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: 22/04/2019

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: 13/05/2019

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
  - Temas Especiales

- 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: 17/06/2019

ATENCION: Cada grupo deberá designar un integrante para el envío de los correos durante todo el cuatrimestre.

TEMAS COMUNES

ITERACIONES
Implementación de ciclo *WHILE*

DECISIONES
Implementación de *IF*

ASIGNACIONES
Asignaciones simples *A:=B*

TIPO DE DATOS
Constantes numéricas

- reales (32 bits)
- enteras (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"
```

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.

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 1°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 implementarán como se muestra en el siguiente ejemplo:

Ejemplo:

```
DISPLAY "ewr" */ donde "ewr" debe ser una cte string /*
GET base      */ donde base es una variable /*
DISPLAY var1 */ donde var1 es una variable numérica 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 de la siguiente manera:

```
DEFVAR
            Tipo de Variable : Lista de Variables  (1)

ENDDEF
```

El bloque delimitado con DEFVAR – VAR puede contener varias líneas de declaración de la forma (1)
El bloque delimitado con DEFVAR – VAR no puede estar vacío
El Tipo de Variable debe ser único para cada línea de la forma (1)
Un tipo de variable podría estar en más de una línea de la forma (1)
La Lista de variables debe separarse por punto y coma y no puede estar vacía

Ejemplos de formato:

```
DEFVAR
            FLOAT : a1; b1
            STRING: variable1
            FLOAT : p1; p2; p3
            INT : a;b

ENDDEF
```

TEMAS ESPECIALES

1. **FOR**
   La estructura de la sentencia FOR tendrá el siguiente formato

   FOR Variable = Expresion TO Expresion [Step Cte]

   NEXT Variable

   La variable en la instrucción NEXT deberá ser la misma que la variable contadora de la cláusula FOR.
   Los [] indican que los pasos son opcionales. En caso de que no se indique nada, por default asumirá pasos de 1 en 1.

2. **Longitud**
Esta función calcula la longitud de una lista de constantes y/o variables. Devuelve un entero que puede ser usado en cualquier expresión. La lista puede no contener elementos.

variable1 = long ([a,b,c,e]) */ Arrojara 4 como resultado /*

IF long([30,60]) = 2 then ..... */ True /*

3. Ciclo Especial
La estructura de la sentencia será

WHILE
Variable IN [Lista de Expresiones]

DO
Sentencias

ENDWHILE

Lista de expresiones es una lista de expresiones separadas por comas.

4. **Fibonacci**
La función especial Fibonacci (sucesión de Fibonacci) tendrá el siguiente formato:

FIB (n)

Tomará como entrada un número natural "n", y devolverá el enésimo término de la sucesión, sabiendo que los dos primeros términos de la misma son 0 y 1, y cada uno de los demás es igual a la suma de los dos anteriores. El resultado deberá operar en cualquier expresión numérica.

5. Take
TAKE (Operador ; cte ; [lista de constantes]).

Esta función toma como entrada un operador de suma, resta, multiplicación o división entera, una constante entera y una lista de constantes, devolviendo el valor que resulta de aplicar el operador a los primeros “n” elementos. El valor de **n** quedará establecido en la componente ***cte***. El resultado de la función puede ser utilizado en otras expresiones dentro del lenguaje y admite listas vacías, en cuyo caso retorna el valor 0.
En todo momento deberá validarse la cantidad definida en **cte** con la cantidad de elementos de la lista.

Ej.

```
TAKE (*  ; 3 ;  [2 ; 12 ; 24 ; 48] )   */ Resultado: 576 /*
TAKE (+  ; 2 ;  [2 ; 12 ; 24 ; 48] )   */ Resultado: 14 /*
TAKE (-  ; 3 ;  [2 ; 12 ; 24 ; 48] )   */ Resultado: -34 /*
TAKE (/  ; 4 ;  [2 ; 12 ; 24 ; 48] )   */ Resultado: 0 /*
TAKE (+  ; 3 ;  [2 ; 12 ] )            */ Error /*
TAKE (/  ; 4 ;  [ ] )                  */ Resultado: 0 /*
```

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	TIPODATO	VALOR	LONGITUD
a1	Float	—	
b1	Int	—	
_variable1		variable1	9
_30.5		30.5	
_0b111		7	
_0xF3A1		62369	

Tabla de símbolos