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
214 end u_lexica;
215
216
217
218 package body u_lexica is
219
220 ##
221
222 begin
223
224          tbuida(tn);
225
226 end u_lexica;
```

1.2 Taula de noms

1.2.1 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_tc   (tn : IN OUT taula_de_noms;
39                          nom : IN string);
40
```

```
41      PROCEDURE posa_str  (tn : IN OUT taula_de_noms;
42                           ids : OUT rang_tcar;
43                             s : IN string);
44
45      FUNCTION cons_nom   (tn : IN taula_de_noms;
46                           idn : IN id_nom) RETURN string;
47
48      FUNCTION cons_str   (tn : IN taula_de_noms;
49                           ids : IN rang_tcar) RETURN string;
50
51
52  PRIVATE
53
54      TYPE t_identificador IS RECORD
55              pos_tcar : rang_tcar;
56               seguent : id_nom;
57          long_paraula : Natural;
58      END RECORD;
59
60      TYPE taula_identificadors IS ARRAY
61        (1 .. id_nom'Last) OF t_identificador;
62
63      TYPE taula_caracters IS ARRAY (rang_tcar)
64        OF character;
65
66      TYPE taula_de_noms IS RECORD
67           td : taula_dispersio;
68          tid : taula_identificadors;
69           tc : taula_caracters;
70          nid : id_nom;
71         ncar : rang_tcar;
72      END RECORD;
73
74
75  END decls.d_taula_de_noms;
```

1.2.2 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_tc(tn, nom);
76
77              Tn.Tc(Tn.Ncar) := '$';
78              Tn.Ncar := Tn.Ncar + 1;
79          END IF;
80
81      END posa_id;
82
83
84
```

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

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

1.3 Tokens i atributs

1.3.1 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;
```

1.3.2 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'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        => car : character;
47
48              WHEN a_lit_string     => ids : rang_tcar;
49
50          END CASE;
51
52      END RECORD;
53
54
55      PROCEDURE mt_atom
56        (l, c : IN natural;
57             a : OUT atribut);
58
59      PROCEDURE mt_identificador
60        (l, c : IN natural;
61             s : IN string;
62             a : OUT atribut);
63
64      PROCEDURE mt_string
65        (l, c : IN natural;
66             s : IN string;
67             a : OUT atribut);
68
69      PROCEDURE mt_caracter
70        (l, c : IN natural;
71          car : IN string;
72             a : OUT atribut);
73
74      PROCEDURE mt_numero
75        (l, c : IN natural;
76             s : IN string;
77             a : OUT atribut);
78
79
80  END decls.d_atribut;
```

1.3.3 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                           s : IN string;
63                           a : OUT atribut) IS
64      BEGIN
65          a := (a_lit_num, l, c, Integer'value(s));
66      END mt_numero;
67
68
69  END decls.d_atribut;
```

2 Anàlisi Sintàctica

2.1 Gramàtica del nostre llenguatge

programa → *procediment*

procediment → **PC_PROCEDIMENT** *encap* **PC_ES**
declaracions
PC_INICI
bloc
PC_FI identificador;

encap → identificador *args*

args → (*lparam*)
| λ

lparam → *lparam* ; *param*
| *param*

param → identificador : *mode* identificador

mode → **PC_ENTRA**
| **PC_SURT**
| **PC_ENTRA PC_SURT**

declaracions → *declaracions declaracio*
| λ

declaracio → *dec_var*
| *dec_constant*
| *dec_tipus*
| *programa*

– Manual d'usuari variables
dec_var → *lid* : identificador;

lid → *lid*, identificador
| identificador

– Manual d'usuari constant

dec_constant → identificador : **PC_CONSTANT** identificador := *valor*;

valor → *lit*
| − *lit*

lit → lit_num
| lit_car
| lit_string

– Manual d'usuari tipus

dec_tipus → *dec_subrang*
| *dec_registre*
| *dec_coleccio*

dec_subrang → **PC_TIPUS** identificador **PC_ES PC_NOU** identificador
PC_RANG *valor .. valor*;

dec_registre → **PC_TIPUS** identificador **PC_ES PC_REGISTRE**
ldc
PC_FI PC_REGISTRE;

ldc → *ldc dc*
| *dc*

dc → identificador : identificador;

– Tipus colecció (*array*)

dec_coleccio → **PC_TIPUS** identificador **PC_ES PC_COLECCIO**
(*lid*) **PC_DE** identificador;

lid → *lid*, identificador
| identificador

– Bloc d'instruccions

bloc → *bloc sent*
| *sent*

sent → *sassig*
| *scond*
| *srep*

| | *crida_proc*

sassig → *referencia* := *expressio*;

scond → **PC_SI** *expressio* **PC_LLAVORS**
bloc
PC_FI PC_SI;
| **PC_SI** *expressio* **PC_LLAVORS**
bloc
PC_SINO
bloc
PC_FI PC_SI;

srep → **PC_MENTRE** *expressio* **PC_FER**
bloc
PC_FI PC_MENTRE;

crida_proc → *referencia*;

referencia → identificador
| *referencia*.identificador
| *referencia* (*prparam*)

prparam → *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_NO** *expressio*

| (*expressio*)
| *referencia*
| *lit*

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  :   0.2
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_num_ident : CONSTANT integer := 40;
26    TYPE rang_tcar IS NEW integer
27      RANGE 0 .. (long_num_ident*max_id);
28
29    -- Taula de dispersio:
30    tam_dispersio : CONSTANT integer := 101;
31    TYPE rang_dispersio IS NEW integer
32      RANGE -1 .. tam_dispersio;
33
34    -- Valor nul per el rang dispersio
35    dispersio_nul : CONSTANT rang_dispersio := -1;
36
37    TYPE taula_dispersio IS ARRAY (rang_dispersio)
38      OF id_nom;
39
40
41 END decls.dgenerals;
```

3.3 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;
```

3.4 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;
```

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 := Yylex;
41
42    WHILE tk /= end_of_input LOOP
43        Put(tok_begin_line'Img);
44        Put_Line(Token'Image(Tk));
45        tk := Yylex;
46    END LOOP;
47
48     close_Input;
49
50  --    exception
51  --      when E_Tids_Plena =>
52  --      Put_Line("ERROR: La taula d'identificadors
53  --                es plena.");
54
55  --       when E_Tcar_Plena =>
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*

```
procedimiento
PRocedimiento
PROCEDURE
PROCEDURe
BEGIN
BEgIN
END
estocastico
proves
Es
ES
procedimiento
prova
prova
si
sino
```

Índex