```
%{
  #include<stdio.h>
  #include<string.h>
  #include<ctype.h>
  char lexema[255];
  void yyerror(char *);
%}

// Especificamos los tokens
%token NUMHEXADECIMAL
%token MAS MENOS POR ENTRE RAIZ IGUAL
%token LPAR RPAR COMA PCOMA
%token LIB HTAG LIBNAME
%token PROG PROGNAME VAR ID INI FIN

// Especificamos la gramatica
%%
code: liblst progstm varstm body;
liblst: LIB HTAG LIBNAME HTAG liblst
      |  ;
progstm: PROG PROGNAME PCOMA;
varstm: VAR idlst PCOMA;
idlst: ID
     | ID COMA idlst;
body: INI asiglst FIN;
asiglst: ID IGUAL exp PCOMA
       | ID IGUAL exp PCOMA asiglst;
exp: exp MAS term
   | exp MENOS term
   | LPAR exp RPAR
   | term;
term: term POR NUMHEXADECIMAL
    | term ENTRE NUMHEXADECIMAL
    | term raiz
    | NUMHEXADECIMAL
    | ID;
raiz: RAIZ LPAR exp RPAR;
%%

void yyerror(char *msg) {
  printf("error: %s", msg);
}

// Especificamos las reglas de los tokens
int yylex() {
  char c;
```

```
while(1) {
  c = getchar();
  // if(c == '\n') continue;
  if(isspace(c)) continue;

  if(c == '+') return MAS;
  if(c == '-') return MENOS;
  if(c == '*') return POR;
  if(c == '/') return ENTRE;
  if(c == '#') return HTAG;
  if(c == ',') return COMA;
  if(c == ';') return PCOMA;
  if(c == '(') return LPAR;
  if(c == ')') return RPAR;
  if(c == '=') return IGUAL;

  char CADENA_LIB[] = "libreria";
  if(c == CADENA_LIB[0]) {
    int i = 0, j = 0;
    do {
      lexema[i++] = c;
      c = getchar();
      j++;
    } while (c == CADENA_LIB[j] && j < 8);
    if(j == 8) {
      ungetc(c, stdin);
      lexema[i] == 0;
      return LIB;
    }
  }

  char CADENA_PROG[] = "Programa";
  if(c == CADENA_PROG[0]) {
    int i = 0, j = 0;
    do {
      lexema[i++] = c;
      c = getchar();
      j++;
    } while (c == CADENA_PROG[j] && j < 8);
    if(j == 8) {
      ungetc(c, stdin);
      lexema[i] == 0;
      return PROG;
    }
  }
```

```
    char CADENA_VAR[] = "var";
    if(c == CADENA_VAR[0]) {
      int i = 0, j = 0;
      do {
        lexema[i++] = c;
        c = getchar();
        j++;
      } while (c == CADENA_VAR[j] && j < 3);
      if(j == 3) {
        ungetc(c, stdin);
        lexema[i] == 0;
        return VAR;
      }
    }

    char CADENA_RAIZ[] = "raiz";
    if(c == CADENA_RAIZ[0]) {
      int i = 0, j = 0;
      do {
        lexema[i++] = c;
        c = getchar();
        j++;
      } while (c == CADENA_RAIZ[j] && j < 4);
      if(j == 4) {
        ungetc(c, stdin);
        lexema[i] == 0;
        return RAIZ;
      }
    }

    char CADENA_INI[] = "Inicio";
    if(c == CADENA_INI[0]) {
      int i = 0, j = 0;
      do {
        lexema[i++] = c;
        c = getchar();
        j++;
      } while (c == CADENA_INI[j] && j < 6);
      if(j == 6) {
        ungetc(c, stdin);
        lexema[i] == 0;
        return INI;
      }
    }

    char CADENA_FIN[] = "Fin";
```

```
    if(c == CADENA_FIN[0]) {
      int i = 0, j = 0;
      do {
        lexema[i++] = c;
        c = getchar();
        j++;
      } while (c == CADENA_FIN[j] && j < 3);
      if(j == 3) {
        ungetc(c, stdin);
        lexema[i] == 0;
        return FIN;
      }
    }

    // Nombres de programa caneda que inicia con mayuscula
    if(c >= 65 && c <= 90) {
      int i = 0;
      do {
        lexema[i++] = c;
        c = getchar();
      } while(isalpha(c));
      ungetc(c, stdin);
      lexema[i] == 0;
      return PROGNAME;
    }

    if((c >= 65 && c <= 70) || isdigit(c)) {
      int i = 0;
      do {
        lexema[i++] = c;
        c = getchar();
      } while((c >= 65 && c <= 70) || isdigit(c));
      ungetc(c, stdin);
      lexema[i] == 0;
      return NUMHEXADECIMAL;
    }

    if(isalpha(c)) {
      // Nombres de libreria cadena con solo minusculas
      if(c >= 97 && c <= 122) {
        int i = 0;
        do {
          lexema[i++] = c;
          c = getchar();
        } while(c >= 97 && c <= 122);
        if(c == '.') {
```

```
          lexema[i++] = c;
          c = getchar();
          if(c == 'x') {
            lexema[i++] = c;
            return LIBNAME;
          }
        }
        do {
          lexema[i++] = c;
          c = getchar();
        } while(isalnum(c));
        ungetc(c, stdin);
        lexema[i] == 0;
        return ID;
      }

      int i = 0;
      do {
        lexema[i++] = c;
        c = getchar();
      } while(isalnum(c));
      ungetc(c, stdin);
      lexema[i] == 0;
      return ID;
    }

    return c;
  }
}

int main() {
  if(!yyparse()) printf("\ncadena valida\n");
  else printf("cadena invalida\n");
  return 0;
}
```

Números hexadecimales

```
[wensespl@LAPTOP-1D2RDUOO ···/USUARIO/Desktop/Compiladores/examparcial]
$ ./a.out
2F

cadena valida
```

```
[wensespl@LAPTOP-1D2RDUOO ···/USUARIO/Desktop/Compiladores/examparcial]
⏱ 4s233ms $ ./a.out
9
```

```
┌─【wensespl@LAPTOP-1D2RDUOO ⋯/USUARIO/Desktop/Compiladores/examparcial】
└─ ⏱ 4s361ms $ ./a.out
H
error: syntax errorcadena invalida
```

```
┌─【wensespl@LAPTOP-1D2RDUOO ⋯/USUARIO/Desktop/Compiladores/examparcial】
└─ ⏱ 7s951ms $ ./a.out
1A

cadena valida
```

```
┌─【wensespl@LAPTOP-1D2RDUOO ⋯/USUARIO/Desktop/Compiladores/examparcial】
└─ ⏱ 11s261ms $ ./a.out
12

cadena valida
```

```
┌─【wensespl@LAPTOP-1D2RDUOO ⋯/USUARIO/Desktop/Compiladores/examparcial】
└─ ⏱ 3s97ms $ ./a.out
3C

cadena valida
```