Proyecto Final compiladores e interpretes

El proyecto final de la materia consiste en modificar el código del compilador tiny que tiene como código objetivo la maquina virtual de tiny (TM), disponible en su versión de java creada en el laboratorio en el repositorio git (http://lmgtfy.com/?q=git) pudiendo usar herramientas como tortoise git para descargarlo o ir directamente a su pagina web:

git://github.com/xiul/ejemplos-clases-compiladores-e-interpretes.git

Para esto deberán comenzar a modificar el código existente para cumplir con los siguientes requisitos:

1. Permitir que el lenguaje acepte secuencias de sentencias terminadas en punto y coma, al contrario de lo actualmente existente.
2. Añadir los operadores relacionales >, <=, >=, !=
3. Añadir los operadores lógicos && (y lógico) y || (o lógico).
4. Se añadirán los testigo **mainbegin** y **end** al código principal de la maquina, quedando ahora los programas delimitados de la forma siguiente.

```
mainbegin
{codigo}
end
```

5. Se debe añadir el ciclo for con la estructura siguiente

```
for( asignación , comprobación,expresión)
  {código del ciclo}
endfor
```

donde:

   a. Asignación será una asignación de una expresión aritmética.
   b. Comprobación será una expresión lógica similar a la aceptada por los if.
   c. Expresión será la expresión aritmética de asignación a ejecutar cada vez que se llega al final de ciclo.
   d. Su ejecución será así, en un inicio se lleva a cabo la asignación (solo la primera vez), posteriormente se ejecuta el código del ciclo y luego la expresión para posteriormente llevar a cabo la comprobación.
   e. Sera labor del programador y no del analizador semántico el ver si el ciclo tiene sentido o no (infinito, mal estructurado, etc), se asume se usara de forma similar a los ciclos normales de c, for(i=n+3; i<=a+3; i:=i+1)

6. Se debe añadir el ciclo mientras con la siguiente estructura

```
while( comprobación )
  {código del ciclo}
endwhile
```

donde:

   a. Comprobación será una expresión lógica similar a la aceptada por los if.
   b. Su ejecución se llevara a cabo asi: se ejecuta la comprobación, en caso de resultar verdadera se ejecuta el código del ciclo, en caso de ser falsa se continúa en la instrucción siguiente al ciclo.

7. Añadir declaración de variabl(es) (solo int) y soporte de vectores, como se ejemplifica:
   a. **Declaración:** int variable, variable2[tamano]; donde tamaño representa una constante entera
   b. **Uso de vectores:** en una asignación en su parte izquierda o derecha como cualquier otra variable donde su índice puede ser una operación por ejemplo i+1 o una constante entera por ejemplo 4.

c. **Su uso será global pero su declaración no:** es decir las variables podrán ser declaradas solo al inicio de los programas añadiendo ahora el testigo **endvar** para delimitar la zona donde se declararan las variables, por ejemplo:

```
Mainbegin
  int x,y,z;
endvar
 read x;
 read y;
 z:=x+y;
 write z
end
```

8. Añadir un verificador semántico para:
    a. No permitir el uso de variables no declaradas en programas o su uso de forma incorrecta, por ejemplo usar una variable simple como vector o viceversa.
    b. Permitir solo expresiones lógicas en las sentencia if, definiendo como expresión lógica aquella que puede devolver el valor verdadero o falso.
    c. Debido a que solo tenemos variables enteras, no se deberá permitir almacenar valores lógicos en las mismas o en otras palabras, las asignaciones solo aceptan expresiones de tipo aritmético.
    d. Si se hace el llamado a una función esta debe existir.
    e. Este código debe ser llevado a cabo de forma manual en java como un recorrido especial al AST luego de su construcción y no mediante añadidos al código de CUP generado.
9. Se deben implementar funciones que cumplan con los siguientes requisitos:
    a. Las variables que usaran las funciones son las mismas declaradas en el cuerpo principal entre los tokens **mainbegin** {aca} **endvar**.
    b. Las funciones no retornaran valores debido a que las variables son globales y tampoco tendrán parámetros por la misma razón.
    c. El cuerpo de una función quedara definido de la siguiente forma:

```
function nombre
begin
  {código aqui}
End
```

    d. La posición de las funciones será debajo del código principal **mainbegin** {..} **endvar** {aca}, se debe ver prestar atención a la parte d del apartado 7.

El trabajo debe ser presentado en equipos de máximo 4 personas, donde **TODOS** los integrantes **deben trabajar** en el proyecto y conocer como se hace cada parte del mismo, y debido a que la universidad además de enseñarles conocimientos técnicos, les debe enseñar algún tipo de valores personales a sus estudiantes, como por ejemplo una ética mínima sobretodo a los estudiantes que ya están tan cerca de culminar sus estudios, el hecho de entregar un compilador cuyo funcionamiento interno EN **CUALQUIER ASPECTO** sea una incógnita para **UNO** o todos los integrantes de un equipo conllevara a la perdida de nota correspondiente al equipo (mínimo 50% de la nota del proyecto A TODOS SUS INTEGRANTES) y en caso de ser una falta considerada grave, se elevara el caso hasta las instancias correspondientes en la universidad para la aplicación del reglamento interno. Aunque suena redundante les recuerdo que se validara mediante una entrega en vivo con asistencia física de TODOS los integrantes del equipo, el trabajo de los mismos para verificar lo anteriormente expuesto, y en caso de no asistir el estudiante no tendrá nota en el proyecto final de la materia.

FECHA ENTREGA: Jueves 26 de Abril 2012 NO HAY PRORROGA.