```c
/* A Bison parser, made by GNU Bison 2.3.  */

/* Skeleton implementation for Bison's Yacc-like parsers in C

   Copyright (C) 1984, 1989, 1990, 2000, 2001, 2002, 2003, 2004, 2005, 2006
   Free Software Foundation, Inc.

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2, or (at your option)
   any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 51 Franklin Street, Fifth Floor,
   Boston, MA 02110-1301, USA.  */

/* As a special exception, you may create a larger work that contains
   part or all of the Bison parser skeleton and distribute that work
   under terms of your choice, so long as that work isn't itself a
   parser generator using the skeleton or a modified version thereof
   as a parser skeleton.  Alternatively, if you modify or redistribute
   the parser skeleton itself, you may (at your option) remove this
   special exception, which will cause the skeleton and the resulting
   Bison output files to be licensed under the GNU General Public
   License without this special exception.

   This special exception was added by the Free Software Foundation in
   version 2.2 of Bison.  */

/* C LALR(1) parser skeleton written by Richard Stallman, by
   simplifying the original so-called "semantic" parser.  */

/* All symbols defined below should begin with yy or YY, to avoid
   infringing on user name space.  This should be done even for local
   variables, as they might otherwise be expanded by user macros.
   There are some unavoidable exceptions within include files to
   define necessary library symbols; they are noted "INFRINGES ON
   USER NAME SPACE" below.  */

/* Identify Bison output.  */
#define YYBISON 1

/* Bison version.  */
#define YYBISON_VERSION "2.3"

/* Skeleton name.  */
#define YYSKELETON_NAME "yacc.c"

/* Pure parsers.  */
#define YYPURE 0

/* Using locations.  */
#define YYLSP_NEEDED 0



/* Tokens.  */
#ifndef YYTOKENTYPE
# define YYTOKENTYPE
   /* Put the tokens into the symbol table, so that GDB and other debuggers
      know about them.  */
   enum yytokentype {
     OPERADOR_ADITIVO = 258,
     INICIO = 259,
     FIN = 260,
     LEER = 261,
     ESCRIBIR = 262,
```

```
74          ASIGNACION = 263,
75          CONSTANTE = 264,
76          IDENTIFICADOR = 265
77        };
78    #endif
79    /* Tokens.  */
80    #define OPERADOR_ADITIVO 258
81    #define INICIO 259
82    #define FIN 260
83    #define LEER 261
84    #define ESCRIBIR 262
85    #define ASIGNACION 263
86    #define CONSTANTE 264
87    #define IDENTIFICADOR 265
88
89
90
91
92    /* Copy the first part of user declarations.  */
93    #line 1 "yacc_micro.y"
94
95    #include <stdio.h>
96    #include <string.h>
97    #define VARMAXLENGTH 32
98
99    /*
100       El elemento yyin debe declararse como extern pues el mismo esta declarado
          inicialmente en el programa YACC y de lo contrario
101       obtendríamos un error porque estaríamos redefiniendo el elemento.
102   */
103   extern FILE *yyin;
104   void yyerror(const char *str);
105
106   int stringLength(char* str);
107   int identificadorValido(char* id);
108   int yywrap();
109   #line 18 "yacc_micro.y"
110
111   /*
112       Lo siguiente son las declaraciones de TOKEN para YACC. Las mismas son
          convertidas a INT para poder ser retornadas en LEX.
113   */
114
115
116   /* Enabling traces.  */
117   #ifndef YYDEBUG
118   # define YYDEBUG 0
119   #endif
120
121   /* Enabling verbose error messages.  */
122   #ifdef YYERROR_VERBOSE
123   # undef YYERROR_VERBOSE
124   # define YYERROR_VERBOSE 1
125   #else
126   # define YYERROR_VERBOSE 0
127   #endif
128
129   /* Enabling the token table.  */
130   #ifndef YYTOKEN_TABLE
131   # define YYTOKEN_TABLE 0
132   #endif
133
134   #if ! defined YYSTYPE && ! defined YYSTYPE_IS_DECLARED
135   typedef union YYSTYPE
136   #line 25 "yacc_micro.y"
137   {char* identificador;}
138   /* Line 193 of yacc.c.  */
139   #line 140 "y.tab.c"
140       YYSTYPE;
141   # define yystype YYSTYPE /* obsolescent; will be withdrawn */
142   # define YYSTYPE_IS_DECLARED 1
143   # define YYSTYPE_IS_TRIVIAL 1
144   #endif
```

```
145
146
147
148 /* Copy the second part of user declarations.  */
149
150
151 /* Line 216 of yacc.c.  */
152 #line 153 "y.tab.c"
153
154 #ifdef short
155 # undef short
156 #endif
157
158 #ifdef YYTYPE_UINT8
159 typedef YYTYPE_UINT8 yytype_uint8;
160 #else
161 typedef unsigned char yytype_uint8;
162 #endif
163
164 #ifdef YYTYPE_INT8
165 typedef YYTYPE_INT8 yytype_int8;
166 #elif (defined __STDC__ || defined __C99__FUNC__ \
167      || defined __cplusplus || defined _MSC_VER)
168 typedef signed char yytype_int8;
169 #else
170 typedef short int yytype_int8;
171 #endif
172
173 #ifdef YYTYPE_UINT16
174 typedef YYTYPE_UINT16 yytype_uint16;
175 #else
176 typedef unsigned short int yytype_uint16;
177 #endif
178
179 #ifdef YYTYPE_INT16
180 typedef YYTYPE_INT16 yytype_int16;
181 #else
182 typedef short int yytype_int16;
183 #endif
184
185 #ifndef YYSIZE_T
186 # ifdef __SIZE_TYPE__
187 #  define YYSIZE_T __SIZE_TYPE__
188 # elif defined size_t
189 #  define YYSIZE_T size_t
190 # elif ! defined YYSIZE_T && (defined __STDC__ || defined __C99__FUNC__ \
191      || defined __cplusplus || defined _MSC_VER)
192 #  include <stddef.h> /* INFRINGES ON USER NAME SPACE */
193 #  define YYSIZE_T size_t
194 # else
195 #  define YYSIZE_T unsigned int
196 # endif
197 #endif
198
199 #define YYSIZE_MAXIMUM ((YYSIZE_T) -1)
200
201 #ifndef YY_
202 # if defined YYENABLE_NLS && YYENABLE_NLS
203 #  if ENABLE_NLS
204 #   include <libintl.h> /* INFRINGES ON USER NAME SPACE */
205 #   define YY_(msgid) dgettext ("bison-runtime", msgid)
206 #  endif
207 # endif
208 # ifndef YY_
209 #  define YY_(msgid) msgid
210 # endif
211 #endif
212
213 /* Suppress unused-variable warnings by "using" E.  */
214 #if ! defined lint || defined __GNUC__
215 # define YYUSE(e) ((void) (e))
216 #else
217 # define YYUSE(e) /* empty */
```

```
218  #endif
219
220  /* Identity function, used to suppress warnings about constant conditions.  */
221  #ifndef lint
222  # define YYID(n) (n)
223  #else
224  #if (defined __STDC__ || defined __C99__FUNC__ \
225       || defined __cplusplus || defined _MSC_VER)
226  static int
227  YYID (int i)
228  #else
229  static int
230  YYID (i)
231      int i;
232  #endif
233  {
234    return i;
235  }
236  #endif
237
238  #if ! defined yyoverflow || YYERROR_VERBOSE
239
240  /* The parser invokes alloca or malloc; define the necessary symbols.  */
241
242  # ifdef YYSTACK_USE_ALLOCA
243  #  if YYSTACK_USE_ALLOCA
244  #   ifdef __GNUC__
245  #    define YYSTACK_ALLOC __builtin_alloca
246  #   elif defined __BUILTIN_VA_ARG_INCR
247  #    include <alloca.h> /* INFRINGES ON USER NAME SPACE */
248  #   elif defined _AIX
249  #    define YYSTACK_ALLOC __alloca
250  #   elif defined _MSC_VER
251  #    include <malloc.h> /* INFRINGES ON USER NAME SPACE */
252  #    define alloca _alloca
253  #   else
254  #    define YYSTACK_ALLOC alloca
255  #    if ! defined _ALLOCA_H && ! defined _STDLIB_H && (defined __STDC__ || defined
     __C99__FUNC__ \
256       || defined __cplusplus || defined _MSC_VER)
257  #     include <stdlib.h> /* INFRINGES ON USER NAME SPACE */
258  #     ifndef _STDLIB_H
259  #      define _STDLIB_H 1
260  #     endif
261  #    endif
262  #   endif
263  #  endif
264  # endif
265
266  # ifdef YYSTACK_ALLOC
267     /* Pacify GCC's `empty if-body' warning.  */
268  #  define YYSTACK_FREE(Ptr) do { /* empty */; } while (YYID (0))
269  #  ifndef YYSTACK_ALLOC_MAXIMUM
270      /* The OS might guarantee only one guard page at the bottom of the stack,
271         and a page size can be as small as 4096 bytes.  So we cannot safely
272         invoke alloca (N) if N exceeds 4096.  Use a slightly smaller number
273         to allow for a few compiler-allocated temporary stack slots.  */
274  #   define YYSTACK_ALLOC_MAXIMUM 4032 /* reasonable circa 2006 */
275  #  endif
276  # else
277  #  define YYSTACK_ALLOC YYMALLOC
278  #  define YYSTACK_FREE YYFREE
279  #  ifndef YYSTACK_ALLOC_MAXIMUM
280  #   define YYSTACK_ALLOC_MAXIMUM YYSIZE_MAXIMUM
281  #  endif
282  #  if (defined __cplusplus && ! defined _STDLIB_H \
283         && ! ((defined YYMALLOC || defined malloc) \
284           && (defined YYFREE || defined free)))
285  #   include <stdlib.h> /* INFRINGES ON USER NAME SPACE */
286  #   ifndef _STDLIB_H
287  #    define _STDLIB_H 1
288  #   endif
289  #  endif
```

```
#  ifndef YYMALLOC
#   define YYMALLOC malloc
#   if ! defined malloc && ! defined _STDLIB_H && (defined __STDC__ || defined __C99__FUNC__ \
     || defined __cplusplus || defined _MSC_VER)
void *malloc (YYSIZE_T); /* INFRINGES ON USER NAME SPACE */
#   endif
#  endif
#  ifndef YYFREE
#   define YYFREE free
#   if ! defined free && ! defined _STDLIB_H && (defined __STDC__ || defined __C99__FUNC__ \
     || defined __cplusplus || defined _MSC_VER)
void free (void *); /* INFRINGES ON USER NAME SPACE */
#   endif
#  endif
# endif
#endif /* ! defined yyoverflow || YYERROR_VERBOSE */


#if (! defined yyoverflow \
     && (! defined __cplusplus \
     || (defined YYSTYPE_IS_TRIVIAL && YYSTYPE_IS_TRIVIAL)))

/* A type that is properly aligned for any stack member.  */
union yyalloc
{
  yytype_int16 yyss;
  YYSTYPE yyvs;
  };

/* The size of the maximum gap between one aligned stack and the next.  */
# define YYSTACK_GAP_MAXIMUM (sizeof (union yyalloc) - 1)

/* The size of an array large to enough to hold all stacks, each with
   N elements.  */
# define YYSTACK_BYTES(N) \
     ((N) * (sizeof (yytype_int16) + sizeof (YYSTYPE)) \
      + YYSTACK_GAP_MAXIMUM)

/* Copy COUNT objects from FROM to TO.  The source and destination do
   not overlap.  */
# ifndef YYCOPY
#  if defined __GNUC__ && 1 < __GNUC__
#   define YYCOPY(To, From, Count) \
      __builtin_memcpy (To, From, (Count) * sizeof (*(From)))
#  else
#   define YYCOPY(To, From, Count)      \
      do                     \
    {                    \
      YYSIZE_T yyi;              \
      for (yyi = 0; yyi < (Count); yyi++)    \
        (To)[yyi] = (From)[yyi];         \
    }                    \
      while (YYID (0))
#  endif
# endif

/* Relocate STACK from its old location to the new one.  The
   local variables YYSIZE and YYSTACKSIZE give the old and new number of
   elements in the stack, and YYPTR gives the new location of the
   stack.  Advance YYPTR to a properly aligned location for the next
   stack.  */
# define YYSTACK_RELOCATE(Stack)                     \
    do                                   \
      {                                  \
    YYSIZE_T yynewbytes;                         \
    YYCOPY (&yyptr->Stack, Stack, yysize);               \
    Stack = &yyptr->Stack;                       \
    yynewbytes = yystacksize * sizeof (*Stack) + YYSTACK_GAP_MAXIMUM; \
    yyptr += yynewbytes / sizeof (*yyptr);               \
      }                                  \
    while (YYID (0))
```

```
361
362   #endif
363
364   /* YYFINAL -- State number of the termination state.  */
365   #define YYFINAL  8
366   /* YYLAST -- Last index in YYTABLE.  */
367   #define YYLAST   27
368
369   /* YYNTOKENS -- Number of terminals.  */
370   #define YYNTOKENS  15
371   /* YYNNTS -- Number of nonterminals.  */
372   #define YYNNTS  8
373   /* YYNRULES -- Number of rules.  */
374   #define YYNRULES  16
375   /* YYNRULES -- Number of states.  */
376   #define YYNSTATES  36
377
378   /* YYTRANSLATE(YYLEX) -- Bison symbol number corresponding to YYLEX.  */
379   #define YYUNDEFTOK  2
380   #define YYMAXUTOK   265
381
382   #define YYTRANSLATE(YYX)                                \
383     ((unsigned int) (YYX) <= YYMAXUTOK ? yytranslate[YYX] : YYUNDEFTOK)
384
385   /* YYTRANSLATE[YYLEX] -- Bison symbol number corresponding to YYLEX.  */
386   static const yytype_uint8 yytranslate[] =
387   {
388          0,     2,     2,     2,     2,     2,     2,     2,     2,     2,
389          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
390          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
391          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
392         12,    13,     2,     2,    14,     2,     2,     2,     2,     2,
393          2,     2,     2,     2,     2,     2,     2,     2,     2,    11,
394          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
395          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
396          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
397          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
398          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
399          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
400          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
401          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
402          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
403          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
404          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
405          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
406          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
407          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
408          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
409          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
410          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
411          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
412          2,     2,     2,     2,     2,     2,     2,     2,     2,     2,
413          2,     2,     2,     2,     2,     2,     1,     2,     3,     4,
414          5,     6,     7,     8,     9,    10
415   };
416
417   #if YYDEBUG
418   /* YYPRHS[YYN] -- Index of the first RHS symbol of rule number YYN in
419      YYRHS.  */
420   static const yytype_uint8 yyprhs[] =
421   {
422          0,     0,     3,     7,    10,    12,    17,    23,    29,    31,
423         35,    37,    41,    43,    47,    49,    51
424   };
425
426   /* YYRHS -- A `-1'-separated list of the rules' RHS.  */
427   static const yytype_int8 yyrhs[] =
428   {
429         16,     0,    -1,     4,    17,     5,    -1,    18,    17,    -1,
430         18,    -1,    10,     8,    21,    11,    -1,     6,    12,    19,
431         13,    11,    -1,     7,    12,    20,    13,    11,    -1,    10,
432         -1,    10,    14,    19,    -1,    21,    -1,    21,    14,    20,
433         -1,    22,    -1,    22,     3,    21,    -1,    10,    -1,     9,
```

```
       -1,    12,    21,    13,    -1
};

/* YYRLINE[YYN] -- source line where rule number YYN was defined.  */
static const yytype_uint8 yyrline[] =
{
       0,    33,    33,    37,    38,    40,    47,    48,    53,    54,
      59,    60,    62,    63,    65,    71,    72
};
#endif

#if YYDEBUG || YYERROR_VERBOSE || YYTOKEN_TABLE
/* YYTNAME[SYMBOL-NUM] -- String name of the symbol SYMBOL-NUM.
   First, the terminals, then, starting at YYNTOKENS, nonterminals.  */
static const char *const yytname[] =
{
  "$end", "error", "$undefined", "OPERADOR_ADITIVO", "INICIO", "FIN",
  "LEER", "ESCRIBIR", "ASIGNACION", "CONSTANTE", "IDENTIFICADOR", "';'",
  "'('", "')'", "','", "$accept", "programa", "listaSentencias",
  "sentencia", "listaIdentificadores", "listaExpresiones", "expresion",
  "primaria", 0
};
#endif

# ifdef YYPRINT
/* YYTOKNUM[YYLEX-NUM] -- Internal token number corresponding to
   token YYLEX-NUM.  */
static const yytype_uint16 yytoknum[] =
{
       0,   256,   257,   258,   259,   260,   261,   262,   263,   264,
     265,    59,    40,    41,    44
};
# endif

/* YYR1[YYN] -- Symbol number of symbol that rule YYN derives.  */
static const yytype_uint8 yyr1[] =
{
       0,    15,    16,    17,    17,    18,    18,    18,    19,    19,
      20,    20,    21,    21,    22,    22,    22
};

/* YYR2[YYN] -- Number of symbols composing right hand side of rule YYN.  */
static const yytype_uint8 yyr2[] =
{
       0,     2,     3,     2,     1,     4,     5,     5,     1,     3,
       1,     3,     1,     3,     1,     1,     3
};

/* YYDEFACT[STATE-NAME] -- Default rule to reduce with in state
   STATE-NUM when YYTABLE doesn't specify something else to do.  Zero
   means the default is an error.  */
static const yytype_uint8 yydefact[] =
{
       0,     0,     0,     0,     0,     0,     0,     4,     1,     0,
       0,     0,     2,     3,     8,     0,    15,    14,     0,     0,
      10,    12,     0,     0,     0,     0,     0,     0,     0,     5,
       9,     6,    16,     7,    11,    13
};

/* YYDEFGOTO[NTERM-NUM].  */
static const yytype_int8 yydefgoto[] =
{
      -1,     2,     6,     7,    15,    19,    20,    21
};

/* YYPACT[STATE-NUM] -- Index in YYTABLE of the portion describing
   STATE-NUM.  */
#define YYPACT_NINF -12
static const yytype_int8 yypact[] =
{
       4,    -5,     9,    -2,    -1,     5,     7,    -5,   -12,     6,
      -6,    -6,   -12,   -12,     0,     2,   -12,   -12,    -6,     8,
      10,    15,    11,     6,    12,    13,    14,    -6,    -6,   -12,
```

```
507       -12,   -12,   -12,   -12,   -12,   -12
508   };
509
510   /* YYPGOTO[NTERM-NUM].  */
511   static const yytype_int8 yypgoto[] =
512   {
513       -12,   -12,    20,   -12,    -4,    -7,   -11,   -12
514   };
515
516   /* YYTABLE[YYPACT[STATE-NUM]].  What to do in state STATE-NUM.  If
517      positive, shift that token.  If negative, reduce the rule which
518      number is the opposite.  If zero, do what YYDEFACT says.
519      If YYTABLE_NINF, syntax error.  */
520   #define YYTABLE_NINF -1
521   static const yytype_uint8 yytable[] =
522   {
523         22,     3,     4,    16,    17,     5,    18,    25,     1,     8,
524          9,    10,    12,    11,    23,    24,    14,    35,    28,    30,
525         34,    26,    29,    31,    27,    33,    32,    13
526   };
527
528   static const yytype_uint8 yycheck[] =
529   {
530         11,     6,     7,     9,    10,    10,    12,    18,     4,     0,
531         12,    12,     5,     8,    14,    13,    10,    28,     3,    23,
532         27,    13,    11,    11,    14,    11,    13,     7
533   };
534
535   /* YYSTOS[STATE-NUM] -- The (internal number of the) accessing
536      symbol of state STATE-NUM.  */
537   static const yytype_uint8 yystos[] =
538   {
539          0,     4,    16,     6,     7,    10,    17,    18,     0,    12,
540         12,     8,     5,    17,    10,    19,     9,    10,    12,    20,
541         21,    22,    21,    14,    13,    21,    13,    14,     3,    11,
542         19,    11,    13,    11,    20,    21
543   };
544
545   #define yyerrok     (yyerrstatus = 0)
546   #define yyclearin   (yychar = YYEMPTY)
547   #define YYEMPTY     (-2)
548   #define YYEOF       0
549
550   #define YYACCEPT    goto yyacceptlab
551   #define YYABORT     goto yyabortlab
552   #define YYERROR     goto yyerrorlab
553
554
555   /* Like YYERROR except do call yyerror.  This remains here temporarily
556      to ease the transition to the new meaning of YYERROR, for GCC.
557      Once GCC version 2 has supplanted version 1, this can go.  */
558
559   #define YYFAIL      goto yyerrlab
560
561   #define YYRECOVERING()  (!!yyerrstatus)
562
563   #define YYBACKUP(Token, Value)                  \
564   do                                \
565     if (yychar == YYEMPTY && yylen == 1)              \
566       {                                \
567         yychar = (Token);                       \
568         yylval = (Value);                       \
569         yytoken = YYTRANSLATE (yychar);               \
570         YYPOPSTACK (1);                         \
571         goto yybackup;                          \
572       }                                \
573     else                                \
574       {                                \
575         yyerror (YY_("syntax error: cannot back up")); \
576         YYERROR;                              \
577       }                                \
578   while (YYID (0))
579
```

```
580
581   #define YYTERROR    1
582   #define YYERRCODE   256
583
584
585   /* YYLLOC_DEFAULT -- Set CURRENT to span from RHS[1] to RHS[N].
586      If N is 0, then set CURRENT to the empty location which ends
587      the previous symbol: RHS[0] (always defined).  */
588
589   #define YYRHSLOC(Rhs, K) ((Rhs)[K])
590   #ifndef YYLLOC_DEFAULT
591   # define YYLLOC_DEFAULT(Current, Rhs, N)                \
592       do                                  \
593         if (YYID (N))                                                    \
594       {                                 \
595         (Current).first_line   = YYRHSLOC (Rhs, 1).first_line;     \
596         (Current).first_column = YYRHSLOC (Rhs, 1).first_column;   \
597         (Current).last_line    = YYRHSLOC (Rhs, N).last_line;      \
598         (Current).last_column  = YYRHSLOC (Rhs, N).last_column;    \
599       }                                 \
600         else                                \
601       {                                 \
602         (Current).first_line   = (Current).last_line   =       \
603           YYRHSLOC (Rhs, 0).last_line;                 \
604         (Current).first_column = (Current).last_column =       \
605           YYRHSLOC (Rhs, 0).last_column;               \
606       }                                 \
607       while (YYID (0))
608   #endif
609
610
611   /* YY_LOCATION_PRINT -- Print the location on the stream.
612      This macro was not mandated originally: define only if we know
613      we won't break user code: when these are the locations we know.  */
614
615   #ifndef YY_LOCATION_PRINT
616   # if defined YYLTYPE_IS_TRIVIAL && YYLTYPE_IS_TRIVIAL
617   #  define YY_LOCATION_PRINT(File, Loc)          \
618       fprintf (File, "%d.%d-%d.%d",           \
619             (Loc).first_line, (Loc).first_column, \
620             (Loc).last_line,  (Loc).last_column)
621   # else
622   #  define YY_LOCATION_PRINT(File, Loc) ((void) 0)
623   # endif
624   #endif
625
626
627   /* YYLEX -- calling `yylex' with the right arguments.  */
628
629   #ifdef YYLEX_PARAM
630   # define YYLEX yylex (YYLEX_PARAM)
631   #else
632   # define YYLEX yylex ()
633   #endif
634
635   /* Enable debugging if requested.  */
636   #if YYDEBUG
637
638   # ifndef YYFPRINTF
639   #  include <stdio.h> /* INFRINGES ON USER NAME SPACE */
640   #  define YYFPRINTF fprintf
641   # endif
642
643   # define YYDPRINTF(Args)            \
644   do {                        \
645     if (yydebug)                    \
646       YYFPRINTF Args;               \
647   } while (YYID (0))
648
649   # define YY_SYMBOL_PRINT(Title, Type, Value, Location)            \
650   do {                                      \
651     if (yydebug)                                \
652       {                                     \
```

```
653         YYFPRINTF (stderr, "%s ", Title);                   \
654         yy_symbol_print (stderr,                            \
655             Type, Value); \
656         YYFPRINTF (stderr, "\n");                           \
657       }                                     \
658   } while (YYID (0))
659
660
661   /*--------------------------------.
662   | Print this symbol on YYOUTPUT.  |
663   `--------------------------------*/
664
665   /*ARGSUSED*/
666   #if (defined __STDC__ || defined __C99__FUNC__ \
667        || defined __cplusplus || defined _MSC_VER)
668   static void
669   yy_symbol_value_print (FILE *yyoutput, int yytype, YYSTYPE const * const yyvaluep)
670   #else
671   static void
672   yy_symbol_value_print (yyoutput, yytype, yyvaluep)
673       FILE *yyoutput;
674       int yytype;
675       YYSTYPE const * const yyvaluep;
676   #endif
677   {
678     if (!yyvaluep)
679       return;
680   # ifdef YYPRINT
681     if (yytype < YYNTOKENS)
682       YYPRINT (yyoutput, yytoknum[yytype], *yyvaluep);
683   # else
684     YYUSE (yyoutput);
685   # endif
686     switch (yytype)
687       {
688         default:
689       break;
690       }
691   }
692
693
694   /*--------------------------------.
695   | Print this symbol on YYOUTPUT.  |
696   `--------------------------------*/
697
698   #if (defined __STDC__ || defined __C99__FUNC__ \
699        || defined __cplusplus || defined _MSC_VER)
700   static void
701   yy_symbol_print (FILE *yyoutput, int yytype, YYSTYPE const * const yyvaluep)
702   #else
703   static void
704   yy_symbol_print (yyoutput, yytype, yyvaluep)
705       FILE *yyoutput;
706       int yytype;
707       YYSTYPE const * const yyvaluep;
708   #endif
709   {
710     if (yytype < YYNTOKENS)
711       YYFPRINTF (yyoutput, "token %s (", yytname[yytype]);
712     else
713       YYFPRINTF (yyoutput, "nterm %s (", yytname[yytype]);
714
715     yy_symbol_value_print (yyoutput, yytype, yyvaluep);
716     YYFPRINTF (yyoutput, ")");
717   }
718
719   /*------------------------------------------------------------------.
720   | yy_stack_print -- Print the state stack from its BOTTOM up to its |
721   | TOP (included).                                                   |
722   `------------------------------------------------------------------*/
723
724   #if (defined __STDC__ || defined __C99__FUNC__ \
725        || defined __cplusplus || defined _MSC_VER)
```

```
726  static void
727  yy_stack_print (yytype_int16 *bottom, yytype_int16 *top)
728  #else
729  static void
730  yy_stack_print (bottom, top)
731      yytype_int16 *bottom;
732      yytype_int16 *top;
733  #endif
734  {
735    YYFPRINTF (stderr, "Stack now");
736    for (; bottom <= top; ++bottom)
737      YYFPRINTF (stderr, " %d", *bottom);
738    YYFPRINTF (stderr, "\n");
739  }
740
741  # define YY_STACK_PRINT(Bottom, Top)                \
742  do {                                  \
743    if (yydebug)                                \
744      yy_stack_print ((Bottom), (Top));                \
745  } while (YYID (0))
746
747
748  /*------------------------------------------------.
749  | Report that the YYRULE is going to be reduced.  |
750  `------------------------------------------------*/
751
752  #if (defined __STDC__ || defined __C99__FUNC__ \
753       || defined __cplusplus || defined _MSC_VER)
754  static void
755  yy_reduce_print (YYSTYPE *yyvsp, int yyrule)
756  #else
757  static void
758  yy_reduce_print (yyvsp, yyrule)
759      YYSTYPE *yyvsp;
760      int yyrule;
761  #endif
762  {
763    int yynrhs = yyr2[yyrule];
764    int yyi;
765    unsigned long int yylno = yyrline[yyrule];
766    YYFPRINTF (stderr, "Reducing stack by rule %d (line %lu):\n",
767               yyrule - 1, yylno);
768    /* The symbols being reduced.  */
769    for (yyi = 0; yyi < yynrhs; yyi++)
770      {
771        fprintf (stderr, "   $%d = ", yyi + 1);
772        yy_symbol_print (stderr, yyrhs[yyprhs[yyrule] + yyi],
773                         &(yyvsp[(yyi + 1) - (yynrhs)])
774                                         );
775        fprintf (stderr, "\n");
776      }
777  }
778
779  # define YY_REDUCE_PRINT(Rule)          \
780  do {                                  \
781    if (yydebug)                                \
782      yy_reduce_print (yyvsp, Rule); \
783  } while (YYID (0))
784
785  /* Nonzero means print parse trace.  It is left uninitialized so that
786     multiple parsers can coexist.  */
787  int yydebug;
788  #else /* !YYDEBUG */
789  # define YYDPRINTF(Args)
790  # define YY_SYMBOL_PRINT(Title, Type, Value, Location)
791  # define YY_STACK_PRINT(Bottom, Top)
792  # define YY_REDUCE_PRINT(Rule)
793  #endif /* !YYDEBUG */
794
795
796  /* YYINITDEPTH -- initial size of the parser's stacks.  */
797  #ifndef YYINITDEPTH
798  # define YYINITDEPTH 200
```

```
#endif

/* YYMAXDEPTH -- maximum size the stacks can grow to (effective only
   if the built-in stack extension method is used).

   Do not make this value too large; the results are undefined if
   YYSTACK_ALLOC_MAXIMUM < YYSTACK_BYTES (YYMAXDEPTH)
   evaluated with infinite-precision integer arithmetic.  */

#ifndef YYMAXDEPTH
# define YYMAXDEPTH 10000
#endif

FF

#if YYERROR_VERBOSE

# ifndef yystrlen
#  if defined __GLIBC__ && defined _STRING_H
#   define yystrlen strlen
#  else
/* Return the length of YYSTR.  */
#if (defined __STDC__ || defined __C99__FUNC__ \
     || defined __cplusplus || defined _MSC_VER)
static YYSIZE_T
yystrlen (const char *yystr)
#else
static YYSIZE_T
yystrlen (yystr)
    const char *yystr;
#endif
{
  YYSIZE_T yylen;
  for (yylen = 0; yystr[yylen]; yylen++)
    continue;
  return yylen;
}
#  endif
# endif

# ifndef yystpcpy
#  if defined __GLIBC__ && defined _STRING_H && defined _GNU_SOURCE
#   define yystpcpy stpcpy
#  else
/* Copy YYSRC to YYDEST, returning the address of the terminating '\0' in
   YYDEST.  */
#if (defined __STDC__ || defined __C99__FUNC__ \
     || defined __cplusplus || defined _MSC_VER)
static char *
yystpcpy (char *yydest, const char *yysrc)
#else
static char *
yystpcpy (yydest, yysrc)
    char *yydest;
    const char *yysrc;
#endif
{
  char *yyd = yydest;
  const char *yys = yysrc;

  while ((*yyd++ = *yys++) != '\0')
    continue;

  return yyd - 1;
}
#  endif
# endif

# ifndef yytnamerr
/* Copy to YYRES the contents of YYSTR after stripping away unnecessary
   quotes and backslashes, so that it's suitable for yyerror.  The
   heuristic is that double-quoting is unnecessary unless the string
   contains an apostrophe, a comma, or backslash (other than
```

```
872       backslash-backslash).  YYSTR is taken from yytname.  If YYRES is
873       null, do not copy; instead, return the length of what the result
874       would have been.  */
875    static YYSIZE_T
876    yytnamerr (char *yyres, const char *yystr)
877    {
878      if (*yystr == '"')
879        {
880          YYSIZE_T yyn = 0;
881          char const *yyp = yystr;
882
883          for (;;)
884        switch (*++yyp)
885          {
886          case '\'':
887          case ',':
888            goto do_not_strip_quotes;
889
890          case '\\':
891            if (*++yyp != '\\')
892              goto do_not_strip_quotes;
893            /* Fall through.  */
894          default:
895            if (yyres)
896              yyres[yyn] = *yyp;
897            yyn++;
898            break;
899
900          case '"':
901            if (yyres)
902              yyres[yyn] = '\0';
903            return yyn;
904          }
905        do_not_strip_quotes: ;
906        }
907
908      if (! yyres)
909        return yystrlen (yystr);
910
911      return yystpcpy (yyres, yystr) - yyres;
912    }
913    # endif
914
915    /* Copy into YYRESULT an error message about the unexpected token
916       YYCHAR while in state YYSTATE.  Return the number of bytes copied,
917       including the terminating null byte.  If YYRESULT is null, do not
918       copy anything; just return the number of bytes that would be
919       copied.  As a special case, return 0 if an ordinary "syntax error"
920       message will do.  Return YYSIZE_MAXIMUM if overflow occurs during
921       size calculation.  */
922    static YYSIZE_T
923    yysyntax_error (char *yyresult, int yystate, int yychar)
924    {
925      int yyn = yypact[yystate];
926
927      if (! (YYPACT_NINF < yyn && yyn <= YYLAST))
928        return 0;
929      else
930        {
931          int yytype = YYTRANSLATE (yychar);
932          YYSIZE_T yysize0 = yytnamerr (0, yytname[yytype]);
933          YYSIZE_T yysize = yysize0;
934          YYSIZE_T yysize1;
935          int yysize_overflow = 0;
936          enum { YYERROR_VERBOSE_ARGS_MAXIMUM = 5 };
937          char const *yyarg[YYERROR_VERBOSE_ARGS_MAXIMUM];
938          int yyx;
939
940    # if 0
941          /* This is so xgettext sees the translatable formats that are
942         constructed on the fly.  */
943          YY_("syntax error, unexpected %s");
944          YY_("syntax error, unexpected %s, expecting %s");
```

```
945         YY_("syntax error, unexpected %s, expecting %s or %s");
946         YY_("syntax error, unexpected %s, expecting %s or %s or %s");
947         YY_("syntax error, unexpected %s, expecting %s or %s or %s or %s");
948   # endif
949         char *yyfmt;
950         char const *yyf;
951         static char const yyunexpected[] = "syntax error, unexpected %s";
952         static char const yyexpecting[] = ", expecting %s";
953         static char const yyor[] = " or %s";
954         char yyformat[sizeof yyunexpected
955                       + sizeof yyexpecting - 1
956                       + ((YYERROR_VERBOSE_ARGS_MAXIMUM - 2)
957                          * (sizeof yyor - 1))];
958         char const *yyprefix = yyexpecting;
959
960         /* Start YYX at -YYN if negative to avoid negative indexes in
961        YYCHECK.  */
962         int yyxbegin = yyn < 0 ? -yyn : 0;
963
964         /* Stay within bounds of both yycheck and yytname.  */
965         int yychecklim = YYLAST - yyn + 1;
966         int yyxend = yychecklim < YYNTOKENS ? yychecklim : YYNTOKENS;
967         int yycount = 1;
968
969         yyarg[0] = yytname[yytype];
970         yyfmt = yystpcpy (yyformat, yyunexpected);
971
972         for (yyx = yyxbegin; yyx < yyxend; ++yyx)
973       if (yycheck[yyx + yyn] == yyx && yyx != YYTERROR)
974         {
975           if (yycount == YYERROR_VERBOSE_ARGS_MAXIMUM)
976             {
977           yycount = 1;
978           yysize = yysize0;
979           yyformat[sizeof yyunexpected - 1] = '\0';
980           break;
981             }
982           yyarg[yycount++] = yytname[yyx];
983           yysize1 = yysize + yytnamerr (0, yytname[yyx]);
984           yysize_overflow |= (yysize1 < yysize);
985           yysize = yysize1;
986           yyfmt = yystpcpy (yyfmt, yyprefix);
987           yyprefix = yyor;
988         }
989
990         yyf = YY_(yyformat);
991         yysize1 = yysize + yystrlen (yyf);
992         yysize_overflow |= (yysize1 < yysize);
993         yysize = yysize1;
994
995         if (yysize_overflow)
996       return YYSIZE_MAXIMUM;
997
998         if (yyresult)
999       {
1000         /* Avoid sprintf, as that infringes on the user's name space.
1001            Don't have undefined behavior even if the translation
1002            produced a string with the wrong number of "%s"s.  */
1003         char *yyp = yyresult;
1004         int yyi = 0;
1005         while ((*yyp = *yyf) != '\0')
1006           {
1007             if (*yyp == '%' && yyf[1] == 's' && yyi < yycount)
1008           {
1009             yyp += yytnamerr (yyp, yyarg[yyi++]);
1010             yyf += 2;
1011           }
1012             else
1013           {
1014             yyp++;
1015             yyf++;
1016           }
1017           }
```

```
1018        }
1019          return yysize;
1020        }
1021    }
1022    #endif /* YYERROR_VERBOSE */
1023    FF
1024
1025    /*-----------------------------------------------.
1026    | Release the memory associated to this symbol.  |
1027    `-----------------------------------------------*/
1028
1029    /*ARGSUSED*/
1030    #if (defined __STDC__ || defined __C99__FUNC__ \
1031         || defined __cplusplus || defined _MSC_VER)
1032    static void
1033    yydestruct (const char *yymsg, int yytype, YYSTYPE *yyvaluep)
1034    #else
1035    static void
1036    yydestruct (yymsg, yytype, yyvaluep)
1037        const char *yymsg;
1038        int yytype;
1039        YYSTYPE *yyvaluep;
1040    #endif
1041    {
1042      YYUSE (yyvaluep);
1043
1044      if (!yymsg)
1045        yymsg = "Deleting";
1046      YY_SYMBOL_PRINT (yymsg, yytype, yyvaluep, yylocationp);
1047
1048      switch (yytype)
1049        {
1050
1051          default:
1052        break;
1053        }
1054    }
1055    FF
1056
1057    /* Prevent warnings from -Wmissing-prototypes.  */
1058
1059    #ifdef YYPARSE_PARAM
1060    #if defined __STDC__ || defined __cplusplus
1061    int yyparse (void *YYPARSE_PARAM);
1062    #else
1063    int yyparse ();
1064    #endif
1065    #else /* ! YYPARSE_PARAM */
1066    #if defined __STDC__ || defined __cplusplus
1067    int yyparse (void);
1068    #else
1069    int yyparse ();
1070    #endif
1071    #endif /* ! YYPARSE_PARAM */
1072
1073
1074
1075    /* The look-ahead symbol.  */
1076    int yychar;
1077
1078    /* The semantic value of the look-ahead symbol.  */
1079    YYSTYPE yylval;
1080
1081    /* Number of syntax errors so far.  */
1082    int yynerrs;
1083
1084
1085
1086    /*----------.
1087    | yyparse.  |
1088    `----------*/
1089
1090    #ifdef YYPARSE_PARAM
```

```
#if (defined __STDC__ || defined __C99__FUNC__ \
     || defined __cplusplus || defined _MSC_VER)
int
yyparse (void *YYPARSE_PARAM)
#else
int
yyparse (YYPARSE_PARAM)
    void *YYPARSE_PARAM;
#endif
#else /* ! YYPARSE_PARAM */
#if (defined __STDC__ || defined __C99__FUNC__ \
     || defined __cplusplus || defined _MSC_VER)
int
yyparse (void)
#else
int
yyparse ()

#endif
#endif
{

  int yystate;
  int yyn;
  int yyresult;
  /* Number of tokens to shift before error messages enabled.  */
  int yyerrstatus;
  /* Look-ahead token as an internal (translated) token number.  */
  int yytoken = 0;
#if YYERROR_VERBOSE
  /* Buffer for error messages, and its allocated size.  */
  char yymsgbuf[128];
  char *yymsg = yymsgbuf;
  YYSIZE_T yymsg_alloc = sizeof yymsgbuf;
#endif

  /* Three stacks and their tools:
     `yyss': related to states,
     `yyvs': related to semantic values,
     `yyls': related to locations.

     Refer to the stacks thru separate pointers, to allow yyoverflow
     to reallocate them elsewhere.  */

  /* The state stack.  */
  yytype_int16 yyssa[YYINITDEPTH];
  yytype_int16 *yyss = yyssa;
  yytype_int16 *yyssp;

  /* The semantic value stack.  */
  YYSTYPE yyvsa[YYINITDEPTH];
  YYSTYPE *yyvs = yyvsa;
  YYSTYPE *yyvsp;



#define YYPOPSTACK(N)   (yyvsp -= (N), yyssp -= (N))

  YYSIZE_T yystacksize = YYINITDEPTH;

  /* The variables used to return semantic value and location from the
     action routines.  */
  YYSTYPE yyval;


  /* The number of symbols on the RHS of the reduced rule.
     Keep to zero when no symbol should be popped.  */
  int yylen = 0;

  YYDPRINTF ((stderr, "Starting parse\n"));

  yystate = 0;
  yyerrstatus = 0;
```

```
1164      yynerrs = 0;
1165      yychar = YYEMPTY;     /* Cause a token to be read.  */
1166
1167      /* Initialize stack pointers.
1168         Waste one element of value and location stack
1169         so that they stay on the same level as the state stack.
1170         The wasted elements are never initialized.  */
1171
1172      yyssp = yyss;
1173      yyvsp = yyvs;
1174
1175      goto yysetstate;
1176
1177    /*------------------------------------------------------------.
1178    | yynewstate -- Push a new state, which is found in yystate.  |
1179    `------------------------------------------------------------*/
1180     yynewstate:
1181      /* In all cases, when you get here, the value and location stacks
1182         have just been pushed.  So pushing a state here evens the stacks.  */
1183      yyssp++;
1184
1185     yysetstate:
1186      *yyssp = yystate;
1187
1188      if (yyss + yystacksize - 1 <= yyssp)
1189        {
1190          /* Get the current used size of the three stacks, in elements.  */
1191          YYSIZE_T yysize = yyssp - yyss + 1;
1192
1193    #ifdef yyoverflow
1194          {
1195        /* Give user a chance to reallocate the stack.  Use copies of
1196           these so that the &'s don't force the real ones into
1197           memory.  */
1198        YYSTYPE *yyvs1 = yyvs;
1199        yytype_int16 *yyss1 = yyss;
1200
1201
1202        /* Each stack pointer address is followed by the size of the
1203           data in use in that stack, in bytes.  This used to be a
1204           conditional around just the two extra args, but that might
1205           be undefined if yyoverflow is a macro.  */
1206        yyoverflow (YY_("memory exhausted"),
1207                    &yyss1, yysize * sizeof (*yyssp),
1208                    &yyvs1, yysize * sizeof (*yyvsp),
1209
1210                    &yystacksize);
1211
1212        yyss = yyss1;
1213        yyvs = yyvs1;
1214          }
1215    #else /* no yyoverflow */
1216    # ifndef YYSTACK_RELOCATE
1217          goto yyexhaustedlab;
1218    # else
1219          /* Extend the stack our own way.  */
1220          if (YYMAXDEPTH <= yystacksize)
1221        goto yyexhaustedlab;
1222          yystacksize *= 2;
1223          if (YYMAXDEPTH < yystacksize)
1224        yystacksize = YYMAXDEPTH;
1225
1226          {
1227        yytype_int16 *yyss1 = yyss;
1228        union yyalloc *yyptr =
1229          (union yyalloc *) YYSTACK_ALLOC (YYSTACK_BYTES (yystacksize));
1230        if (! yyptr)
1231          goto yyexhaustedlab;
1232        YYSTACK_RELOCATE (yyss);
1233        YYSTACK_RELOCATE (yyvs);
1234
1235    #  undef YYSTACK_RELOCATE
1236        if (yyss1 != yyssa)
```

```
1237          YYSTACK_FREE (yyss1);
1238          }
1239    # endif
1240    #endif /* no yyoverflow */
1241
1242          yyssp = yyss + yysize - 1;
1243          yyvsp = yyvs + yysize - 1;
1244
1245
1246          YYDPRINTF ((stderr, "Stack size increased to %lu\n",
1247              (unsigned long int) yystacksize));
1248
1249          if (yyss + yystacksize - 1 <= yyssp)
1250        YYABORT;
1251          }
1252
1253      YYDPRINTF ((stderr, "Entering state %d\n", yystate));
1254
1255      goto yybackup;
1256
1257    /*-----------.
1258    | yybackup.  |
1259    `-----------*/
1260    yybackup:
1261
1262      /* Do appropriate processing given the current state.  Read a
1263         look-ahead token if we need one and don't already have one.  */
1264
1265      /* First try to decide what to do without reference to look-ahead token.  */
1266      yyn = yypact[yystate];
1267      if (yyn == YYPACT_NINF)
1268        goto yydefault;
1269
1270      /* Not known => get a look-ahead token if don't already have one.  */
1271
1272      /* YYCHAR is either YYEMPTY or YYEOF or a valid look-ahead symbol.  */
1273      if (yychar == YYEMPTY)
1274        {
1275          YYDPRINTF ((stderr, "Reading a token: "));
1276          yychar = YYLEX;
1277        }
1278
1279      if (yychar <= YYEOF)
1280        {
1281          yychar = yytoken = YYEOF;
1282          YYDPRINTF ((stderr, "Now at end of input.\n"));
1283        }
1284      else
1285        {
1286          yytoken = YYTRANSLATE (yychar);
1287          YY_SYMBOL_PRINT ("Next token is", yytoken, &yylval, &yylloc);
1288        }
1289
1290      /* If the proper action on seeing token YYTOKEN is to reduce or to
1291         detect an error, take that action.  */
1292      yyn += yytoken;
1293      if (yyn < 0 || YYLAST < yyn || yycheck[yyn] != yytoken)
1294        goto yydefault;
1295      yyn = yytable[yyn];
1296      if (yyn <= 0)
1297        {
1298          if (yyn == 0 || yyn == YYTABLE_NINF)
1299        goto yyerrlab;
1300          yyn = -yyn;
1301          goto yyreduce;
1302        }
1303
1304      if (yyn == YYFINAL)
1305        YYACCEPT;
1306
1307      /* Count tokens shifted since error; after three, turn off error
1308         status.  */
1309      if (yyerrstatus)
```

```
1310      yyerrstatus--;
1311
1312    /* Shift the look-ahead token.  */
1313    YY_SYMBOL_PRINT ("Shifting", yytoken, &yylval, &yylloc);
1314
1315    /* Discard the shifted token unless it is eof.  */
1316    if (yychar != YYEOF)
1317      yychar = YYEMPTY;
1318
1319    yystate = yyn;
1320    *++yyvsp = yylval;
1321
1322    goto yynewstate;
1323
1324
1325  /*-----------------------------------------------------------.
1326  | yydefault -- do the default action for the current state.  |
1327  `-----------------------------------------------------------*/
1328  yydefault:
1329    yyn = yydefact[yystate];
1330    if (yyn == 0)
1331      goto yyerrlab;
1332    goto yyreduce;
1333
1334
1335  /*-----------------------------.
1336  | yyreduce -- Do a reduction.  |
1337  `-----------------------------*/
1338  yyreduce:
1339    /* yyn is the number of a rule to reduce with.  */
1340    yylen = yyr2[yyn];
1341
1342    /* If YYLEN is nonzero, implement the default value of the action:
1343       `$$ = $1'.
1344
1345       Otherwise, the following line sets YYVAL to garbage.
1346       This behavior is undocumented and Bison
1347       users should not rely upon it.  Assigning to YYVAL
1348       unconditionally makes the parser a bit smaller, and it avoids a
1349       GCC warning that YYVAL may be used uninitialized.  */
1350    yyval = yyvsp[1-yylen];
1351
1352
1353    YY_REDUCE_PRINT (yyn);
1354    switch (yyn)
1355      {
1356          case 2:
1357  #line 33 "yacc_micro.y"
1358      {
1359                                                  printf("Compilado
                                                    correctamente!\n");
1360                                              }
1361      break;
1362
1363    case 5:
1364  #line 40 "yacc_micro.y"
1365      {
1366                                                          //Rutina semántica:
                                                            Comprobar largo de
                                                            variable.
1367
                                                            if(identificadorValido
                                                            ((yyvsp[(1) -
                                                            (4)].identificador))
                                                            == 0)YYABORT;
1368
1369
1370                                                  }
1371      break;
1372
1373    case 6:
1374  #line 47 "yacc_micro.y"
1375      { }
```

```
      break;

    case 9:
#line 54 "yacc_micro.y"
      {

if(identificadorValido((yyvsp[(1) - (3)].identificador)) == 0)
                                                                    YYABORT;
                                                                }
      break;

    case 14:
#line 65 "yacc_micro.y"
      {
                                                    if(identificadorValido((yyvsp[(1) -
                                                    (1)].identificador)) == 0)
                                                        YYABORT;

                                                }
      break;

    case 16:
#line 72 "yacc_micro.y"
      { ; }
      break;


/* Line 1267 of yacc.c.  */
#line 1403 "y.tab.c"
        default: break;
      }
    YY_SYMBOL_PRINT ("-> $$ =", yyr1[yyn], &yyval, &yyloc);

    YYPOPSTACK (yylen);
    yylen = 0;
    YY_STACK_PRINT (yyss, yyssp);

    *++yyvsp = yyval;


    /* Now `shift' the result of the reduction.  Determine what state
       that goes to, based on the state we popped back to and the rule
       number reduced by.  */

    yyn = yyr1[yyn];

    yystate = yypgoto[yyn - YYNTOKENS] + *yyssp;
    if (0 <= yystate && yystate <= YYLAST && yycheck[yystate] == *yyssp)
      yystate = yytable[yystate];
    else
      yystate = yydefgoto[yyn - YYNTOKENS];

    goto yynewstate;


/*------------------------------------.
| yyerrlab -- here on detecting error |
`------------------------------------*/
yyerrlab:
    /* If not already recovering from an error, report this error.  */
    if (!yyerrstatus)
      {
        ++yynerrs;
#if ! YYERROR_VERBOSE
        yyerror (YY_("syntax error"));
#else
        {
      YYSIZE_T yysize = yysyntax_error (0, yystate, yychar);
      if (yymsg_alloc < yysize && yymsg_alloc < YYSTACK_ALLOC_MAXIMUM)
        {
          YYSIZE_T yyalloc = 2 * yysize;
          if (! (yysize <= yyalloc && yyalloc <= YYSTACK_ALLOC_MAXIMUM))
            yyalloc = YYSTACK_ALLOC_MAXIMUM;
```

```
1447          if (yymsg != yymsgbuf)
1448            YYSTACK_FREE (yymsg);
1449          yymsg = (char *) YYSTACK_ALLOC (yyalloc);
1450          if (yymsg)
1451            yymsg_alloc = yyalloc;
1452          else
1453            {
1454          yymsg = yymsgbuf;
1455          yymsg_alloc = sizeof yymsgbuf;
1456            }
1457        }
1458
1459      if (0 < yysize && yysize <= yymsg_alloc)
1460        {
1461          (void) yysyntax_error (yymsg, yystate, yychar);
1462          yyerror (yymsg);
1463        }
1464      else
1465        {
1466          yyerror (YY_("syntax error"));
1467          if (yysize != 0)
1468            goto yyexhaustedlab;
1469        }
1470        }
1471   #endif
1472      }
1473
1474
1475
1476    if (yyerrstatus == 3)
1477      {
1478        /* If just tried and failed to reuse look-ahead token after an
1479       error, discard it.  */
1480
1481        if (yychar <= YYEOF)
1482      {
1483        /* Return failure if at end of input.  */
1484        if (yychar == YYEOF)
1485          YYABORT;
1486      }
1487        else
1488      {
1489        yydestruct ("Error: discarding",
1490                yytoken, &yylval);
1491        yychar = YYEMPTY;
1492      }
1493      }
1494
1495    /* Else will try to reuse look-ahead token after shifting the error
1496       token.  */
1497    goto yyerrlab1;
1498
1499
1500   /*---------------------------------------------------.
1501   | yyerrorlab -- error raised explicitly by YYERROR.  |
1502   `---------------------------------------------------*/
1503   yyerrorlab:
1504
1505    /* Pacify compilers like GCC when the user code never invokes
1506       YYERROR and the label yyerrorlab therefore never appears in user
1507       code.  */
1508    if (/*CONSTCOND*/ 0)
1509       goto yyerrorlab;
1510
1511    /* Do not reclaim the symbols of the rule which action triggered
1512       this YYERROR.  */
1513    YYPOPSTACK (yylen);
1514    yylen = 0;
1515    YY_STACK_PRINT (yyss, yyssp);
1516    yystate = *yyssp;
1517    goto yyerrlab1;
1518
1519
```

```
1520  /*---------------------------------------------------------.
1521  | yyerrlab1 -- common code for both syntax error and YYERROR.  |
1522  `---------------------------------------------------------*/
1523  yyerrlab1:
1524    yyerrstatus = 3;  /* Each real token shifted decrements this.  */
1525
1526    for (;;)
1527      {
1528        yyn = yypact[yystate];
1529        if (yyn != YYPACT_NINF)
1530      {
1531        yyn += YYTERROR;
1532        if (0 <= yyn && yyn <= YYLAST && yycheck[yyn] == YYTERROR)
1533          {
1534            yyn = yytable[yyn];
1535            if (0 < yyn)
1536          break;
1537          }
1538      }
1539
1540        /* Pop the current state because it cannot handle the error token.  */
1541        if (yyssp == yyss)
1542      YYABORT;
1543
1544
1545        yydestruct ("Error: popping",
1546            yystos[yystate], yyvsp);
1547        YYPOPSTACK (1);
1548        yystate = *yyssp;
1549        YY_STACK_PRINT (yyss, yyssp);
1550      }
1551
1552    if (yyn == YYFINAL)
1553      YYACCEPT;
1554
1555    *++yyvsp = yylval;
1556
1557
1558    /* Shift the error token.  */
1559    YY_SYMBOL_PRINT ("Shifting", yystos[yyn], yyvsp, yylsp);
1560
1561    yystate = yyn;
1562    goto yynewstate;
1563
1564
1565  /*-------------------------------------.
1566  | yyacceptlab -- YYACCEPT comes here.  |
1567  `-------------------------------------*/
1568  yyacceptlab:
1569    yyresult = 0;
1570    goto yyreturn;
1571
1572  /*-----------------------------------.
1573  | yyabortlab -- YYABORT comes here.  |
1574  `-----------------------------------*/
1575  yyabortlab:
1576    yyresult = 1;
1577    goto yyreturn;
1578
1579  #ifndef yyoverflow
1580  /*-------------------------------------------------.
1581  | yyexhaustedlab -- memory exhaustion comes here.  |
1582  `-------------------------------------------------*/
1583  yyexhaustedlab:
1584    yyerror (YY_("memory exhausted"));
1585    yyresult = 2;
1586    /* Fall through.  */
1587  #endif
1588
1589  yyreturn:
1590    if (yychar != YYEOF && yychar != YYEMPTY)
1591       yydestruct ("Cleanup: discarding lookahead",
1592           yytoken, &yylval);
```

```
1593      /* Do not reclaim the symbols of the rule which action triggered
1594         this YYABORT or YYACCEPT.  */
1595      YYPOPSTACK (yylen);
1596      YY_STACK_PRINT (yyss, yyssp);
1597      while (yyssp != yyss)
1598        {
1599          yydestruct ("Cleanup: popping",
1600              yystos[*yyssp], yyvsp);
1601          YYPOPSTACK (1);
1602        }
1603    #ifndef yyoverflow
1604      if (yyss != yyssa)
1605        YYSTACK_FREE (yyss);
1606    #endif
1607    #if YYERROR_VERBOSE
1608      if (yymsg != yymsgbuf)
1609        YYSTACK_FREE (yymsg);
1610    #endif
1611      /* Make sure YYID is used.  */
1612      return YYID (yyresult);
1613    }
1614
1615
1616    #line 74 "yacc_micro.y"
1617
1618
1619    int identificadorValido(char* id)
1620    {
1621        int len = stringLength(id);
1622        if(len > VARMAXLENGTH)
1623        {
1624            printf("Error semantico: Variable '%s' con cant. caracteres %d mayor a
            32.\n", id, len);
1625            free(id);
1626            return 0;
1627        }
1628
1629        free(id); // Se debe liberar la memoria asignada en LEX con strdup().
1630        return 1;
1631    }
1632
1633    int stringLength(char* str)
1634    {
1635        int len = 0;
1636        while(str[len]) len++;
1637        return len;
1638    }
1639
1640    void yyerror(const char *str)
1641    {
1642            fprintf(stderr,"Error al compilar: %s\n",str);
1643    }
1644
1645    int yywrap()
1646    {
1647            return 1;
1648    }
1649
1650    int main(int argc, char* argv[]) {
1651        if (argc == 2)
1652        {
1653            //printf("Para analizar un archivo, ejecute: ./Micro <nombre del archivo>
            \n\n");
1654            FILE *source = fopen(argv[1], "r");
1655
1656            if (!source) {
1657                printf("Imposible abrir el archivo %s.\n", argv[1]);
1658                return -1;
1659            }
1660
1661            yyin = source;
1662        }
1663
```

```
1664        yyparse();
1665        return 0;
1666    }
1667
1668
```