Práctica 02

DOCENTE	CARRERA	CURSO
MSc. Vicente Enrique Machaca Arceda	Escuela Profesional de Ingeniería de Software	Fundamentos de Lenguajes de Programación

PRÁCTICA	TEMA	DURACIÓN
02	Lenguajes de programación	3 horas

1. Datos de los estudiantes

- Grupo: 03
- Integrantes:
  - Andree Alvarez Guzman
  - Jonathan Aguirre Soto

2. Ejercicios

Lea el libro "Programming Language Pragmatics" Scott (2000), Capitulo 1, Secci on 1.1 - 1.6 y responda las siguientes preguntas:

1. ¿Cuál es la diferencia entre lenguaje de maquina y lenguaje ensamblador?
   - El lenguaje de máquina es la secuencia de bits que controla directamente un procesador, es el lenguaje de programación de nivel más bajo donde las instrucciones se ejecutan directamente por la CPU. Mientras que el lenguaje ensamblador requiere un ensamblador para convertirlo en código de máquina / código de objeto, además el lenguaje ensamblador es comprensible para los humanos.
2. ¿En qué circunstancia un lenguaje de alto nivel es superior al lenguaje ensamblador?
   - Es superior a la hora de que el humano entienda lo que hace el código, ya que se hace muy tedioso y complicado el tener que convertir o transformar el lenguaje máquina o ensamblador a lenguaje humano, por eso surgieron varios lenguajes como Fortran que son de alto nivel, que es más fácil de interpretar para una persona.
3. ¿En qué circunstancia un lenguaje ensamblador es superior al lenguaje de alto nivel?
   - A la hora de procesar información, es decir, la máquina sólo reconoce y responde a un lenguaje binario, por ende el lenguaje ensamblador es de alto nivel, entonces es necesario que pase por un proceso de ensamblado a código máquina, ya que la computadora no comprenderá un lenguaje de programación de alto nivel.
4. ¿Por qué hay tantos lenguajes de programación?

- Porque fueron creados para diferentes propósitos o fines según iba avanzando la historia. Es decir, están los que nos sirven para resolver problemas generales de programación como C++, C, Java entre otros. También los orientados a trabajar sobre páginas web o a crearlas como PHP, HTML, JavaScript, etc. Otros en cambio están dedicados a las Bases de Datos, como DDL,DML, etc. Por ende hay demasiados lenguajes en la actualidad pero cada uno se desempeña en un cierto ámbito que a nosotros nos favorece ya que tenemos la potestad de poder decidir cuál nos conviene.

5. Nombre 3 lenguajes en las categorias de von Neumann, funcional y orientado a objetos. Dos lenguajes lógicos y 2 concurrentes.

   - Von Neumann: Fortran, Ada 83 y C.
   - Funcional: Lisp, ML y Haskell
   - POO: Smalltalk, Eiffel y Java
   - Lógicos: Prolog y spreadsheets
   - Concurrentes: Pascal y C++

6. ¿Qué distingue a los lenguajes declarativos e imperativos?

   - En el lenguaje imperativo hay una serie de pasos que nos llevan hacia un resultado; es de decir, podemos como cada uno de estos pasos es ejecutado en el código. En cambio, en el lenguaje declarativo, solo ponemos el resultado que deseamos obtener.

7. ¿Cuál es considerado el primer lenguaje de alto nivel?

   - El primer lenguaje de alto nivel fue Fortran ya que tiene una mejor construcción en arquitecturas de instrucciones hacia el computador y es amigable, además fue creado por John Backus de IBM en 1954.

8. ¿Cuál es considerado el primer lenguaje funcional?

   - Lisp es considero el primer lenguaje funcional de alto nivel creado en 1958 por John McCarthy. En su época fue un lenguaje revolucionario que introdujo nuevos conceptos de programación no existentes entonces: funciones como objetos primitivos, funciones de orden superior, polimorfismo, listas, recursión, símbolos, homogeneidad de datos y programas, bucle REPL (Read-Eval-Print Loop).

9. ¿Por qué los lenguajes concurrentes no estan considerados en la clasificación de Scott (2000) (Figura 1.1).?

   - En particular, es posible que un lenguaje funcional esté orientado a objetos y muchos autores no consideran que la programación funcional sea declarativa.

10. Lista las principales fases de un compilador y describe la función de cada fase.

    - Análisis Léxico: la primera fase de escáner funciona como un texto escáner. Esta fase busca en el código fuente como una secuencia de caracteres y la convierte en un lexema resultante.
    - Análisis Sintáctico: toma el token de análisis léxico como entrada y genera un árbol de sintaxis. En esta fase, los tokens se comprueban con la gramática, es decir, el analizador comprueba si la expresión de los tokens es sintácticamente correcto.
    - Análisis Semántico: comprueba si el árbol de sintaxis construido sigue las reglas del idioma.

- Generación de código intermedio: el compilador genera un código intermedio entre el código fuente y el el código de la maquina objetivo (binario).
- Optimización: realiza una optimización del código intermedio generado. La optimización puede asumirse como algo que quita lineas de código innecesarias, y ordena una secuencia de declaraciones que aceleran la ejecución del programa sin desperdiciar recursos de CPU o memoria RAM.
- Generación de código: se toma la versión optimizada del código intermedio y se comprueba con el lenguaje de máquina objetivo. La generación de código traduce entonces el código intermedio en una secuencia de código de máquina, luego es enlazado al sistema operativo donde tomará instrucciones y le asignará un espacio en la memoria para así poder hacerlo funcionar.
- Tabla de símbolos: en esta fase el compilador se jacta de esta estructura de datos para almacenar identificadores, tipos y valores de las variables a lo largo de todo el proceso de compilación, de forma que sea fácil y rápido obtener cualquiera de las propiedades de un símbolo.

11. ¿En qué circunstancias tiene sentido que un compilador pase o revise el código varias veces?
    - Necesitan leer el código fuente varias veces antes de poder producir el código máquina; es decir, tiene que examinar que no haya ningún error antes de compilarlo.
12. ¿Cuál es el próposito de la tabla de símbolos en un compilador?
    - El próposito de la tabla de símbolos es colaborar en las comprobaciones semánticas y facilitar la generación de código.
13. ¿En la actualidad, qué programa es más eficiente, uno desarrollado desde cero en ensamblador o uno generado por un compilador?
    - En la actualidad un programa es más eficiente sí es generado por un compilador debido a que incorpora técnicas que nos permiten mejorar el rendimiento del código a generarse.