# Analizador Léxico de C creado con Flex

## Instituto Tecnológico de Costa Rica

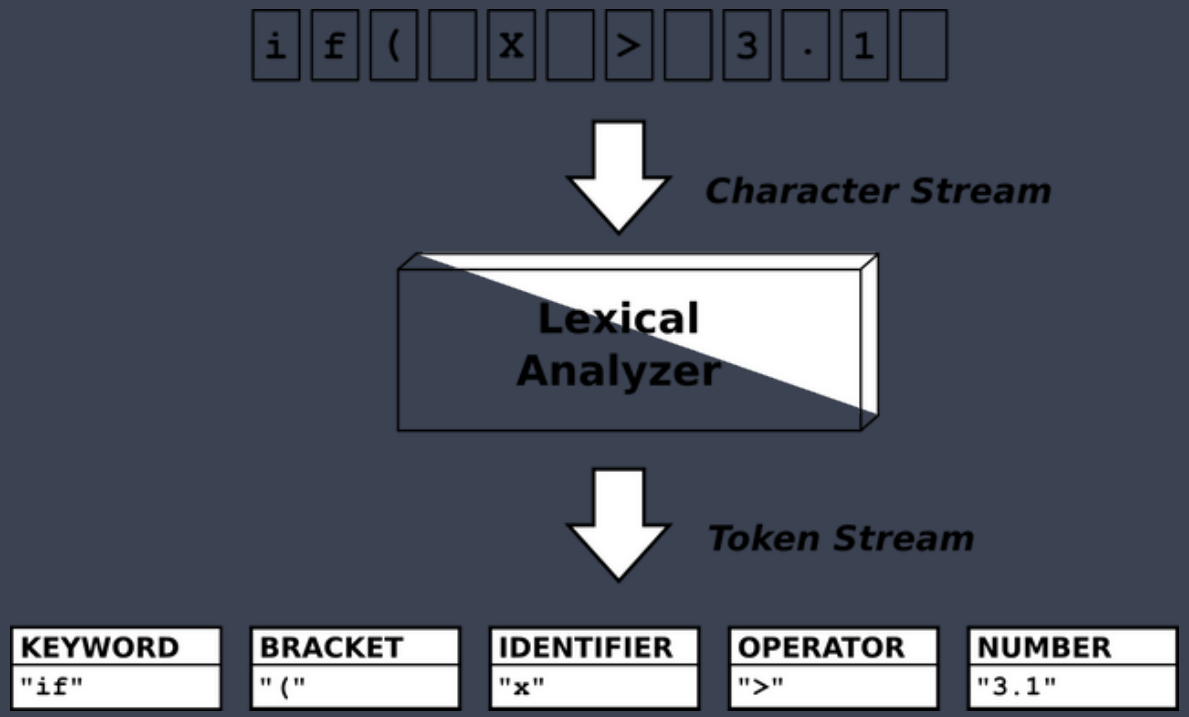
## Compiladores e Intérpretes
## Semestre II 2020

by Ronald Herrera Gámez

on November 25, 2020

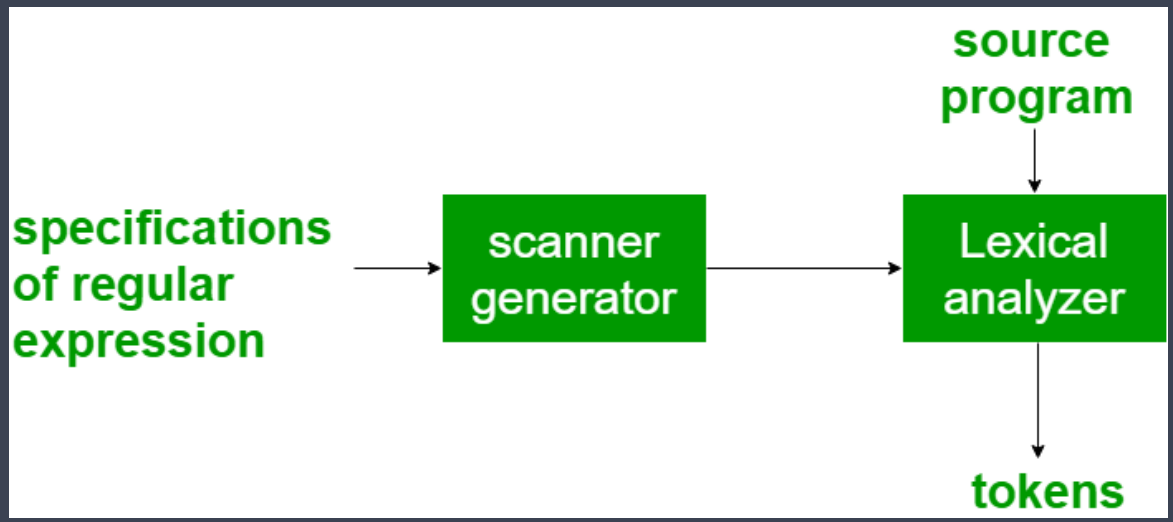
# » Proceso de Scanning

Consiste en determinar las diferentes unidades elementales de un programa fuente, es decir, identifica los distintos lexemas de un lenguaje. Para este proceso, el scanner busca patrones dentro del fuente que cumplan con instrucciones, operadores, identificadores, constantes, entre otros, del lenguaje que se escanea.



# » Flex (Fast Lexical Analyzer)

Flex es una herramienta para generar analizadores léxicos basados en la teoría de autómatas finitos. Para ello, se crea la descripción del escáner en forma de pares de expresiones regulares y código C, llamados reglas. Flex genera un archivo fuente en C llamado "lex.yy.c" que se puede compilar y vincular para producir un ejecutable. Este ejecutable analiza su entrada en busca de ocurrencias de texto que coinciden con las expresiones regulares para cada regla y siempre que encuentra una coincidencia, ejecuta el código C correspondiente.



# » Código Después Del Preproceso

## TOKENS:

KEYWORD
IDENTIFIER
LITERAL
OPERATOR
PUNTUACTOR
COMMENT
LEXICALERROR
PREPROCESSOR

```
typedef unsigned char bit ; struct Nodo * primero = NULL ; struct Persona { char * nombre ; int edad ; bit id ; } ; struct Nodo { struct Persona persona ; struct Nodo * sig ; } ; void agregarPersona ( const char * nombre , int edad , bit id ) { struct Nodo * nuevo = ( struct Nodo * ) malloc ( sizeof ( struct Nodo ) ) ; nuevo -> sig = NULL ; nuevo -> persona . nombre = ( char * ) malloc ( 1 ) ; strcpy ( nuevo -> persona . nombre , nombre ) ; nuevo -> persona . edad = edad ; nuevo -> persona . id = id ; if ( primero == NULL ) { primero = nuevo ; } else { nuevo -> sig = primero ; primero = nuevo ; } } unsigned char aislarBit ( int n , unsigned char ID ) { unsigned char bit = ID ; bit = bit « ( n -1 ) ; bit = bit » 7 ; return bit ; } void imprimir ( ) { puts ( "\nLista de personas:\n" ) ; struct Nodo * aux = primero ; while ( aux != NULL ) { printf ( "Nombre: %s, Edad: %i, ID: %i, Bit2: %d\n" , aux -> persona . nombre , aux -> persona . edad , aux -> persona . id , aislarBit ( 2 , aux -> persona . id ) ) ; aux = aux -> sig ; } } int main ( ) { agregarPersona ( "Ronald" , 21 , 12 ) ; agregarPersona ( "Rose" , 18 , 13 ) ; agregarPersona ( "Ashly" , 17 , 12 ) ; agregarPersona ( "Jack" , 14 , 18 ) ; agregarPersona ( "Chester" , 13 , 21 ) ; agregarPersona ( "Mike" , 12 , 32 ) ; agregarPersona ( "Jen" , 38 , 21 ) ; imprimir ( ) ; return 0 ; }
```

# » Histograma Tokens Usados