Paradigmas de Programación
Notas

Emerson Monge H.

II Semestre 2024

Contenidos

Introducción

El estudio de los lenguajes de programación busca maneras de diseñar lenguajes que combinen poder expresivo, simplicidad y eficiencia.

El estudio de lenguajes de programación ayuda con el desarrollo de algoritmos eficaces, el uso, las elecciones, el aprendizaje y diseño de los lenguajes.

Requisitos de un Lenguaje de Programación

A fin de que un lenguaje de programación sea considerado como tal debe cumplir con:

1. Ser **universal** de manera que para todo problema que pueda resolverse en una computadora debe poder programarse la solución en el lenguaje independientemente de su tamaño.
2. Ser **natural** para facilitar (como mínimo) la resolución de problemas del área de aplicación.
3. **Implementable** de manera que todo programa bien formado en el lenguaje pueda ejecutarse sin problemas.

Sintaxis vs. Semántica

La sintaxis se refiere a la parte del lenguaje que afecta cómo los programas deben ser **escritos** por los programadores. La semántica determina cómo los programas con **compuestos** por los programadores y cómo son **comprendidos e interpretados** por el computador.

Procesadores de Lenguajes

Hace referencia a los sistemas que lleven a cabo el procesamiento, ejecución o la preparación para la ejecución de los programas. Aquí están incluídos:

1. Compiladores
2. Intérpretes
3. Herramientas auxiliares como editores de código y debbugers

Programas de Alto Nivel

Se consideran de alto nivel aquellos programas que sean independientes de la máquina donde se ejecutan. Son compilados en lenguaje máquina, interpretados de manera directa o una combinación de ambos.

1.1 Historia de los lenguajes de programación

Los primeros lenguajes de programación de alto nivel fueron desarrollados en los años 50. En esta época se desconocía si era factible una traducción automática entre un lenguaje orientado a usuarios y el lenguaje máquina.

1.2 Evolución de los lenguajes de programación

A partir de la invención de los lenguajes de alto nivel, muchos lenguajes han surgido combinando conceptos de lenguajes anteriores con nuevos enfoques.

Los lenguajes actuales no deben considerarse como el producto último y definitivo del diseño de lenguajes dado que nuevos conceptos y paradigmas aún hoy se están desarrollando.

Paradigmas de Programación

Paradigma	Descripción
Imperativo	Uso de comandos para actualizar variables caracterizado por el uso de variables, comandos y procedimientos.
Orientado a Objetos	Las variables pueden accederse únicamente por medio de operaciones asociadas a ellas caracterizado por el uso de objetos, clases y herencia.
Funcional	Hace uso de funciones como objetos de primera clase con avanzados sistemas de tipos. Se basan en funciones sobre listas y árboles, permiten la resolución de problemas **sin el uso de variables**.
Lógico	Uso de principios lógicos para hacer deducciones que resuelven problemas Caracterizado por el uso de relaciones. Se basan en un subconjunto de la lógica matemática. Se infieren asociaciones entre valores en lugar de calcular valores de salida a partir de valores de entrada.

Table 1.1: Paradigmas de Programación

Lenguajes de Programación

- **FORTRAN (Formula Translating System)** fue desarrollado por IBM en 1957, introduciendo expresiones simbólicas, arreglos y subprogramas con parámetros. Inicialmente de bajo nivel, evolucionó hasta su estandarización en 1997.

- **COBOL (Common Business-Oriented Language)**, creado en 1959, se destacó por su capacidad de descripción de datos y aplicaciones no numéricas, con su última versión estandarizada en 2002.
- **ALGOL 60 (Algorithmic Language)**, sucesor de ALGOL 58 y creado en 1960, fue pionero en el uso de estructuras de bloques y tuvo una gran influencia en lenguajes posteriores como CPL, BCPL, Simula y Pascal.
- **PL/I (Programming Language 1)**, propuesto por IBM en 1970, combinó capacidades numéricas y de procesamiento de datos de FORTRAN, ALGOL y COBOL. Introdujo manejo de excepciones y concurrencia, aunque resultó ser un lenguaje complejo y difícil de programar.
- **Pascal**, creado por Niklaus Wirth entre 1968 y 1969, fue popular en la enseñanza de principios de programación debido a su simplicidad y estructura eficiente.
- **C**, desarrollado por Dennis Ritchie entre 1969 y 1972, fue diseñado para UNIX y permitió programación tanto a bajo como a alto nivel. Su uso inadecuado puede resultar en sistemas no portables y difíciles de mantener.
- **C++**, diseñado por Bjarne Stroustrup a mediados de los años 80, extendió C con mecanismos para la manipulación de objetos, popularizando la programación orientada a objetos.
- **Java**, simplificación de C++ creada para programación distribuida y concurrente, es conocido por su portabilidad gracias a la máquina virtual Java.
- **LISP**, especificado por John McCarthy en 1958, fue el primer lenguaje funcional basado en listas y soportó variables y asignaciones.
- **Prolog (Programming in Logic)**, ideado en los años 70, es un clásico de la programación lógica, eficiente en su forma pura, aunque ampliado con características no lógicas para mayor utilidad.

Interpretación recursiva directa

2.1 Interpretación de un lenguaje imperativo

2.2 Interpretación de un lenguaje funcional

Programación funcional

Programación lógica

Programación imperativa

Programación orientada a objetos

Elementos avanzados de lenguajes de programación

7.1 Concurrencia, paralelismo y distribución

7.2 Sistema de tipos

7.3 Ligas entre programas de distintos lenguajes

7.4 Elementos del diseño de lenguajes de programación

7.5 Evaluación y selección de lenguajes de programación