PREGUNTAS TEST COMPILADORES - TEMA 1 (2° Grado en Informática)

FACULTAD DE INFORMÁTICA

1. La **sintaxis de los lenguajes de programación**:
   a) Se analiza con *autómatas de pila*, puesto que suelen ser generados por *gramáticas libres de contexto*.
   b) Se analiza con *autómatas linealmente acotados*, puesto que suelen ser generados por *gramáticas sensibles al contexto*.
   c) Se analiza con *autómatas de pila*, aunque suelen ser *sensibles al contexto*.
2. Realizar un compilador...
   a) ...que siempre garantice la *generación del código más óptimo es posible*, aunque generalmente no merece la pena.
   b) ...que siempre garantice la *generación del código más óptimo es imposible*, puesto que su realización entra dentro de los llamados *problemas NP-completos*.
   c) ...que *genere código óptimo* es imposible, puesto que aún no se conocen las técnicas de optimización necesarias.
3. Una **máquina virtual**:
   a) Es un programa que sirve para que funcionen algunos compiladores que generan código intermedio, con el fin de aumentar la velocidad en el proceso de traducción.
   b) Es un intérprete de un lenguaje de bajo nivel que funciona, salvo en la velocidad, como una máquina real para dicho lenguaje.
   c) Se usa en aquellos lenguajes que jamás podrían ser analizados por una máquina real, funcionando de forma semejante a dicha máquina.
4. Elige, entre las siguientes, la frase que consideres más adecuada para definir el proceso de **arranque**:
   a) Se produce cuando en un procesador de lenguaje coinciden el *lenguaje de implementación* y el *lenguaje fuente*.
   b) Se produce cuando en un procesador de lenguaje coinciden el *lenguaje de implementación* y el *lenguaje destino*.
   c) Se produce cuando en un procesador de lenguaje el *lenguaje destino* es un *lenguaje intermedio*.
5. Un **compilador portable**:
   a) Se consigue escribiéndolo en un lenguaje de alto nivel y compilándolo cada vez que se quiera cambiar de máquina.
   b) Puede conseguirse implementando un intérprete para un lenguaje intermedio adecuado.
   c) No puede existir, puesto que al cambiar de máquina seguirá generando código para la máquina original.
6. Las técnicas de **recuperación de errores** en el proceso de traducción:
   a) Tienen como objetivo obtener un código objeto sin los errores introducidos por el usuario.
   b) Pretenden exclusivamente informar al usuario de los errores de la forma más precisa posible.
   c) Intentan localizar el máximo posible de errores.
7. La **tabla de símbolos**:
   a) Almacena información útil acerca de los identificadores de un lenguaje de programación, siendo útil exclusivamente en la etapa de análisis del proceso de compilación.
   b) Es una estructura de datos que se usa a lo largo de todo el proceso de compilación y que sirve, entre otras cosas, para poder controlar las restricciones contextuales de un lenguaje de programación.
   c) Es una estructura de datos donde se almacena el código intermedio generado por el compilador.

8. Las **restricciones contextuales de los lenguajes de programación**:

   a) Suelen formar parte de la *semántica* de dichos lenguajes, en la que se usa la *tabla de símbolos*.

   b) Se ignoran hasta que no se lleva a cabo la fase *optimización dependiente de la máquina*.

   c) Suelen resolverse implementando *autómatas linealmente acotados* que reconocen *lenguajes sensibles al contexto*.

9. El uso de un **intérprete** es adecuado cuando:

   a) Se espera que el programa se ejecute frecuentemente.

   b) Se espera que cada instrucción se ejecute una vez.

   c) Las instrucciones tienen formatos complicados.

10. El **arranque pleno**:

    a) Se basa en el uso de un compilador portable.

    b) Se basa en el uso de un compilador cruzado.

    c) Facilita el mantenimiento de un compilador, puesto que se consigue que el lenguaje de implementación coincida con el lenguaje fuente, aunque implica la escritura completa de dicho compilador.

11. Elegir, entre los siguientes, el tipo de *máquina abstracta* más adecuado para realizar el **análisis sintáctico** de un lenguaje de programación:

    a) *Autómata Finito*, pues siempre necesitaremos analizar las palabras del lenguaje.

    b) *Autómata de Pila*, a pesar de que la mayoría de los lenguajes de programación no son *libres de contexto* sino *sensibles al contexto*.

    c) *Autómata Linealmente Acotado*, puesto que la mayoría de los lenguajes de programación tienen restricciones contextuales (por ejemplo, la declaración de variables).

12. Un **compilador interpretado** consiste en:

    a) Un compilador que da la opción de generar *código máquina* o *código intermedio* dependiendo de las necesidades del usuario.

    b) Un compilador que genera código escrito en *lenguaje de alto nivel* que posteriormente es interpretado por algún intérprete adecuado existente en el sistema.

    c) Un compilador que genera *código intermedio* que posteriormente es interpretado por una **máquina virtual**.

13. Con respecto al **código generado por un compilador**:

    a) Podemos implementarlo de manera que exista garantía de que siempre genera *código óptimo*, aunque generalmente no merece la pena.

    b) Nunca podremos implementar un compilador que genere código óptimo en ninguna situación, puesto que se trata de un problema *NP-completo*.

    c) Nunca podremos implementar un compilador que garantice la generación de código óptimo en todos los casos.

14. En el proceso de **arranque** debe darse la circunstancia de que:

    a) Coinciden el *lenguaje de implementación* y el *lenguaje destino*.

    b) Coinciden el *lenguaje de implementación* y el *lenguaje fuente*.

    c) Coinciden el *lenguaje de implementación* con el de una *máquina virtual* existente en el sistema.

15. Un **preprocesador** es:

    a) Una herramienta que toma como entrada varios ficheros escritos en lenguaje de alto nivel y permite traducirlos a código máquina.

    b) Un traductor cuyo *lenguaje fuente* está constituido por una serie de macros y su *lenguaje destino* es una forma extendida de algún lenguaje de alto nivel.

    c) Un traductor cuyo *lenguaje fuente* es una forma extendida de un lenguaje de alto nivel y su *lenguaje destino* es la forma estándar del mismo lenguaje.

16. Señalar la razón por la que se considera qua las **técnicas de recuperación de errores** son importantes:

    a) Porque permiten obtener un código objeto sin los errores introducidos por el usuario.

    b) Porque permiten informar al usuario de una forma más precisa.

    c) Porque posibilitan la detección de más de un error.

17. La **tabla de símbolos**:

    a) Es una estructura de datos útil para analizar la *semántica* de un lenguaje de programación, aunque en ningún caso se usa en la *comprobación de tipos*.

    b) Es una estructura de datos que resulta útil para el almacenamiento de las variables y sus atributos, de manera que suele pasar información de las *declaraciones* a los *usos*.

    c) Es una estructura de datos que sirve para almacenar información semántica, de forma que dicha información se suele añadir a lo largo de la *fase de síntesis* para usarla posteriormente en la *fase de análisis*.

18. Podemos asegurar que un lenguaje de programación en el que se requiera la declaración de variables previa a su uso es:

    a) *libre de contexto*.

    b) *sensible al contexto*.

    c) de ambos tipos.

19. Una *máquina abstracta*:

    a) traduce el código intermedio a código máquina, que posteriormente será ejecutado.

    b) es un intérprete para un lenguaje de alto nivel.

    c) puede considerarse como la implementación software de una máquina.

20. Podemos afirmar que un *compilador interpretado*:

    a) tarda más tiempo en generar la salida que un compilador normal.

    b) genera una salida que se ejecutará a más velocidad que la generada por un compilador normal.

    c) genera una salida más portable que la generada por un compilador normal.

21. La tabla de símbolos:

    a) no tiene nada que ver con el análisis semántico.

    b) sólo sirve para comprobar si un identificador se declaró antes de usarlo.

    c) puede usarse, entre otras cosas, para para almacenar los tipos de cada variable.

22. Una gramática cuyas producciones tienen la forma:
    $\alpha A \beta \rightarrow \alpha \gamma \beta$, $A \in V_N$, $\alpha\beta \in (V_N \cup V_T)^*$, $\gamma \in (V_N \cup V_T)^+$

    a) es una gramática libre de contexto.

    b) es una gramática regular.

    c) es una gramática dependiente del contexto.

23. Gracias a la teoría de *lenguajes formales*:

    a) El análisis léxico y sintáctico pueden realizarse de forma sistemática, puesto que consisten en la simulación de un autómata de pila y un autómata linealmente acotado, respectivamente.

    b) El análisis semántico puede realizarse de forma sistemática, pues todos los lenguajes de programación son sensibles al contexto y, por tanto, dicho análisis consiste exclusivamente en la simulación de un autómata finito.

    c) La primeras fases del proceso de análisis de un compilador se pueden realizar de forma eficiente y sistemática, pues suelen consistir en la simulación de autómatas deterministas.

24. El lenguaje de implementación de un compilador:

    a) Debe ser de bajo nivel, para que pueda funcionar en una máquina dada.

    b) En ningún caso puede ser el propio lenguaje fuente del compilador, puesto que no podríamos compilarlo de ninguna forma.

    c) Suele ser un lenguaje de alto nivel.

25. Un compilador:

    a) No se puede escribir en lenguaje de alto nivel, puesto que habría que compilarlo.

    b) No se puede escribir en su propio lenguaje fuente, puesto que no podría compilarse.

    c) Podría estar escrito en un lenguaje intermedio y ejecutarse sobre una máquina virtual.

26. Un compilador cruzado:

    a) Es un traductor en el que el lenguaje fuente y el lenguaje de implementación son distintos.

    b) Es un traductor capaz de generar código ejecutable para una plataforma distinta de aquella en la que se está ejecutando él mismo.

    c) Es un compilador en el que el lenguaje de implementación y el lenguaje destino coinciden.

27. La gramática $G = (\{i, a, b\}, \{S, C\}, S, P)$, con el conjunto de reglas P siguiente:

$$
\begin{aligned}
S &\rightarrow i\ C\ i \\
&\quad \mid i\ i \\
i\ C\ i\ &\rightarrow i\ a\ i \\
&\quad \mid i\ b\ i
\end{aligned}
$$

    a) Es una gramática regular para representar cadenas que comienzan y acaban por el símbolo i, aunque podria especificarse de forma más simple usando expresiones regulares.

    b) Es sensible al contexto y podría representar algunos aspectos sensibles al contexto de un lenguaje de programación, por ejemplo, la necesidad de declarar los identificadores.

    c) Está mal construida.

28. El preprocesador de C:

    a) Es un traductor.

    b) Es un intérprete.

    c) No es ni un traductor ni un intérprete.

29. El analizador léxico simula:

    a) Un Autómata Linealmente Acotado.

    b) Un Autómata de Pila.

    c) Un Autómata Finito.

30. El preprocesador y el ensamblador:

    a) Son lenguajes necesarios para que se realice correctamente el proceso de traducción en algunos lenguajes, como C.

    b) Son traductores, que generalmente complementan al compilador.

    c) Son intérpretes.

31. El primer compilador:

    a) Se implementó en los años 30, y traducía BASIC.

    b) Se implementó en lenguaje de alto nivel.

    c) Necesitó 18 años/persona para ser implementado.

32. En el contexto de la jerarquía de Chomsky:

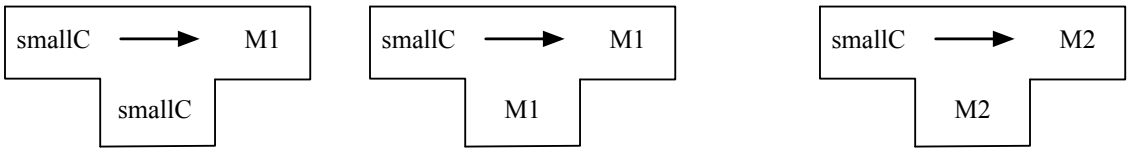
    a) No existen gramáticas para reconocer las restricciones de contexto, como el uso correcto de parámetros en una función. Por eso se usan las gramáticas atribuidas.

    b) Existen gramáticas para reconocer las restricciones de contexto, como el uso correcto de parámetros en una función. Sin embargo, se usan las gramáticas atribuidas.

    c) Usamos las gramáticas sensibles al contexto para reconocer las restricciones de contexto, como el uso correcto de parámetros en una función.

PREGUNTAS CORTAS.

1. *(0.5 puntos)* Se dispone de un compilador del lenguaje fuente *smallC* para la máquina *M1*. Dicho compilador ha sido escrito en el lenguaje de implementación *smallC* (configuración de partida). Se pretende crear una versión del compilador de *smallC* para la máquina *M2* (configuración objetivo). Para hacerlo no se dispone de ningún otro compilador distinto al indicado en la configuración de partida. ¿Cuál es el proceso más adecuado para llegar a la configuración objetivo? Indica las etapas de dicho proceso.



a) Configuración de partida

b) Configuración objetivo