Introducción a los lenguajes de Programación

martes, 21 de diciembre de 2021 16:36

- **Paradigmas de Programación**
  - Imperativo
    - Basado en Maquina de von Neumann
    - Ejecución secuencial
    - Variables de memoria
    - Asignación
    - E/S
    - Típicamente procedural
    - Dominan por mejor desempeño
    - Ejemplos
      - C
      - Pascal
      - Fortran
      - Algol
  - Funcional
    - Basado en calculo Lambda
    - Usa funciones y recursión
    - Ejemplos
      - LISP
      - Scheme
      - Haskell
  - Declarativo
    - Se declara lo que se quiere hacer, no como
    - Mas abstracto ya que no especifica un algoritmo
    - Ejemplos
      - SQL
      - Prolog
    - Lógico
      - Es una subcategoría de los lenguajes declarativos
      - Basado en el cálculo de predicados
      - Fundamentalmente basado en reglas como Prolog
  - Orientado a Objetos
    - Conjunto de objetos/piezas que interactúan controladamente, intercambiando mensajes
    - Extienden el paradigma imperativo
    - Ejemplos
      - Smalltalk
      - C++
      - Java

Ejemplo: Máximo común denominado (MCD)

```
int mcd(int a, int b) {   // C
  while (a != b) {
    if (a > b) a = a - b;
    else b = b - a;
  }
  return a;
}
```

```
(define mcd          ; Scheme
 (lambda (a b)
  (cond ((= a b) a)
        ((> a b) (mcd (- a b) b))
        (else (mcd (- b a) a)))))
```

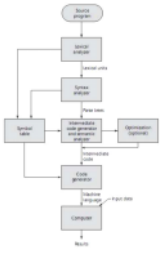
```
mcd(A,B,G) :- A = B, G = A.          % Prolog
mcd(A,B,G) :- A > B, C is A-B, mcd(C,B,G).
mcd(A,B,G) :- B > A, C is B-A, mcd(C,A,G).
```

- **Otros modelos de Programación**
  - Programación basada en eventos
    - El flujo de control está determinado por eventos que procesa el manejador de eventos
    - Ejemplos
      - Interfaces graficas
      - Manejo de interrupciones
      - Sistema de sensores
  - Programación Concurrente
    - Conjunto de procesos cooperativos que se pueden ejecutar en paralelo
    - Se requiere sincronizaciones el acceso a recursos compartidos
    - Ejemplos
      - Sistemas Operativos
      - Sistemas Distribuidos
  - Programación Visual
    - Permite crear programa manipulando objetos gráficos
    - Normalmente se integra a otros lenguajes
    - Ejemplos
      - Ingeniería de software
      - Kodu para juegos
      - LabVIEW para ingeniería

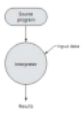
- **Abstracciones**
  - Datos
    - Primitivos
      - Tipos de datos básicos y variables
      - Enteros, reales y caracteres
    - Simples
      - Tipos de datos no estructurados
      - Puede ser primitivo o definido por el usuario en base a no primitivo
    - Estructurados
      - Permite agrupar conjuntos de datos en una unidad
      - Define n nuevo tipo de datos
      - Arreglos, registros, archivos de texto
  - Control
    - Sentencias
      - Abstrae conjunto de instrucciones
    - Estructuras de control
      - Secuenciación, condición y repetición
      - If, while e iteradores
    - Abstracción procedural
      - Permite invocar un procedimiento con un nombre y parámetros
      - Funciones, procedimientos y subprogramas
    - Concurrencia
      - Permite computación paralela
      - Procesos, hebras y tareas
      - Se introducen también abstracciones de comunicación
  - Tipos de Datos Abstractos
    - Agrupa datos y operaciones relacionadas en una unidad
    - Abstrae el tipo de dato con sus operaciones de la implementación del tipo
    - Módulos y Clases

- **Modelos de Implantación**
  - Compilación
    - Se traduce a lenguaje de máquina para su posterior ejecución
    - Ejemplos
      - C
      - C++
  - Interpretación
    - Una máquina virtual interpreta directamente el código fuente durante la ejecución
    - Ejemplos
      - LISP
      - Python
  - Esquema Hibrido
    - Se compila a un lenguaje intermedio que luego es interpretado por una máquina virtual
    - Ejemplos
      - Java
      - C#

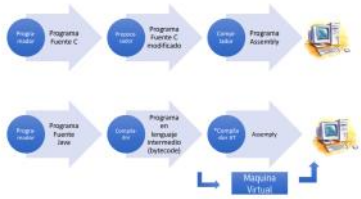
Compilación



Interpretación



Ejemplos de C y de Java



- **Programación "en grande"**
  - Modularizarían
    - Necesidad de descomponer los programas en unidades de desarrollo más pequeñas
    - Apoya el concepto de *ocultamiento de información* lo que reduce la carga cognitiva

- Compilación separada o independiente
  - Módulos de un programa se pueden traducir aparte
  - Facilita la mantención
- Reutilización
  - Módulos se pueden reutilizar para diferentes programas
  - Bibliotecas, Módulos y Paquetes
- Ambientes de Desarrollo
  - Se tienen ambientes de desarrollo con diferentes tipos de herramientas y facilidades para apoyar el proceso en todo ciclo de vida del software

- **Aspectos de Diseño**
  - Arquitectura
    - Maquina objetivo donde se ejecuta
    - La mayoría de los computadores siguen basados en el modelo de von Neumann
  - Estándares
    - Lenguajes populares tienden a estandarizarse, haciéndolos más portables
    - Se incorpora la estandarización a os tipos de datos primitivos y las bibliotecas, lo que hace más difícil la innovación
  - Sistemas legados
    - Mantener retrocompatibilidad
    - Permite mantener código legado como es el caso de Cobol y C++
    - Hace más complejo el diseño de los lenguajes

Criterios de Evaluación

Característica \ Criterio	Facilidad de Lectura	Facilidad de Escritura	Fiabilidad
Simplicidad	√	√	√
Ortogonalidad	√	√	√
Tipos de datos	√	√	√
Diseño de sintaxis	√	√	√
Soporte para abstracción		√	√
Expresividad		√	√
Prueba de tipos			√
Manejo de excepciones			√
Restricción de alias			√