TRABAJO PRÁCTICO COMPILADOR

CONSIDERACIONES GENERALES

Es necesario cumplir con las siguientes consideraciones para evaluar el TP.

1. Cada grupo deberá desarrollar el compilador teniendo en cuenta:
   - Todos los temas comunes.
   - El tema especial según el número de tema asignado al grupo.
   - El método de generación intermedia que le sea especificado a cada grupo

2. Se fijarán puntos de control con fechas y consignas determinadas

3. Todos los ejecutables deberán correr sobre Windows.

PRIMERA ENTREGA

OBJETIVO: Realizar un analizador sintáctico utilizando las herramientas FLEX y BISON. El programa ejecutable deberá mostrar por pantalla las reglas sintácticas que va analizando el parser en base a un archivo de entrada (prueba.txt). Las impresiones deben ser claras. Las reglas que no realizan ninguna acción no deben generar salida.

Se deberá entregar una carpeta con nombre: **GrupoXX** que incluirá:

- El archivo flex que se llamará **Lexico.l**
- El archivo bison que se llamará **Sintactico.y**
- El archivo ejecutable que se llamará **Primera.exe**
  De no ser posible el envío de un archivo ejecutable deberán renombrarse de la siguiente manera: Primera.exe como Primera.e
- Un archivo de pruebas generales que se llamará **prueba.txt** y que dispondrá de un lote de pruebas generales que abarcará todos los temas especiales y comunes.(No deberán faltar selecciones y ciclos anidados, temas especiales, verificación de cotas para las constantes, chequeo de longitud de los nombres de los identificadores, comentarios)
- Un archivo con la tabla de símbolos **ts.txt**

Todo el material deberá ser enviado a: **lenguajesycompiladores@gmail.com**
Asunto : NombredelDocente_GrupoXX (Nombre de Pila del Docente)
Fecha de entrega: 03/10/2016

SEGUNDA ENTREGA

OBJETIVO: Realizar un generador de código intermedio utilizando el archivo BISON generado en la primera entrega. El programa ejecutable deberá procesar el archivo de entrada (prueba.txt) y devolver el código intermedio del mismo junto con la tabla de símbolos.

Se deberá entregar una carpeta con nombre: **GrupoXX** que incluirá:

- El archivo flex que se llamará **Lexico.l**
- El archivo bison que se llamará **Sintactico.y**
- El archivo ejecutable que se llamará **Segunda.exe**
  De no ser posible el envío de un archivo ejecutable deberán renombrarse de la siguiente manera: Segunda.exe como Segunda.e
- Un archivo de pruebas generales que se llamará **prueba.txt** y que dispondrá de un lote de pruebas generales que abarcará todos los temas especiales y comunes.
- Un archivo con la tabla de símbolos **ts.txt**
- Un archivo con la notación intermedia que se llamará **intermedia.txt** y que contiene el código intermedio

Todo el material deberá ser enviado a: **lenguajesycompiladores@gmail.com**
Asunto : NombredelDocente_GrupoXX (Nombre de Pila del Docente)
Fecha de entrega: 17/10/2016

ENTREGA FINAL

OBJETIVO: Realizar un compilador utilizando el archivo generado en la segunda entrega. El programa ejecutable deberá procesar el archivo de entrada (prueba.txt) , compilarlo y ejecutarlo.

Se deberá entregar una carpeta con nombre: **GrupoXX** que incluirá:

- El archivo flex que se llamará **Lexico.l**
- El archivo bison que se llamará **Sintactico.y**
- El archivo ejecutable del compilador que se llamará **Grupoxx.exe** y que generará el código assembler final que se llamará **Final.asm**

  De no ser posible el envío de un archivo ejecutable deberán renombrarse de la siguiente manera: Grupoxx.exe como Grupoxx.e
- Un archivo de pruebas generales que se llamará **prueba.txt** y que dispondrá de un lote de pruebas generales que abarcará :
  - Asignaciones
  - Selecciones
  - Impresiones
  - Un tema especial que se asignará por grupo

- Un archivo por lotes (**Grupoxx.bat**) que incluirá las sentencias necesarias para compilar con TASM y TLINK el archivo **Final.asm** generado por el compilador

  De no ser posible el envío de los archivos ejecutables deberán renombrarse de la siguiente manera: Grupoxx.bat como Grupoxx.b

En todos los casos el compilador **Grupoxx.exe** deberá generar los archivos **intermedia.txt** y **Final.asm**

Todo el material deberá ser enviado a: **lenguajesycompiladores@gmail.com**
Asunto : NombredelDocente_GrupoXX (Nombre de Pila del Docente)
Fecha de entrega: 07/11/2016

TEMAS COMUNES

ITERACIONES
Implementación de ciclo *REPEAT*

DECISIONES
Implementación de *IF*

ASIGNACIONES
Asignaciones simples *A:=B*

TIPO DE DATOS
Constantes numéricas

- reales (32 bits)
- enteros (16 bits)

El separador decimal será el punto "."

Ejemplo:

```
a = 99999.99
a = 99.
a = .9999
```

Constantes string
Constantes de 30 caracteres alfanuméricos como máximo, limitada por comillas (" ") ,de la forma *"XXXX"*

Ejemplo:

```
b = "@sdADaSjfla%dfg"
b = "asldk  fh sjf"
```

Las constantes deben ser reconocidas y validadas en el *analizador léxico*, de acuerdo a su tipo.

VARIABLES

Variables numéricas
Estas variables reciben valores numéricos tales como constantes numéricas, variables numéricas u operaciones que arrojen un valor numérico, del lado derecho de una asignación.

Variables string
Estas variables pueden recibir una constante string, una variable string, o la concatenación de 2 (máximo) tipos strings (máximo) ya sean constantes o variables.
El operador de concatenación será el símbolo *"++"*

Las variables no guardan su valor en tabla de símbolos.
Las asignaciones deben ser permitidas, solo en los casos en los que los tipos son compatibles, caso contrario deberá desplegarse un error.

COMENTARIOS
Deberán estar delimitados por "-/" y "/-" y podrán estar anidados en un solo nivel.

Ejemplo1:

```
-/ Realizo una selección /-
```

```
            IF (a <= 30)
                  b = ”correcto” -/ asignación string /-
ENDIF
```

Ejemplo2:

```
-/ Así son los comentarios en el 2°Cuat de LyC -/ Comentario /-  /-
```

Los comentarios se ignoran de manera que no generan un componente léxico o token

ENTRADA Y SALIDA
Las salidas y entradas por teclado se implementaran como se muestra en el siguiente ejemplo:

Ejemplo:

```
WRITE “ewr”     --/ donde “ewr” debe ser una cte string
READ base       --/ donde base es una variable
WRITE var1      --/ donde var1 es una variable definida previamente
```

CONDICIONES
Las condiciones para un constructor de ciclos o de selección pueden ser simples *(a < b)* o múltiples.
Las condiciones múltiples pueden ser hasta **dos** condiciones simples ligadas a través del operador lógico *(AND, OR)* o una condición simple con el operador lógico *NOT*

DECLARACIONES
Todas las variables deberán ser declaradas dentro de un bloque especial para ese fin, delimitado por las palabras reservadas DECVAR y ENDDEC, siguiendo el siguiente formato:

DECVAR
Línea_de_Declaración_de_Tipos
ENDDEC

Cada *Línea_de_Declaración_de_Tipos* tendrá la forma: < *Lista de Variables> : Tipo de Dato*

La *Lista de Variables* debe ser una lista de variables separadas por comas.
Pueden existir varias líneas de declaración de tipos, incluso utilizando más de una línea para el mismo tipo.

Ejemplos de formato:

```
DEFVAR
      a1, b1 : FLOAT
      variable1 : STRING
      p1, p2, p3 : FLOAT
      list(3): ARRAY

ENDDEF
```

TEMAS ESPECIALES

1. **Alias**
   Este comando, permite añadir otro nombre para un identificador existente. Si el identificador original no se encuentra definido, el comando arroja error.

   ALIAS nuevo_nombre % variable

   nuevo_nombre: NO debe existir con anterioridad, y a partir de este momento no puede usarse en otra declaración o como un alias.
   variable: debe estar declarada previamente.

2. **Between**
   Esta función del lenguaje, tomará como entrada una variable numérica y dos expresiones numéricas. Devolverá verdadero o falso según la variable enunciada se encuentre dentro del rango definido por ambos límites.

Esta función será utilizada en las condiciones, presentes en ciclos y selecciones.

BETWEEN(variable1, [expresion1; expresion2])

Ejemplo:

BETWEEN (a, [2 ; a*(b+4)])
BETWEEN (z, [2.3 ; 11.22])

3. Average

Esta función del lenguaje, tomará como entrada una lista de expresiones numéricas y devolverá el promedio calculado a partir de la evaluación de los componentes de dicha lista.
Esta función será utilizada en cualquier expresión del lenguaje.

AVG([lista de expresiones])

donde lista de expresiones será una secuencia de expresiones numéricas separadas por coma (,) y delimitada por corchetes

Ejemplo:

AVG ([2 , a+b , c*(d+e) , 48])
AVG ([2.3 , 1.22])

4. Factorial/Combinatorio

La función especial *Factorial* tendrá el siguiente formato:

FACT (Expresion)

Tomará como entrada una expresión y devolverá el número factorial de la misma.

La función especial *Combinatorio* tendrá el siguiente formato:

COMB (Expresión,Expresión)

Tomará como entrada dos expresiones y devolverá el número combinatorio de las mismas.

5. Array

Los vectores serán unidimensionales de límite fijo, de tipo entero (almacenarán enteros) con chequeo de límites cuyos componentes deberán operar como constantes numéricas comunes dentro de cualquier expresión y sus índices serán variables o constantes enteras.

Los elementos de estos vectores podrán obtener sus valores de la siguiente manera:

list[3] = {1, 3, 8}; (cuando se desee asignarle un valor a los primeros tres elementos del vector)
ó
list[3] = 5 (cuando se le asigna un valor a un solo elemento del vector)

Los vectores siempre comenzarán desde el componente 1.

PROGRAMA DE TESTEOS

El TP se entregará con un programa de testeos que permitirá, colocando los comentarios a cada sentencia, comprobar el correcto funcionamiento del programa.
El programa principal de testeos es el siguiente:

```
** pseudocodigo
** hacer este programa en su lenguaje y luego probar lo que está entre paréntesis

definicion a b c d e  int int string string int

a = 10
comentario a = 1000000000000000000000000000000000000 (cota: probar este error y luego
comentario)

b = 20

c = "cadena"
comentario c = "cadenaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" (cota: probar
este error y luego comentarlo)

d = "de prueba"
comentario d = 30 (tipo: probar este error y luego comentarlo)

e = 2

comentario f = 30 (no definida: probar este error y luego comentarlo)

comentario ESTO ES UN ESTO ES UN COMENTARIO ANIDADO COMENTARIO ANIDADO

si a < b y b == 20

  mientras a # 0

        mientras e > 1
             e = e - 1
       finmientras

       a = a - 1

  finmientras

  si a = 0
       salida "a es igual a cero"
  else
       salida "a es distinto de cero"
  fin si

fin si

c = c ++ d

salida c

read e

comentario prueba tema especial 1

(prueba simple de tema especial 1, si es posible como parte de la condicion)

comentario prueba tema especial 2

(prueba simple de tema especial 2, si es posible como parte de la condicion)
```

TABLA DE SIMBOLOS

La tabla de símbolos tiene la capacidad de guardar las variables y constantes con sus atributos.
Los atributos portan información necesaria para operar con constantes, variables .
Ejemplo

NOMBRE	TIPO	VALOR	LONGITUD	LIMITE	ALIAS
a1	Float	—			x
b1	Float	—			
_variable1	CteString	variable1	9		
_30.5	CteReal	30.5			
list	Array			3	

Tabla de símbolos