio 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     begin
28         a := (atom, l, c);
29     end mt_atom;
30
31
32     procedure mt_identificador (l, c : in natural;
33                                 s : in string;
34                                 a : out atribut) is
35         id : id_nom;
36     begin
37         id := id_nul;
38         posa_id(tn, id, s);
39         a := (a_ident, l, c, id);
40     end mt_identificador;
41
```

```
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_string, l, c, 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_car, l, c, car(car'First+1));
58      end mt_caracter;
59
60
61      procedure mt_numero (l, c : in natural;
62                           i : in string;
63                           a : out atribut) is
64      begin
65          a := (a_lit_num, l, c, Integer'value(i));
66      end mt_numero;
67
68
69  end decls.d_atribut;
```

4 Declaracions i altres paquets

4.1 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;
```

4.2 Fitxer *decls-dgenerals.ads*

```
1  -- ------------------------------------------------
2  --  Paquet de declaracions generals
3  -- ------------------------------------------------
4  --  Versio  :   0.2
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    -- TAULA DE NOMS
18    max_id : constant integer := 1000;
19    type id_nom is new integer
20      range 0 .. max_id;
21
22    -- Valor nul per al tipus id_nom
23    id_nul : constant id_nom := 0;
24
25    -- 'Long' es el nombre de paraules * la longitud
26    -- de cadascuna
27    long : constant integer := 40;
28    type rang_tcar is new integer
29      range 0 .. (long*max_id);
30
31    -- Taula de dispersio:
32    -- Tipus per la taula de dispersio de la taula de noms
33    tam_dispersio : constant integer := 101;
34    type rang_dispersio is new integer
35      range -1 .. tam_dispersio;
36
37    -- Valor nul per el rang dispersio
38    dispersio_nul : constant rang_dispersio := -1;
39
40    -- Declaracio de la taula de dispersio
41    type taula_dispersio is array (rang_dispersio)
```

```
42        of id_nom;
43
44
45      -- FORA PER LA PRIMERA ENTREGA
46      -- TAULA DE SIMBOLS
47      type despl is new integer;
48
49      max_prof : constant integer := 20;
50      type nivell_prof is new integer
51        range 0 .. max_prof;
52      nul_nprof : constant nivell_prof := 0;
53
54      max_despl : constant integer := max_prof*max_id;
55      type rang_despl is new integer
56        range 0 .. max_despl;
57      nul_despl : constant rang_despl := 0;
58
59      -- Nombre de variables
60      max_var : constant integer := 1000;
61      type num_var is new integer
62        range 0 .. max_var;
63
64      -- Nombre de procediments
65      max_proc : constant integer := 100;
66      type num_proc is new integer
67        range 0 .. max_proc;
68
69      -- Tipus constant
70      type valor is new integer
71        range 0 .. integer'Last;
72
73
74  end decls.dgenerals;
```

4.3 Fitxer *decls-d_hash.ads*

```
1  -- ------------------------------------------------
2  --  Paquet de declaracions de les funcions hash
3  -- ------------------------------------------------
4  --  Versio  :   0.3
5  --  Autors  :   Jose Ruiz Bravo
6  --              Biel Moya Alcover
7  --              Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --  Especificacio de les funcions de hash:
10 --      - Hash de quadratic per accedir a la
11 --      taula de noms.
12 --
13 -- ------------------------------------------------
14
15 with     decls.dgenerals;
16
17 use      decls.dgenerals;
18
19
20 package decls.d_hash is
21
22     pragma pure;
23
24     function fdisp_tn (nom : in string)
25                        return rang_dispersio;
26
27
28 end decls.d_hash;
```

4.4 Fitxer *decls-d_hash.adb*

```
1  -- ------------------------------------------------
2  --  Paquet de procediments de les funcions hash
3  -- ------------------------------------------------
4  --  Versio  :   0.2
5  --  Autors  :   Jose Ruiz Bravo
6  --              Biel Moya Alcover
7  --              Alvaro Medina Ballester
8  -- ------------------------------------------------
9  --  Procediments de les funcions de hash:
10 --      - Hashing quadratic
11 --
12 -- ------------------------------------------------
13
14 package body decls.d_hash is
15
16    function fdisp_tn (nom : in string)
17                       return rang_dispersio is
18
19          a : array (nom'Range) of integer;
20          r : array (1..2*nom'Last) of integer;
21
22          k, c, m, n : integer;
23
24          base : constant Integer :=
25            Character'Pos(Character'Last)+1;
26
27     begin
28
29          n := nom'Last;
30          m := nom'Length;
31
32          for i in 1..n loop
33              a(i) := character'Pos(nom(i));
34          end loop;
35
36          for i in 1..2*n loop
37              r(i) := 0;
38          end loop;
39
40          for i in 1..n loop
41              c := 0; k := i - 1;
```

```
42              for j in 1..n loop
43                  c := c + r(k+j) + a(i) + a(j);
44                  r(k+j) := c mod base;
45                  c := c/base;
46              end loop;
47              r(k+n+1) := r(k+n+1) + c;
48          end loop;
49
50          c := (r(n+1) * base + r(n)) mod (tam_dispersio);
51
52          return rang_dispersio(c);
53
54      end fdisp_tn;
55
56
57  end decls.d_hash;
```

5 Sintàctica

5.1 Gramàtica del nostre llenguatge

6 Proves i programa principal

6.1 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 := Yylex;
41
42    while tk /= end_of_input loop
43        Put_Line(Token'Image(Tk));
44        tk := Yylex;
45    end loop;
46
47     close_Input;
48
49  --    exception
50    --      when E_Tids_Plena =>
51      --        Put_Line("ERROR: La taula d'identificadors es plena.");
52
53  --        when E_Tcar_Plena =>
54    --          Put_Line("ERROR: La taula de caracters es plena.");
55
56  --        when Syntax_Error =>
57    --          Put_Line("ERROR: Error a la linea "&yy_line_number'img&" i colu
58
59  end compilemon;
```

Índex