Programa 3: Generación de código

Código fuente: *matrícula*_c3.c / *matrícula*_c3.cpp

El programa consiste en la implementación de la etapa de generación de código de un compilador para el lenguaje de programación visto en clase.

La siguiente gramática en la notación Backus-Naur extendida reconoce un subconjunto sintáctico del lenguaje C y sus no terminales corresponden con los no terminales presentados en la especificación del programa 2.

```
PROGRAMA = INCLUSIONES FUNCIONES FIN_ARCHIVO

INCLUSIONES = ε
INCLUSIONES = “#” “include” LITERAL_CADENA “\n” INCLUSIONES

FUNCIONES = ε
FUNCIONES = FUNCION FUNCIONES

FUNCION = “int” IDENTIFICADOR “(“ PARAMETROS “)” “{“ INSTRUCCIONES “}”

PARAMETROS = ε
PARAMETROS = “int” IDENTIFICADOR PARAMETROS_CONT
PARAMETROS_CONT = ε
PARAMETROS_CONT = “,” “int” IDENTIFICADOR PARAMETROS_CONT

INSTRUCCIONES = ε
INSTRUCCIONES = INSTRUCCIÓN INSTRUCCIONES

INSTRUCCIÓN = SENTENCIA_DECLARACION
INSTRUCCIÓN = SENTENCIA_IF
INSTRUCCIÓN = SENTENCIA_PRINT
INSTRUCCIÓN = SENTENCIA_RETURN

SENTENCIA_DECLARACION = “int” IDENTIFICADOR “=” EXPRESIÓN “;”

SENTENCIA_IF = “if” “(“ EXPRESIÓN “)” “{“ INSTRUCCIONES “}” SENTENCIA_IF_CONT
SENTENCIA_IF_CONT = ε
SENTENCIA_IF_CONT = ELSE SENTENCIA_IF
SENTENCIA_IF_CONT = ELSE “{“ INSTRUCCIONES “}”

SENTENCIA_PRINT = “printf” “(“ “\“%d\”” “,” EXPRESIÓN “)” “;”
SENTENCIA_RETURN = “return” EXPRESIÓN “;”

EXPRESIÓN = EXPRESIÓN_UNARIA EXPRESIÓN_CONT
EXPRESIÓN_CONT = ε
EXPRESIÓN_CONT = OPERADOR_BINARIO EXPRESIÓN_UNARIA EXPRESIÓN_CONT

OPERADOR_BINARIO = MAS | MENOS | POR | ENTRE | MENOR_QUE | MENOR_IGUAL_QUE |
                   MAYOR_QUE | MAYOR_IGUAL_QUE | IGUAL_QUE | DIFERENTE_QUE

EXPRESIÓN_UNARIA = OPERADORES_PREFIJOS EXPRESIÓN_PRIMARIA OPERADORES_POSFIJOS

OPERADORES_PREFIJOS = ε
OPERADORES_PREFIJOS = OPERADOR_PREFIJO OPERADORES_PREFIJOS
```

```
OPERADOR_PREFIJO = MAS | MENOS
OPERADORES_POSFIJOS = ε
OPERADORES_POSFIJOS = "(" EXPRESIONES ")"

EXPRESIONES = ε
EXPRESIONES = EXPRESIÓN EXPRESIONES_CONT
EXPRESIONES_CONT = ε
EXPRESIONES_CONT = "," EXPRESIÓN EXPRESIONES_CONT

EXPRESIÓN_PRIMARIA = IDENTIFICADOR
EXPRESIÓN_PRIMARIA = LITERAL_ENTERA
EXPRESIÓN_PRIMARIA = "(" EXPRESIÓN ")"
```

El símbolo no terminal inicial es `PROGRAMA` y los no terminales no definidos anteriormente coinciden con lo descrito en la especificación léxica del lenguaje origen. La traducción del lenguaje origen al lenguaje destino puede realizarse mediante la traducción de los no terminales correspondientes.

Es responsabilidad del programa incluir stdio.h en caso de que se requiera la traducción de alguna `SENTENCIA_PRINT`.

Los operadores tienen la misma asociatividad y precedencia relativa que los operadores correspondientes del lenguaje C excepto que los operadores == y != tienen la misma precedencia que el resto de los operadores relacionales.

Entrada: Código fuente que deberá ser leído desde un archivo llamado `codigo.txt`. Puede suponer que el archivo existe y que el análisis léxico no generará errores.

Salida: Código fuente que sea la traducción del código de entrada al lenguaje C o el mensaje `ERROR` en caso de un error sintáctico en el código origen. Puede suponer que si el código es sintácticamente correcto entonces el código C correspondiente es semánticamente correcto.

Recuerde que C no siempre admite comas al final de una lista de expresiones.

Ejemplo de entrada	Ejemplo de salida

`function min(a, b,)`
`{`
`  if a < b {`
`  return a;`
`  }`
 	
`  return b;`
`}`
 	
`function main( )`
`{`
	`#include <stdio.h>`
	
	`int min(int a, int b)`
	`{`
	`  if (a < b) {`
	`  return a;`
	`  }`
	
	`  return b;`
	`}`

```
   var a = 5;
   print min(a, 2);

   return 0;
}
```

```
int main( )
{
   int a = 5;
   printf("%d", min(a, 2));

   return 0;
}
```

Su programa sólo debe imprimir lo solicitado. El código fuente deberá ser enviado como archivo adjunto al correo al203305906@alumnos.azc.uam.mx. No se recibirán ejecutables y de ninguna otra forma.