

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: 12/02/2019

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: 19/02/2019

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 **Grupox.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: Grupox.exe como Grupox.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 (**Grupox.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: Grupox.bat como Grupox.b

En todos los casos el compilador **Grupox.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: 01/03/2019

TEMAS COMUNES

ITERACIONES

Implementación de ciclo *WHILE*

DECISIONES

Implementación de *IF*

ASIGNACIONES

Asignaciones simples $A:=B$

TIPO DE DATOS

Constantes numéricas

- reales (32 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"
```

Constantes en base binaria y hexadecimal

Constantes enteras que responderán al formato 0bdigitosbinarios o 0xdigitoshexadecimales. Estas constantes operarán dentro de expresiones como cualquier constante entera

Ejemplo:

```
a := 0b111 + 35 + 0xF3A1
```

(Resultado para a → 62411 (obtenido de sumar 7 + 35 + 62369))

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.

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 3°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:

```
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:

DIM < Lista de Variables> AS < Tipo de Dato >

La Lista de variables debe separarse por comas y delimitada con [], y pueden existir varias líneas DIM.

La lista de variables y la lista de tipos deben coincidir en cantidad de elementos.

Ejemplos de formato:

```
DIM [ a, b, beta ] AS [ integer, real, string]  
DIM [ c ] AS [ real]
```

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 que no se indique nada, por default asumirá pasos de 1 en 1.

2. Longitud

Esta función calcula la longitud de una lista

$\text{long}([a,b,c,e]) = 4$

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. EQUMAX y EQUMIN

EQUMAX(expresión ; [lista de variables o ctes])

EQUMIN(expresión ; [lista de variables o ctes])

La función será verdadera o falsa, si *expresión* es igual al máximo o mínimo de la lista de variables o constantes.

La lista debe estar separada por comas y delimitada por corchetes.

Esta función deberá operar únicamente en condiciones.

Ejemplos:

```
var1=120  
IF EQUMAX(10+101; [100, 111, var1] ) ## será falsa
```

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

Tabla de símbolos