Práctica 2

Ejercicio 1: ¿Qué define la semántica?

Define el significado de los símbolos, palabras y frases de un lenguaje ya sea lenguaje natural o lenguaje informático que es sintácticamente válido

Ejercicio 2:

a. **¿Qué significa compilar un programa?**
El compilador es un programa que traduce nuestro programa previo a la ejecución. Prepara el programa en el lenguaje máquina para que luego pueda ser ejecutado. El compilador toma todo el programa escrito en un lenguaje de alto nivel que llamamos lenguaje fuente antes de su ejecución. Luego de la compilación va a generar un lenguaje objeto que es generalmente el ejecutable (escrito en lenguaje de máquina .exe) o un lenguaje de nivel intermedio (o lenguaje ensamblador .obj).

b. **Describa brevemente cada uno de los pasos necesarios para compilar un programa.**
Los compiladores pueden ejecutarse en un solo paso o en dos. Dichos pasos cumplen con varias etapas, principalmente se pueden dividir en dos:

- Análisis
    - Análisis léxico (Scanner): Hay un análisis a nivel de palabra (LEXEMA).
        - Se fija que cada elemento del programa se corresponda con un operador, un identificador, etc., básicamente divide el programa en sus elementos: identificadores, delimitadores, símbolos especiales, números, palabras clave, palabras reservadas, comentarios, etc. De esta forma ve si todas las palabras son válidas o no.
        - Es el que lleva más tiempo.
        - Filtra comentarios y separadores como: espacios en blanco, tabulaciones, etc.
        - Genera errores si la entrada no coincide con ninguna categoría léxica.
        - Con esto descubrimos los tokens
        - Pone los identificadores en la tabla de símbolos.
        - Reemplaza cada símbolo por su entrada en la tabla de símbolos
        - El resultado de este paso será el descubrimiento de los items léxicos o tokens.

ID	Componente léxico	Lexema	Patrón
201	IF	if	if
202	OP_RELACIONAL	<, >, !=, ==	<, >, !=, ==
203	IDENTIFICADOR	var, s1, suma, prom	letra (letra | dígito | gb)*
204	ENTERO	2, 23, 5124	(dígito)+

    - Análisis semántico (Parser):

        - Se realiza luego de análisis léxico.
        - Se analiza a nivel de cada sentencia.
        - Busca estructuras, sentencia, declaración, expresiones, variable, ayudándose de los tokens.
        - Se alterna el analizador sintáctico con el análisis semántico.
        - Construye el árbol sintáctico del programa.
    - Análisis semántico (semántica estática):
        - Es la fase más importante.
        - Todas las estructuras sintácticas reconocidas son analizadas.
        - Se realiza una comprobación de tipos y se agrega información implícita (variables no declaradas).
        - Se agregan tablas de símbolos de los descriptores de tipos, etc.
        - Se hace comprobaciones de nombres y duplicados.
        - Nexo entre etapas inicial y final del compilador

Luego de esta etapa se puede realizarla la transformación del código fuente en una representación de código intermedio. Esta representación intermedia llamada código intermedio se parece al código objeto, pero sigue siendo independiente de la máquina. El código objeto es dependiente de la máquina. Este código intermedio debe ser fácil de producir y debe ser fácil de traducir al programa objeto. Hay varias técnicas, la más común es el código de tres dimensiones

- Síntesis: Se construye el programa ejecutable, genera el código necesario y se optimiza el programa generado. Si hay traducción separada de módulos se enlazan los distintos módulos objeto del programa (módulos, unidades, librerías, procedimientos, funciones, subrutinas, macros, etc.) mediante el linkeditor
    - Se genera el módulo de carga. Programa objeto completo.
    - Se realiza el proceso de optimización. (Optativo)

c. ¿En qué paso interviene la semántica y cuál es su importancia dentro de la compilación?
La semántica interviene en el análisis y es de sumamente importancia ya que es el nexo entre las etapas inicial y final del compilador

Ejercicio 3: Con respecto al punto anterior ¿es lo mismo compilar un programa que interpretarlo? Justifique su respuesta mostrando las diferencias básicas, ventajas y desventajas de cada uno.

No, no es lo mismo.

El intérprete y el compilador se puede comparar de diferentes formas

- En como ejecuta
    - Interprete: Lo hace durante la ejecución. Ejecuta sentencia a sentencia, lo traduce y lo ejecuta. Pasa un montón de veces. Para ser ejecutado en otra máquina se necesita tener si o si el intérprete instalado.
    - Compilador: Ocurre antes de ejecutar. Compila todo y genera un código objeto de un lenguaje de un modelo más bajo. El programa fuente no será publico
- Orden que lo ejecuta
    - Interprete: Sigue el orden lógico de ejecución (ya que va sentencia por sentencia del código)
    - Compilador: Sigue el orden físico de las sentencias.
- Tiempo de ejecución

    - Interprete: Por cada sentencia se realiza el proceso de decodificación para determinas las operaciones a ejecutar y sus operandos. Si la sentencia está en un proceso iterativo, se realiza la tarea tantas veces como sea requerido. Puede afectar la velocidad de proceso.
    - Compilador: Genera código de máquina para cada sentencia. No repite lazos, se decodifica una sola vez.
- Eficiencia
    - Interprete: Mas lento en ejecución. Se repite el proceso cada vez que se ejecuta el programa.
    - Compilador: Mas rápido desde el punto de vista del hardware, pero tarda más en compilar. Detectó más errores al pasar por todas las sentencias. Está listo para ser ejecutado. Ya compilado es más eficiente.
- Espacio ocupado
    - Interprete: Ocupa menos espacio de memoria ya que cada sentencia se deja en la forma original y las instrucciones necesarias para ejecutarlas se almacenan en los subprogramas del interprete en memoria.
    - Compilador: Una sentencia puede ocupar decenas o centenas de sentencias de máquina al pasar a código objeto
- Detección de errores
    - Interprete: Las sentencias del código fuente puede ser relacionadas directamente con la que se está ejecutando. Se puede ubicar el error, es más fácil detectarlos por donde pasa la ejecución y es más fácil corregirlos
    - Compilador: Le cuesta más determinar los errores ya que cualquier referencia al código fuente se pierden en el código objeto. Se pierde la referencia entre el código fuente y el código objeto. Es casi imposible ubicar el error, pobres en significado para el programador.

Ejercicio 4: Explique claramente la diferencia entre un error sintáctico y uno semántico. Ejemplifique cada caso.

Un error sintáctico ocurre cuando el programador escribe código que no va de acuerdo con las reglas de escritura del lenguaje de programación. El formato puede estar especificado en documentos BNF/EBNF.

Ejemplo: String s = “abc”;

Un error semántico ocurre cuando, si bien la sintaxis es correcta, pero la semántica o significado no es el que se pretendía. Hay errores semánticos que se detectan en compilación (semántica estática) y otros durante la ejecución (semántica dinámica).

Ejemplo: String s; s = 9;