Hola chicos

Les envío la devolución que les debía. El trabajo tiene cosas que se deben corregir para la próxima entrega, y así aprobar esta primera etapa.

Creo que el mayor problema se encuentra en la gramática de expresiones que está mal planteada. En la devolución que les copio a continuación verán qué problemas detecté, y además, podrán usar algunos casos de prueba que les adjunto para ver problemas específicos.

En la devolución hay puntos resaltados en amarillo. En esos párrafos están las cosas que sí o sí deben resolver. En otros items hay aclaraciones o sugerencias que pueden tener en cuenta para mejorar el compilador.

Si hay algún comentario que no se entiende, me avisan.

Con respecto a los casos de prueba, fueron creados a partir de los que ustedes presentaron, y puede suceder que el problema esté en algún detalle sintáctico que no quedó bien. Pero, como los errores informados en muchos casos sólo indican syntax error, no es directo encontrar por qué dichos ejemplos dan error.

DEvolución:

**Trabajo en general**

·        . En la salida ponen: Estructuras sintácticas, pero están informando también los tokens detectados, en algunos casos. Sería más correcto decir: Estructuras léxicas y sintácticas, por ejemplo

·        Informa todo como estructuras del parser. Pero si dice se detectó una constante entera, se trata del Léxico no del parser

~~·        NO informan los tipos de tokens (solo informan los números que los identifican)~~

**Análisis Léxico**

~~·        .Están informando sólo los números de token. Se pidió informar los tipos de tokens. Ejemplo: identificador, no 257.~~

·        Constantes

·        .Enteras ui16

o   Rangos ok

o   Las constantes enteras negativas deben ser informadas como error -**CONSULTAR**

·        Flotantes f64

o   Sería bueno normalizar las constantes de punto flotante. Por ejemplo 2.1 es la misma constante que .21D1

o    Rangos ok

~~o   Revisar el tratamiento de constantes negativas: Si aparece una constante positiva, por ejemplo 2., y luego -2., pierden la constante positiva. Y si se coloca una constante negativa grande, la deja en la TdeS como -0.0 (ver programa\_lexico\_flotantes\_neg.txt)~~

·        Identificadores

~~·        .Si bien detectan correctamente el identificador con más de 25 caracteres, el reporte debe ser como warning, ya que se está truncando, y entregando como identificador válido.~~

·        Operadores

·        No se puede chequear sin tener la tabla de equivalencia del nro con tipo de token

**CONSULTAR**

·        Palabras reservadas

·        No se puede chequear sin tener la tabla de equivalencia del nro con tipo de token

**CONSULTAR**

~~·        Las palabras reservadas son informadas correctamente, pero también quedan registradan en la TdeS, como si fueran identificadores. No deben ser registradas en la Tabla..~~

·        Comentarios bien y mal definidos.

·        .Cuando queda un comentario sin cerrar, habría que informarlo con un warning, ya que esto hace que el resto del código quede adentro del comentario

·        Cadenas

~~·        En el caso de prueba presentado para cadenas, el error de la cadena sin cerrar se informa 2 veces~~

**CAMBIO MATRIZ DE TRANSICION FILA 11 COL 3**

~~·        Como las cadenas se guardan sin las comillas, se confunden con identificadores. Si hay un id con lexema hola, y una cadena ‘hola’, queda una sola entrada con el lexema hola.~~

**SE ARREGLO EN EL SINTACTICO AGREGANDO EL ATRIBUTO USO**

**Análisis Sintáctico**

**Estructuras Sintácticas**

~~·      Informa que se realizó conversión explícita en líneas de asignaciones sin conversión. Esta información está mal ubicada en la gramática~~

~~·~~

·      Declaraciones de variables

·        .ok

·      Declaraciones de funciones

ok.

·      Asignaciones

·        ok.

·      Expresiones aritméticas

~~·        No reconoce correctamente las expresiones (ver Programa\_asignaciones\_convexpl.txt)~~

~~·        La gramática exige que un factor dentro de un término esté entre (). Eso es incorrecto.~~

·      Conv explícitas

~~·        No reconoce correctamente las expresiones con conversiones (ver Programa\_asignaciones\_convexpl.txt)~~

~~·        .Las conversiones se ubican en la gramática a nivel expresión. Esto provoca que no puedan estar en cualquier lugar de una expresión. Deben ser tratadas del mismo modo que una invocación a función, a nivel factor~~

·

·      Sentencia de salida de mensajes por pantalla

·        .ok

·        Sentencia IF

~~·        Da error de sintaxis en un if con else (ver Programa\_if.txt)~~

~~Sin embargo, para otro ejemplo funcionó correctamente. Quizá hay algún problema de sintaxis que no se detecta correctamente.~~

**ERROR EN EJEMPLO – LLAVE DE CIERRE DE MAS**

·        Sentencia iterativa (do until)

·        ok

·        Sentencia iterativa (do until) con etiquetado

~~·        Break con etiqueta aceptado pero no informado~~

·        ok

·        Anidamiento de sentencias de control

·        No lo reconoce correctamente. (ver Programa\_do until\_conif.txt y Programa\_do until\_etiqueta\_anidado.txt)

**FALTA CHEQUEAR SEGUNDO CASO**

·        Sentencia BREAK

~~·        Aceptado pero no informado~~

·        defer

·        ok

·        discard

·        OK

·        Pero si se hace la invocación sin discard, da syntax error sin indicar de qué se trata el error. (ver Programa\_discard.txt)

**CHEQUEAR FUNCION SIN DISCARD**

·        Invocación de funciones

·        ok

**Errores:**

·        **Si se coloca un ‘.’ solo, lo informa correctamente como un error léxico, pero da syntax error y aborta la compilación. Lo mismo ocurre si aparece la D del exponente, pero no hay dígitos a continuación**

**Léxicos**

·        .Cuando hay un símbolo no reconocido informa que no es reconocido por la gramática. Sería mejor informalo como no reconocido por el compilador, o el lenguaje.

**Sintácticos**

·        .Informa falta de llave de cierre en un bloque que tiene su llave

·        Algunos errores los marca como syntax error sin ninguna descripción

**Tabla de símbolos**

·        .La información mostrada no está completa, o se registra información para algunos símbolos y para otros no. Para las constantes, muestran el tipo de token. Pero para los identificadores, no. De El  tipo de token no necesita ser guardado en la Tabla, pero si lo registran para un tipo de tokens, para los otros también se debería registrar.

~~·        Para el caso de las constantes, si bien consideran diferente tipo de token para cte entera que para constante flotante, deben registrar el tipo (i16 o f64) en la entrada correspondiente en la Tabla.~~

~~·        Las palabras reservadas no se guardan en la TdeS~~

**Casos de prueba**

·        bien

**Informe**

·        Se indican acciones semánticas que tratan errores. Para los casos en que la acción se encargue de modificar el token con inserción/borrado/reemplazo para entregar un token válido , se debe reportar el problema como warning, no como error

Informen los tokens con sus descripciones, pero no es necesario que sigan informando las estr. sintácticas, ya que éstas se reflejarán en el código intermedio que se genera en la tercera etapa.

Les reitero que, si no se entiende algún comentario, me escriban.

Con respecto a las expresiones, replanteen esa parte de la gramática y me la envían, así no siguen arrastrando los errores para la entrega.

Saludos,

Marcela