PROBLEMAS TEMA 3

GRUPO 4
Sara Martín Rodríguez
Marta Zhao Ladrón de Guevara Cano
Leandro Jorge Fernández Vega
Laura Salas López

1. Un procesador (CPU) puede interpretar y ejecutar directamente las instrucciones de un programa en:

(a) Lenguaje de alto nivel de tipo intérprete.
(b) Lenguaje ensamblador o en lenguaje máquina, cualquiera de los dos.
(c) Sólo lenguaje máquina.
(d) En pseudocódigo o en lenguaje ensamblador.

2. ¿Es lo mismo un token que un lexema? Muestre algún ejemplo.

No. Un lexema o palabra es una secuencia de caracteres con un significado propio (por ejemplo: if, else, <=, ==, var1,...) mientras que un token es el conjunto de dichos lexemas al que se le da un misión sintáctica dependiendo de la gramática del lenguaje fuente (por ejemplo: IF, ELSE, OP_COMP, IDENT...)

3. ¿El compilador es la única utilidad necesaria para generar un programa ejecutable en una computadora?

No, ya que requiere del enlazador para añadir las bibliotecas y otras referencias externas, y así generar los archivos objeto necesarios para el programa ejecutable. Sin embargo, si el programa principal solo depende de un módulo objeto (el suyo propio, sin referencias externas) solo será necesario el compilador.

4. El análisis léxico es una etapa de la compilación cuyo objetivo es:

(a) Extraer la estructura de cada sentencia, reconociendo los componentes léxicos (tokens) del lenguaje.
(b) Descomponer el programa fuente en sus componentes léxicos (tokens).
(c) Extraer el significado de las distintas construcciones sintácticas y elementos terminales.
(d) Sintetizar el programa objeto.

5. El análisis sintáctico es una etapa de la compilación cuyo objetivo es:

(a) Extraer la estructura de cada sentencia, reconociendo los componentes léxicos (tokens) del lenguaje.
(b) Descomponer el programa fuente en sus componentes léxicos (tokens).
(c) Extraer el significado de las distintas construcciones sintácticas y elementos terminales.
(d) Sintetizar el programa objeto.

6. Para el siguiente código que aparece a la izquierda en lenguaje C++ (fichero test.cpp), indique el nombre de la fase en la que el compilador produce el mensaje de error que aparece a la derecha y explique la naturaleza del mismo:

a) test.cpp:9: error: expected primary-expression before ';' token

Este error se produciría en la fase de análisis sintáctico ya que dicha secuencia de tokens no se les puede estructurar, es decir no hay reglas gramaticales aplicables y por tanto no se genera un árbol sintáctico que indique el órden en el que aplicar las producciones de la gramática.

b) test.cpp:6: error: invalid conversion from 'int' to 'char*'

Este error se produce en la fase de análisis semántico, porque al hacer una asignación de variables incompatibles se detecta como una construcción sin un significado concreto.

c) test.cpp:11: error: stray '\302' in program

Este error se produce en la fase de análisis léxico ya que al leer el carácter y agruparlo como lexema no existe ningún token asociado.

7. Muestre un ejemplo a partir de una sentencia en lenguaje C++ en la que un error léxico origine un error sintáctico derivado y otro error léxico que no derive en error sintáctico.

Error léxico que genera uno sintáctico:

```
inte edad;
edad+=1;
```

"inte" (debería ser "int") no se puede asociar a ningún token reconocido, por lo que es imposible declarar la variable. Al usar posteriormente una variable no declarada y operar con ella llegamos a un error sintáctico.

Error léxico que no genera uno sintáctico:

```
if (...){}
fi
…
```

En C++ la finalización de un condicional se lleva a cabo con la llave de cerrado ("}"). La palabra "fi" no se puede asociar a ningún token y no sería reconocida por el compilador. Sin embargo, no afecta a ninguna estructura del programa, por lo que no deriva en error sintáctico.

8. Muestre un ejemplo a partir una sentencia de en lenguaje C++ en la que un error léxico origine un error sintáctico y semántico derivados y otro error léxico que no los derive.

doeble n;
n++;

Como "doeble" está mal escrito (tendría que ser "double") se produce un error léxico que origina un error semántico derivado pues la variable "n" queda sin declarar y después estamos usándola. Además se produce un error sintáctico pues el operador ++ no se está usando correctamente.

int i = 0;
whole (i<10) {
…
}

Por estar "whole" mal escrito (debería ser "while") se produce un error léxico, pero este no deriva en ningún otro error, simplemente no va a reconocer el bucle while.

9. ¿Sería siempre posible realizar la depuración de un archivo objeto? Razone la respuesta.

No sería posible, pues al fin y al cabo la depuración consiste en ejecutar un programa en un ambiente controlado, en el que puedas por ejemplo conocer el valor de las variables en cada instante; es decir, que s ehace un archivo ejecutable resultado del enlazado y no sobre un archivo compilado.

10. Dado un programa escrito en lenguaje ensamblador de una arquitectura concreta, ¿sería directamente interpretable ese código por esa computadora? En caso contrario ¿qué habría que hacer?

No, el computador solo interpreta el lenguaje máquina (o lenguaje de bajo nivel). Por tanto, sería necesario traducirlo bien a través del compilador o del intérprete.

11. ¿Sería necesario usar siempre el enlazador para obtener un programa ejecutable?

Será necesario siempre que haya referencias externas necesarias para la correcta generación del programa ejecutable. Si se dispone de un único fichero de código sin referencias externas, se generará un solo archivo objeto (el suyo propio) y únicamente se hará uso del compilador.

12. Dado un único archivo objeto, ¿podría ser siempre un programa ejecutable y correcto simplemente añadiendo la información de cabecera necesaria?

Los archivos objeto son resultado de la compilación mientras que los ejecutable son producto del enlazado. Los archivos ejecutables cuentan en la cabecera con el punto de inicio del mismo. Además, en cuanto a las regiones, sólo hay información de reubicación si ésta se ha de realizar en la carga.
Por tanto, no basta con añadir solo la información de cabecera, ya que podría darse la existencia de referencias externas sin definir.

13. Dado un programa ejecutable que requiere de una biblioteca dinámica, ¿por qué no es necesario recompilar el código fuente de dicho programa si se modifica la biblioteca?

Porque las bibliotecas dinámicas se cargan en tiempo de ejecución cuando son requeridas por el programa. Como el enlazado se produce en tiempo de ejecución y no de compilación no hace falta recompilar. De hecho, esa es una de las grandes ventajas de las bibliotecas dinámicas con respecto a las bibliotecas estáticas, y es que las estáticas tienen que recompilar todos los programas que dependen de ellas, en caso de modificación.

14. Indique en qué fase del proceso de traducción y ejecución de un programa se realizará cada una de las siguientes tareas:
(a) Enlazar una biblioteca estática.
Se encuentra ya en fase de ejecución en el enlazado.
(b) Eliminar los comentarios del código fuente.
Se encuentra en el proceso de traducción durante durante la fase de análisis concretamente durante el analizador léxico
(c) Mensaje de error de que una variable no ha sido declarada.
Se encuentra en el proceso de traducción durante durante la fase de análisis concretamente durante el analizador semántico
(d) Enlazar una biblioteca dinámica.
Se encuentra en el proceso de ejecución durante la carga y ejecución del programa.

15. Indique en qué fase o fases del proceso de compilación de un lenguaje de programación de alto nivel se detectarían los siguientes errores:
(a) Una variable no está definida.
Análisis semántico, ya que una variable no definida carece de significado para el compilador.
(b) Aparece un carácter o símbolo no esperado.
Análisis léxico, ya que no se corresponde con ningún token.
(c) Aparecen dos identificadores consecutivos.
Análisis sintáctico, ya que dos identificadores seguidos no se corresponden con ninguna estructura.
(d) Aparecen dos funciones denominadas bajo el mismo nombre.
Análisis semántico, aunque únicamente cuando no sea posible la sobrecarga o no se haga de manera adecuada.

(e) Aparece el final de un bloque de sentencias pero no el inicio del mismo.
Análisis sintáctico, ya que supone un bloque de instrucciones incompleto que viola las normas gramaticales del lenguaje.
(f) Aparece un paréntesis cerrado y no se ha podido emparejar con su correspondiente paréntesis abierto.
Análisis sintáctico, ya que el compilador no puede encontrar el que le corresponde y no puede reconocer correctamente la estructura de la orden.
(g) Una llamada a una función que no ha sido definida.
Análisis semántico, ya que una función no declarada carece de sentido para el compilador.
(h) En la palabra reservada main aparece un carácter extraño no esperado, por ejemplo mai¿n.
Análisis léxico, ya que no se puede asociar a ningún token.

16. ¿Todo error sintáctico origina un error semántico? En caso contrario, demuéstrelo usando algún contraejemplo.

No, un posible contraejemplo sería:

```
int n = 0;
n += ;
```

Hay un error sintáctico porque falta un dato a la derecha de +=, pero no sería un error semántico pues "n" está declarada y es correcto escribir += después de una variable declarada.