%{

  #include<stdio.h>

  #include<string.h>

  char lexema[255];

  void yyerror(char \*);

%}

// Especificamos los tokens

%token NUMNAT

// Especificamos las reglas

%%

instruccion: instruccion NUMNAT;

instruccion: ;

%%

void yyerror(char \*msg) {

  printf("error: %s", msg);

}

int yylex() {

  char c;

  while(1){

    c = getchar();

    if(c == '\n') continue;

    if(c == ' ') continue;

    // Verificamos que la cadena inicie con un numero

    if(isdigit(c)) {

      int i = 0;

      do {

        lexema[i++] = c;

        c = getchar();

      } while(isdigit(c)); // Verificamos que los demas

                          // carecteres siguientes sean numeros

      ungetc(c, stdin);

      lexema[i] == 0;

      return NUMNAT; // Retornamos el token numero natural

    }

    // Sino salimos, cadena invalida

    return c;

  }

}

int main() {

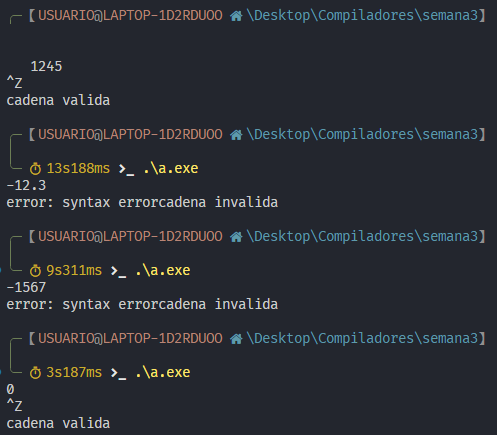
  if(!yyparse()) printf("cadena valida\n");

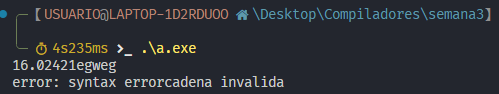
  else printf("cadena invalida\n");

  return 0;

}

Solo para números naturales





%{

  #include<stdio.h>

  #include<string.h>

  char lexema[255];

  void yyerror(char \*);

%}

// Especificamos los tokens

%token NUMDEC

// Especificamos las reglas

%%

instruccion: instruccion NUMDEC;

instruccion: ;

%%

void yyerror(char \*msg) {

  printf("error: %s", msg);

}

int yylex() {

  char c;

  while(1){

    c = getchar();

    if(c == '\n') continue;

    if(c == ' ') continue;

    // Verificamos que inicie con un digito o '-'

    if(isdigit(c) || c == '-') {

      int i = 0;

      do {

        lexema[i++] = c;

        c = getchar();

      } while(isdigit(c)); // Verificamos que los siguientes

                          // sean digitos

      // Verficamos que despues siga un '.'

      if (c == '.') {

        lexema[i++] = c;

        c = getchar();

        // Verificamos que despues del '.' continue un digito

        if(isdigit(c)) {

          do {

            lexema[i++] = c;

            c = getchar();

          } while(isdigit(c)); // Verificamos que los siguientes

                              // sean digitos

          ungetc(c, stdin);

          lexema[i] == 0;

          return NUMDEC; // Retornamos token numero decimal

        }

      }

      ungetc(c, stdin);

      lexema[i] == 0;

    }

    // Sino salimos porque la cadena es invalida

    return c;

  }

}

int main() {

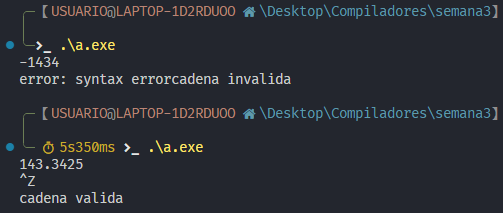
  if(!yyparse()) printf("cadena valida\n");

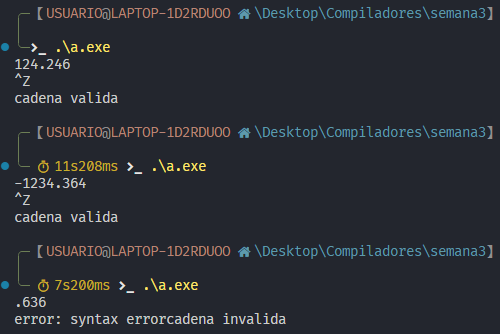
  else printf("cadena invalida\n");

  return 0;

}

Solo para números decimales





%{

  #include<stdio.h>

  #include<string.h>

  char lexema[255];

  void yyerror(char \*);

%}

// Especificamos los tokens

%token NUMDEC NUMNAT

// Especificamos las reglas

%%

instruccion: instruccion NUMDEC;

instruccion: instruccion NUMNAT;

instruccion: ;

%%

void yyerror(char \*msg) {

  printf("error: %s", msg);

}

int yylex() {

  char c;

  while(1){

    c = getchar();

    if(c == '\n') continue;

    if(c == ' ') continue;

    // Si inicia con digito

    if(isdigit(c)) {

      int i = 0;

      do {

        lexema[i++] = c;

        c = getchar();

      } while(isdigit(c)); // Verificamos que los sigientes

                          // sean digitos

      // si continua un '.'

      if (c == '.') {

        lexema[i++] = c;

        c = getchar();

        // Verificamos que el sigiente sea un digito

        if(isdigit(c)) {

          do {

            lexema[i++] = c;

            c = getchar();

          } while(isdigit(c)); // Verificamos que los sigientes

                              // sean digitos

          ungetc(c, stdin);

          lexema[i] == 0;

          return NUMDEC; // Retornamos token numero decimal

        }

        ungetc(c, stdin);

        lexema[i] == 0;

        // Salimos cadena invalida

        return c;

      }

      ungetc(c, stdin);

      lexema[i] == 0;

      return NUMNAT; // Retornamos token numero natural

    }

    // Si inicia con '-'

    if(c == '-') {

      int i = 0;

      lexema[i++] = c;

      c = getchar();

      // Verificamos que el sigiente sea digito

      if(isdigit(c)) {

        do {

          lexema[i++] = c;

          c = getchar();

        } while(isdigit(c)); // Verificamos que los

                            // sigientes sean digitos

        // Si es '.'

        if (c == '.') {

          lexema[i++] = c;

          c = getchar();

          // Verificamos que el sigiente sea digito

          if(isdigit(c)) {

            do {

              lexema[i++] = c;

              c = getchar();

            } while(isdigit(c)); // Verificamos que los

                                // sigientes sean digitos

            ungetc(c, stdin);

            lexema[i] == 0;

            return NUMDEC; // Retornamos token numero decimal

          }

        }

      }

      ungetc(c, stdin);

      lexema[i] == 0;

    }

    // Salimos cadena invalida

    return c;

  }

}

int main() {

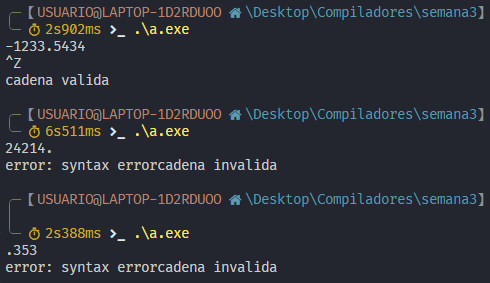
  if(!yyparse()) printf("cadena valida\n");

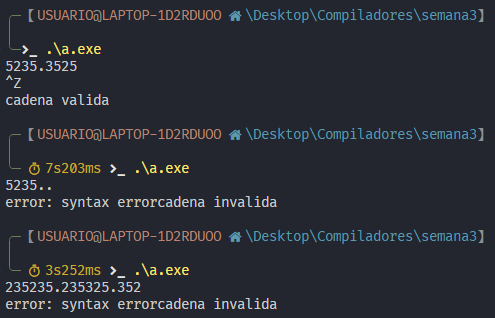
  else printf("cadena invalida\n");

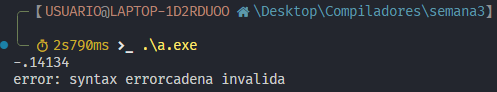
  return 0;

}

Para ambos tipos de tokens (números decimales y naturales)







Penadillo Lazares Wenses Johan