COMPG2 Tarea 1: Diseñar un escáner para C

1. Introducción

En esta tarea, se le pide que **diseñe un escáner para lenguaje C simplificado** que transforme el **código C** en una **serie** correspondiente de **Tokens**.
Lenguaje de programación preferiblemente C/C++ y Java. Sin embargo, las **herramientas de escaneo** como **Lex/Flex** no son ¡permitido!.
¡**EL plagio y código generado como ChapGPT no sera tolerado**!

2. Los tokens del lenguaje C simplificado

Antes de diseñar su escáner, es necesario aprender **los tokens del lenguaje C simplificado**. Para el lenguaje C simplificado, debe tomar en cuenta los 4 tipos de tokens: Palabras Reservadas (Keywords), Símbolos especiales, INT_NUM e ID. Aquí están sus reglas de coincidencia escritas en expresiones regulares:

2.1 Palabras Reservadas (Keywords)

int {INT}	main { MAIN }	void { VOID }	break { BREAK }
do { DO }	else { ELSE }	if { IF }	while { WHILE }
return { RETURN }		scanf { READ }	printf { WRITE }

2.2 Símbolos Especiales

‘‘{” {LBRACE} ‘‘}” {RBRACE} ‘‘[” {LSQUARE} ‘‘]” {RSQUARE}
‘‘(” {LPAR} ‘‘)” {RPAR} ‘‘;” {SEMI} ‘‘+” {PLUS}
‘‘–” {MINUS} ‘‘∗” {MUL_OP} ‘‘/” {DIV_OP} ‘‘&” {AND_OP}
‘‘|” {OR_OP} ‘‘!” {NOT_OP} ‘‘=” {ASSIGN} ‘‘<” {LT}
‘‘>” {GT} ‘‘<<” {SHL_OP} “>>” {SHR_OP} ‘‘==” {EQ}
‘‘!=” {NOTEQ} ‘‘<=” {LTEQ} ‘‘>=” {GTEQ} ‘‘&&” {ANDAND}
‘‘||” {ORoR} ‘‘,” {COMMA} …

2.3 INT_NUM and ID

digit = [0–9]
letter = [a–z A–Z]
INT_NUM = [digit]+
ID = [letter | _]+[digit | letter |]∗

2.4 Ejemplo

Su programa puede tomar un archivo de código C como entrada y finalmente imprimir una serie de tokens en la salida estándar. Por ejemplo, si este es el archivo de código C de entrada:

```
int main() {
 int a;
 int b;
 a = b + 1;
 return 0;
}
```

Su escáner debería imprimir una serie de tokens como los siguientes:

```
Token: INT “int”
Token: MAIN “main”
Token: LPAR “(“
Token: RPAR “)”
Token: LBRACE “{“
Token: INT “int”
Token: ID “a”
Token: SEMI “;”
Token: INT “int”
Token: ID “b”
Token: SEMI “;”
Token: ID “a”
Token: ASSIGN “=”
Token: ID “b”
Token: PLUS “+”
Token: INT_ NUM “1”
Token: SEMI “;”
Token: RETURN “return”
Token: INT_NUM “0”
Token: SEMI “;”
Token: RBRACE “}”
```

3 Presentación y evaluación

3.1 Envío debe ser en el Brighspace no establecemos ninguna limitación en los lenguajes y herramientas de programación, debe enviar su código en un rar o zip para la calificación. Deben hacer un informe técnico (pdf).

3.2 Informe Técnico

Sólo nos importan tres preguntas de su informe:

- ¿Qué tipo de autómata diseñaron, AFD o AFND?
- ¿Cómo se diseñó el autómata escogido?
- ¿Cómo se diseña la función de escaneo?

No es necesario que el informe sea muy largo, pero el formato debe ser claro. Mientras las tres preguntas se responden claramente y el formato es correcto, su informe obtendrá la máxima puntuación.

3.3 Criterio de calificación

- Corrección del programa: 80%. Hemos preparado 10 casos de prueba y 5 de ellos se proporcionarán a usted desde el principio.
- Si su programa puede superar los 5 casos indicados, obtendrá 60 puntos.
- Si tu programa puede pasar el resto de los casos, obtendrás los 20 puntos restantes.
- Estilo de Código y Comentario: 10%.
- Si su código está limpio y tiene los comentarios necesarios, obtendrá 10 puntos.
- Informe Técnico: 10%. Si su informe responde claramente a las tres preguntas y el formato es limpio, obtendrás 10 puntos

Palabras Reservadas
auto double int struct
break else long switch
case enum register typedef
char extern return union
const float short unsigned
continue for signed void
default goto sizeof volatile
do if static while

Directivas de pre procesamiento
Define, elif, else, endif, error, if, ifdef, ifndef
Include, message, undef

Operadores Lógicos
&&, ||, !

Operadores Lógicos bits
&, |, ^, ~, >>, <<

Operadores Relacionales
>, >=, <, <=, ==, !=
Operadores Agrupación
[,], (,),{,}

Otros Operadores
. ->, “,”, **“”””**

También se tendrán en cuenta los caracteres de escape:
“\n”, “\0”, “\r”, “\t”, ”\\”, “\v”, “\f”, “\a”, “\””, “\’”