Universidad La Salle
Ingeniería de Sofware

Fundamentos de Lenguajes de Programación

Karlo Pacha, Susana Mancilla
Fecha: Agosto 23, 2022

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

1. ¿Cuál es la diferencia entre lenguaje de maquina y lenguaje ensamblador?
   (a) El lenguaje de máquina se ejecuta de forma directa; el lenguaje de ensamblador requiere conversión a código de máquina para ser ejecutado.

2. ¿En qué circunstancia un lenguaje de alto nivel es superior al lenguaje ensamblador?
   (a) Para proyectos grandes, se hace uso de menos instrucciones.

3. ¿En qué circunstancia un lenguaje ensamblador es superior al lenguaje de alto nivel?
   (a) Cuando se necesite usar o manipular de manera más directa el hardware.

4. ¿Por qué hay tantos lenguajes de programación?
   (a) Por la evolución.
   (b) Prósitos especiales.
   (c) Preferencia personal.
   (d) Expresividad.
   (e) Facilidad de uso para los novatos.
   (f) Facilidad de implementación.
   (g) Fuente abierta.
   (h) Excelentes compiladores.

(i) Economía, mecenazgo e inercia.

5. Nombre 3 lenguajes en las categorias de von Neumann, funcional y orientado a objetos. Dos lenguajes lógicos y 2 concurrentes.
   (a) C, Ada, Fortran.
   (b) Haskell, ML, Lisp.
   (c) Java, Smalltalk, C++.
   (d) Prolog, Spreadsheets.
   (e) Ada, Modula-3.

6. ¿Qué distingue a los lenguajes declarativos e imperativos?
   (a) Su filosofía, uno sigue la secuencia de pasos indicada y otro va directamente al resultado final esperado.

7. ¿Cuál es considerado el primer lenguaje de alto nivel?
   (a) Fortran.

8. ¿Cuál es considerado el primer lenguaje funcional?
   (a) Lisp.

9. ¿Por qué los lenguajes concurrentes no estan considerados en la clasificación de Scott (2000) (Figura 1.1).?
   (a) Debido a que la distinción entre ejecución concurrente y secuencial es ortogonal a las clasificaciones ya existentes. Los lenguajes "concurrentes" no lo son explicitamente.

10. Lista las principales fases de un compilador y describe la función de cada fase.
    (a) Análisis léxico y sintáctico: Reconocimiento de los elementos, formación y estructura del lenguaje.

(b) Análisis semántico y generación de código intermedio: Revisión de la coherencia de lo ingresado, luego se transforma a código para una máquina abstracta.

(c) Generación de código objetivo: Transformación del código intermedio en código objeto.

(d) Mejora del código: Optimización del código, es una fase opcional.

11. ¿En qué circunstancias tiene sentido que un compilador pase o revise el código varias veces?

    (a) Cuando la cantidad de instrucciones son muy altas y es necesario una optimización.

12. ¿Cuál es el propósito de la tabla de símbolos en un compilador?

    (a) La traducción de la entrada.

13. ¿En la actualidad, que programa es mas eficiente, uno desarrollado desde cero en ensamblador o uno generado por un compilador?

    (a) Uno generado por compilador, actualmente se dispone de recursos suficientes para no tener que limitarnos tanto, añadiendo además que los proyectos son mucho más grandes y hacer algo tan cercano al hardware sería costoso para las personas.

GitHub: https://github.com/KEPCU/Report01FdLdP