

Principios de Lenguajes de Programación

Introducción

Facultad de Informática
Universidad Nacional del Comahue

Primer Cuatrimestre

Contenidos

Motivación de los LdP

Introducción

Criterios de Evaluación de los lenguajes

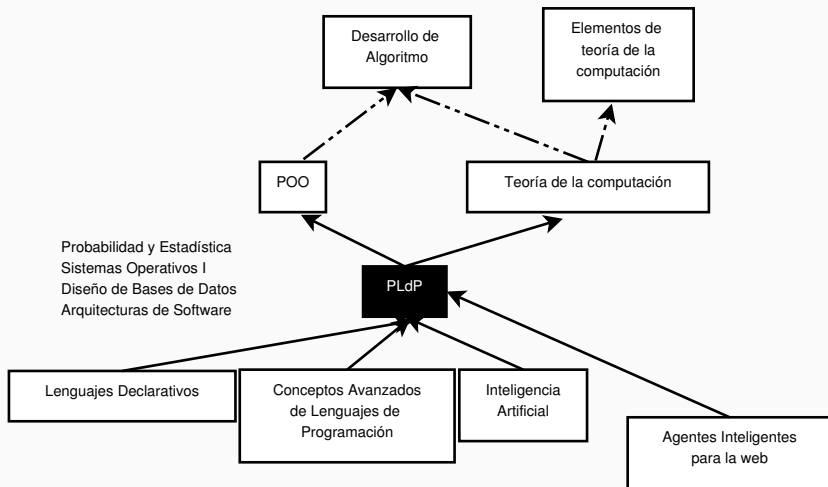
Clasificación de los lenguajes

Diseño y construcción de un LdP

Paradigmas de los LdP

Desafíos

¿Qué estudiamos en la carrera y hacia dónde vamos?



¿Cuál es la idea?

¿Qué lenguajes estudiamos y conocen?



- ▶ ¿en qué lenguajes de programación han trabajado?
- ▶ ¿Porqué los lenguajes de programación son de esa manera?
- ▶ ¿Cómo se implementan ciertas características particulares de algunos lenguajes?
- ▶ ¿Porqué hay tantos lenguajes de programación?

¿Cuál es la idea?

¿Qué lenguajes estudiamos y conocen?



- ▶ ¿en qué lenguajes de programación han trabajado?
- ▶ ¿Porqué los lenguajes de programación son de esa manera?
- ▶ ¿Cómo se implementan ciertas características particulares de algunos lenguajes?
- ▶ ¿Porqué hay tantos lenguajes de programación?

¿Cuál es la idea?

¿Qué lenguajes estudiamos y conocen?



- ▶ ¿en qué lenguajes de programación han trabajado?
- ▶ ¿Porqué los lenguajes de programación son de esa manera?
- ▶ ¿Cómo se implementan ciertas características particulares de algunos lenguajes?
- ▶ ¿Porqué hay tantos lenguajes de programación?

¿Cuál es la idea?

¿Qué lenguajes estudiamos y conocen?



- ▶ ¿en qué lenguajes de programación han trabajado?
- ▶ ¿Porqué los lenguajes de programación son de esa manera?
- ▶ ¿Cómo se implementan ciertas características particulares de algunos lenguajes?
- ▶ ¿Porqué hay tantos lenguajes de programación?

¿Por qué estudiar los LdP?

- ▶ Mejora el conocimiento básico para elegir los mejores y más apropiados lenguajes de programación
- ▶ Mejora el uso del lenguaje de programación
- ▶ Permite una mejor elección del lenguaje de programación
- ▶ Mejora la habilidad para desarrollar programas efectivos y eficientes
- ▶ Facilita el aprendizaje de un nuevo lenguaje de programación
- ▶ Mejorar la calidad y cantidad de conocimiento previo para la apropiada selección de lenguajes.

¿Por qué estudiar los LdP?

- ▶ Mejora el conocimiento básico para elegir los mejores y más apropiados lenguajes de programación
- ▶ Mejora el uso del lenguaje de programación
- ▶ Permite una mejor elección del lenguaje de programación
- ▶ Mejora la habilidad para desarrollar programas efectivos y eficientes
- ▶ Facilita el aprendizaje de un nuevo lenguaje de programación
- ▶ Mejorar la calidad y cantidad de conocimiento previo para la apropiada selección de lenguajes.

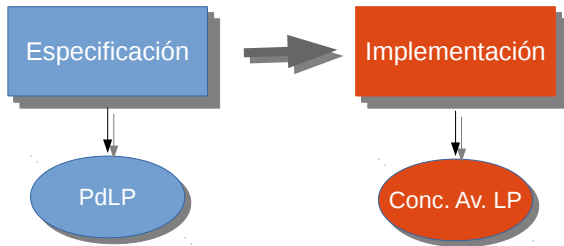
¿Por qué estudiar los LdP?

- ▶ Facilita el diseño de nuevos lenguajes de programación
- ▶ Comprender aspectos oscuros de los lenguajes.
- ▶ Incrementa el vocabulario y uso de conceptos y construcciones de lenguajes de programación
- ▶ Elegir modos alternativos de expresar cosas.
- ▶ Permite comprender mejor el significado de la implementación
- ▶ Realizar buenos debuggers, ensambladores, linkeadores y herramientas relacionadas.
- ▶ Hacer mejor uso de la tecnología de los lenguajes de programación a medida que aparecen.

¿Cuál es la idea?

¿Qué queremos desde la Cs. de la Computación

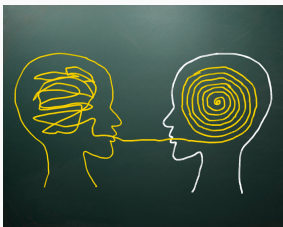
- ▶ Representación de la información
- ▶ Procesamiento de dicha información



¿Qué es un lenguaje de programación?

Un lenguaje en general,

medio del que se vale cualquier **individuo** para **comunicar** ideas y experiencias a otros individuos.



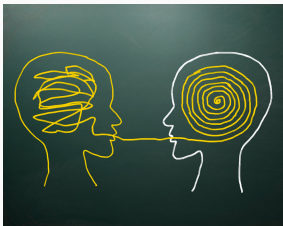
Están **almacenadas** en el **individuo emisor** y **almacenadas** *tal vez con otra representación* en el **receptor**.

El **lenguaje de programación** en general tiene la misma idea, solo que el receptor es una *computadora*, por ello es una de las principales herramientas en el proceso de desarrollo del software.

¿Qué es un lenguaje de programación?

Un lenguaje en general,

medio del que se vale cualquier **individuo** para **comunicar** ideas y experiencias a otros individuos.



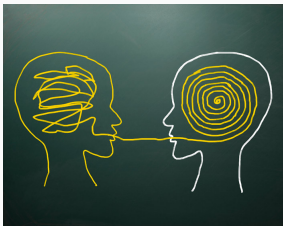
Están **almacenadas** en el **individuo emisor** y **almacenadas** *tal vez con otra representación* en el **receptor**.

El **lenguaje de programación** en general tiene la misma idea, solo que el receptor es una *computadora*, por ello es una de las principales herramientas en el proceso de desarrollo del software.

¿Qué es un lenguaje de programación?

Un lenguaje en general,

medio del que se vale cualquier **individuo** para **comunicar** ideas y experiencias a otros individuos.



Están **almacenadas** en el **individuo emisor** y **almacenadas** *tal vez con otra representación* en el **receptor**.

El **lenguaje de programación** en general tiene la misma idea, solo que el receptor es una *computadora*, por ello es una de las principales herramientas en el proceso de desarrollo del software.

¿Qué es un lenguaje de programación?

Los lenguajes de programación buscan ser:

modelo semántico humano



modelo computacional



¿Qué es un lenguaje de programación?

¿Qué es programar?

¿Qué es un lenguaje de programación?

¿Qué es programar?

- ▶ **Programar** = Plantear una solución para un problema dado

- ▶ En *Computación*, usualmente **Programar** es plantear una solución a un problema mediante un Paradigma de Programación (condiciona la forma en que se expresa la solución a un problema).

Así, el Lenguaje de Programación (que se encuadra en un determinado paradigma) es la herramienta que permite expresar nuestra solución.

¿Qué es un lenguaje de programación?

¿Qué es programar?

- ▶ **Programar** = Plantear una solución para un problema dado
- ▶ En *Computación*, usualmente **Programar** es plantear un solución a un problema mediante un Paradigma de Programación (condiciona la forma en que se expresa la solución a un problema).

Así, el Lenguaje de Programación (que se encuadra en un determinado paradigma) es la herramienta que permite expresar nuestra solución.

¿Qué es un lenguaje de programación?

¿Qué es programar?

- ▶ **Programar** = Plantear una solución para un problema dado
- ▶ En *Computación*, usualmente **Programar** es plantear un solución a un problema mediante un Paradigma de Programación (condiciona la forma en que se expresa la solución a un problema).

Así, el Lenguaje de Programación (que se encuadra en un determinado paradigma) es la herramienta que permite expresar nuestra solución.

¿Qué es un lenguaje de programación?



¿Qué es un lenguaje de programación?

Un lenguaje de programación es:

- ▶ Un **sistema notacional**
- ▶ para describir **computaciones**
- ▶ de una forma **legible**
 - ▶ tanto para la máquina
 - ▶ comprensible para el ser humano

¿Qué es un lenguaje de programación?



¿Qué es un lenguaje de programación?

Un lenguaje de programación es:

- ▶ Un **sistema notacional**
- ▶ para describir **computaciones**
- ▶ de una forma **legible**
 - ▶ tanto para la máquina
 - ▶ comprensible para el ser humano

¿Qué es un lenguaje de programación?



¿Qué es un lenguaje de programación?

Un lenguaje de programación es:

- ▶ Un **sistema notacional**
- ▶ para describir **computaciones**
- ▶ de una forma **legible**
 - ▶ tanto para la **máquina**
 - ▶ comprensible para el **ser humano**

¿Qué es un lenguaje de programación?



¿Qué es un lenguaje de programación?

Un lenguaje de programación es:

- ▶ Un **sistema notacional**
- ▶ para describir **computaciones**
- ▶ de una forma **legible**
 - ▶ tanto para la **máquina**
 - ▶ comprensible para el **ser humano**

¿Qué es un lenguaje de programación?



¿Qué es un lenguaje de programación?

Un lenguaje de programación es:

- ▶ Un **sistema notacional**
- ▶ para describir **computaciones**
- ▶ de una forma **legible**
 - ▶ tanto para la **máquina**
 - ▶ comprensible para el **ser humano**

El arte de diseñar lenguajes

Hay una gran variedad de lenguajes y continúan apareciendo nuevos ¿por qué?

- ▶ Evolución
- ▶ Propósito especial
- ▶ Preferencia personal

¿qué convierte a un lenguaje en exitoso?

El arte de diseñar lenguajes

Hay una gran variedad de lenguajes y continúan apareciendo nuevos ¿por qué?

- ▶ Evolución
- ▶ Propósito especial
- ▶ Preferencia personal

¿qué convierte a un lenguaje en exitoso?

- ▶ Poder expresivo
- ▶ Facilidad de uso para novatos (baja curva de aprendizaje)
- ▶ Facilidad de implementación
- ▶ Código abierto
- ▶ Excelentes compiladores
- ▶ Inercia, economía, quien lo promueve

El arte de diseñar lenguajes

Hay una gran variedad de lenguajes y continúan apareciendo nuevos ¿por qué?

- ▶ Evolución
- ▶ Propósito especial
- ▶ Preferencia personal

¿qué convierte a un lenguaje en exitoso?

- ▶ Poder expresivo
- ▶ Facilidad de uso para novatos (baja curva de aprendizaje)
- ▶ Facilidad de implementación
- ▶ Código abierto
- ▶ Excelentes compiladores
- ▶ Inercia, economía, quien lo promueve

El arte de diseñar lenguajes

Hay una gran variedad de lenguajes y continúan apareciendo nuevos ¿por qué?

- ▶ Evolución
- ▶ Propósito especial
- ▶ Preferencia personal

¿qué convierte a un lenguaje en exitoso?

- ▶ Poder expresivo
- ▶ Facilidad de uso para novatos (baja curva de aprendizaje)
- ▶ Facilidad de implementación
- ▶ Código abierto
- ▶ Excelentes compiladores
- ▶ Inercia, economía, quien lo promueve

El arte de diseñar lenguajes

Hay una gran variedad de lenguajes y continúan apareciendo nuevos ¿por qué?

- ▶ Evolución
- ▶ Propósito especial
- ▶ Preferencia personal

¿qué convierte a un lenguaje en exitoso?

- ▶ Poder expresivo
- ▶ Facilidad de uso para novatos (baja curva de aprendizaje)
- ▶ Facilidad de implementación
- ▶ Código abierto
- ▶ Excelentes compiladores
- ▶ Inercia, economía, quien lo promueve

El arte de diseñar lenguajes

Hay una gran variedad de lenguajes y continúan apareciendo nuevos ¿por qué?

- ▶ Evolución
- ▶ Propósito especial
- ▶ Preferencia personal

¿qué convierte a un lenguaje en exitoso?

- ▶ Poder expresivo
- ▶ Facilidad de uso para novatos (baja curva de aprendizaje)
- ▶ Facilidad de implementación
- ▶ Código abierto
- ▶ Excelentes compiladores
- ▶ Inercia, economía, quien lo promueve

El arte de diseñar lenguajes

Hay una gran variedad de lenguajes y continúan apareciendo nuevos ¿por qué?

- ▶ Evolución
- ▶ Propósito especial
- ▶ Preferencia personal

¿qué convierte a un lenguaje en exitoso?

- ▶ Poder expresivo
- ▶ Facilidad de uso para novatos (baja curva de aprendizaje)
- ▶ Facilidad de implementación
- ▶ Código abierto
- ▶ Excelentes compiladores
- ▶ Inercia, economía, quien lo promueve

Desarrollo de Lenguajes de Programación

El **lenguaje de programación** surge de la **vinculación** entre las diferentes **metodologías de diseño** y las **arquitecturas** de la computadoras.

Otro de los factores que terminan teniendo especial relevancia es el **dominio de aplicación** pretendido.

Características deseables en el software

De cualquier software se espera, básicamente, que sea:

- ▶ confiable
- ▶ extensible
- ▶ reusable
- ▶ robusto
- ▶ eficiente
- ▶ flexibilidad
- ▶ etc.

Características deseables en el software

De cualquier software se espera, básicamente, que sea:

- ▶ confiable
- ▶ extensible
- ▶ reusable
- ▶ robusto
- ▶ eficiente
- ▶ flexibilidad
- ▶ etc.

Características deseables en el software

Los lenguajes de programación también son software pero al tener otro tipo de propósito se espera que posea otras características:

- ▶ facilidad de lectura y escritura (legibilidad)
- ▶ seguridad
- ▶ costo

Criterio de evaluación de los lenguajes

- ▶ **Readability (Legibilidad)**: la facilidad con la que se pueden leer y comprender los programas.
- ▶ **Writability (Escribibilidad)**: la facilidad con la cual un lenguaje puede usarse para crear programas.
- ▶ **Reliability (Fiabilidad)**: ajuste a las especificaciones (ie, cumple las especificaciones)
- ▶ **Costo**: los costos asociados

Criterio de evaluación

Características que influyen en los criterios

- ▶ Simplicidad (LEC)
- ▶ Ortogonalidad (LEC)
- ▶ TD - Diseño de sintáxis (LEC)
- ▶ Soporte para la abstracción (EC)
- ▶ Expresividad (EC)
- ▶ Chequeo de tipos (C)
- ▶ Manejo de excepciones (C)
- ▶ Aliasing (Alias) (C)

Criterio de evaluación

Table 1.1 Language evaluation criteria and the characteristics that affect them

Characteristic	CRITERIA		
	READABILITY	WRITABILITY	RELIABILITY
Simplicity	•	•	•
Orthogonality	•	•	•
Data types	•	•	•
Syntax design	•	•	•
Support for abstraction		•	•
Expressivity		•	•
Type checking			•
Exception handling			•
Restricted aliasing			•

Fuente: Sebesta, Robert, Concepts of Programming Languages.

Criterio de evaluación

Legibilidad

- ▶ La facilidad para leer e interpretar programas es fundamental.
Características que contribuyen a la legibilidad:
- ▶ Características:
 - ▶ **Simplicidad**: Se obtiene en gran medida de combinar un número de pequeño de constructores primitivos y un uso limitado (ni mucho-ni poco) del concepto de ortogonalidad.
 - ▶ **Conjunto manejable** de constructores y características
 - ▶ Problemas de cantidad y **multiplicidad** de características Ej `count++`, `count=count+1`
 - ▶ **Sobrecarga** mínima de operadores Ej `+`

Legibilidad

- ▶ Características:
 - ▶ **Ortogonalidad:** Básicamente significa que un conjunto pequeño de constructores primitivos, puede ser combinado en número relativamente pequeño a la hora de construir estructuras de control y datos. Cada combinación es legal y con sentido.
 - ▶ Más fácil de aprender y leer
 - ▶ Conjunto pequeño de constructores primitivos para combinar de pocas maneras
 - ▶ Todas las combinaciones son válidas
 - ▶ Sin ortogonalidad aparecen excepciones a reglas

Criterio de evaluación

Legibilidad

- ▶ Características:
 - ▶ **Definición de tipos de datos y estructuras:** Contar con estructuras legibles (sólo uso de goto - agregado del tipo boolean) Ej. Bandera=1 o Bandera=true
 - ▶ **Aspectos sintácticos:** Formas de los identificadores - palabras reservadas - forma y significado: constructores autodescriptivos, palabras claves significativas

Criterio de evaluación: Escribibilidad

Cuán fácil es un lenguaje para escribir un programa.

Simplicidad y ortogonalidad

Pocos constructores, pequeño número de primitivas, conjunto pequeño de reglas para combinar que sean ortogonales es mejor. Mucha ortogonalidad genera una menor detección de errores. Si el lenguaje no es simple solo se aprende y utiliza una pequeña porción de él.

Capacidad de abstracción

Capacidad de definir y usar estructuras u operaciones complicadas de manera que sea posible ignorar muchos de los detalles.

Expresividad

El lenguaje posee formas relativamente convenientes de expresar ciertas operaciones, por ejemplo `contador++` en lugar de `contador = contador + 1` o el uso del `for` en lugar del `while` equivalente.

Criterio de evaluación

Fiabilidad

Se dice que un programa es confiable si cumple con sus especificaciones bajo todas las condiciones

- ▶ **Chequeo de tipos:** Control de errores de tipos
- ▶ **Manejo de excepciones:** Interceptar errores en tiempo de ejecución y realizar posibles correcciones
- ▶ **Aliasing:** Tener dos o mas nombres para acceder a una memoria
- ▶ **Legibilidad y escribibilidad:** Si no existe método “natural” de expresar algoritmos requiere aproximaciones “poco naturales”.

Criterio de evaluación

Costo - Categorías

- ▶ Aprender (a escribir programas cercanos a aplicaciones particulares)
- ▶ Usar (entrenar programadores para el uso del lenguaje)
- ▶ Compilación
- ▶ Ejecución
- ▶ Sistema de implementación del lenguaje
- ▶ Disponibilidad
- ▶ Confiabilidad (escasa confiabilidad conlleva altos costos)
- ▶ Mantenimiento
- ▶ Licencias

Compromisos en el diseño de los LdP

- ▶ **Confiabilidad vs. Costo de Ejecución**
ej: Java exige que todas las referencias a elementos de un arreglo se controle rango de los índices
- ▶ **Legibilidad vs. Escribibilidad**
APL brindar muchos operadores poderosos (con un gran número de símbolos asociados) permitiendo cálculos muy complejos en un programa compacto, pero ...
- ▶ **Escribibilidad (flexibilidad) vs. Confiabilidad**
Punteros en C++, poderosos y flexibles pero poco confiables

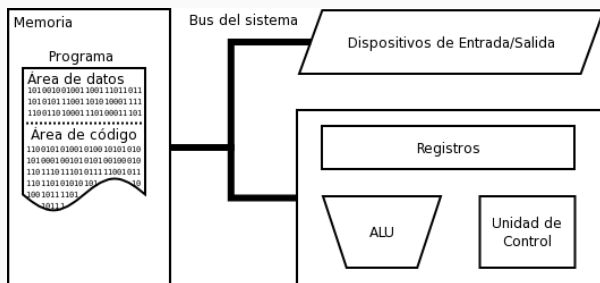
Clasificación de los lenguajes

- ▶ Por aplicabilidad o especificidad
- ▶ Por forma de ejecución
- ▶ Por Paradigma de Programación Asociado
- ▶ Por Nivel de Abstracción
- ▶ Por Tipo de Implementación
- ▶ Por Nivel de Generación (1GL, ... , 4GL, 5GL)
- ▶ ...

Influencia en el diseño de los LdP

- ▶ **Arquitectura de las Computadoras:** Los lenguajes se desarrollan alrededor de las arquitectura prevalentes
- ▶ **Metodologías de Programación:** Las nuevas metodologías de desarrollo de software (por ej, desarrollo OO) guían los nuevos paradigmas de programación, y por ende, los nuevos lenguajes de programación

Arquitectura Von Neuman

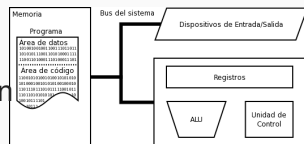


- John von Neuman es considerado como el inventor del **programa almacenado** en una computadora digital

Influencia de las arquitecturas

- ▶ Desarrollo a partir de Von Neumann
- ▶ Uso de lenguajes imperativos:

- ▶ Datos y programas guardados en la misma memoria
- ▶ Memoria separada de la CPU
- ▶ Instrucciones y datos encolados a la CPU
- ▶ Base de los lenguajes imperativos
 - ▶ Variables modelan las celdas de memoria
 - ▶ Las sentencias modelan encolado a CPU
 - ▶ La iteración es muy eficiente



Evolución de los lenguajes de programación

Factores críticos:

- ▶ Dominios de aplicación
- ▶ Conceptos: Abstracción
- ▶ Paradigmas

Dominios de aplicación

○ Aplicaciones científicas.

- Gran cantidad de cálculos sobre grandes números en punto flotante.
- El principal exponente en esta categoría es Fortran

EFICIENCIA

○ Aplicaciones de negocios

- Gran producción de reportes, manejo de números decimales y caracteres.
- El lenguaje más destacado: COBOL

PORTABILIDAD

○ Inteligencia Artificial

- Manipulación de símbolos en lugar de manipulación numérica
- Ejemplo: LISP

COMPLEJIDAD

○ Programación de Sistemas – Base de datos

- Búsqueda de eficiencia (por uso continuo)
- Lenguajes ensambladores o C

SEGURIDAD

○ Software para la web

- Colección ecléctica de lenguajes:
 - Mark-up languages: XHTML
 - Scripting: PHP
 - Propósito general: JAVA

INTEROPERABILIDAD

Evolución en los conceptos: Abstracción

- ▶ Nombres simbólicos
- ▶ Expresiones
- ▶ Subprogramas
- ▶ Tipos de datos
- ▶ Estructuras de control
- ▶ A nivel instrucción
- ▶ A nivel unidad
- ▶ Encapsulamiento
- ▶ Abstracción de datos
- ▶ Polimorfismo

Abstracción

- ▶ **Nombres simbólicos** con restricciones (al principio se utilizaban 0 y 1 en tarjetas perforadas)
- ▶ **Expresiones** más cercanas a la realidad (la expresión se escribe en una línea, no como una secuencia de instrucciones) Introducción de los primeros chequeos: que exista la operación, que los operandos sean accesibles, etc.
- ▶ **Subprogramas** pensados solo como una forma de agrupar código

Abstracción

► Tipos de datos

- Los símbolos indexan variables de memoria con semántica, ya no hay locaciones de memoria anónimas.
- Cada celda está asociada a un tipo
- Avanza el nivel de confiabilidad (chequeo y seguridad)
- Solo tipos de datos que proveía el lenguaje.
- En la siguiente generación de lenguajes se extendió a tipos adecuados a diversas áreas de aplicación (definidos por el usuario)
- La siguiente generación brinda pocos tipos primitivos; pero también más constructores generales combinables entre si (+ ortogonal)

Estructuras de control

A nivel instrucción: Programación Estructurada

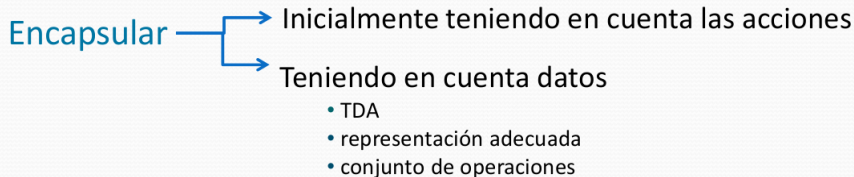
- ▶ Hasta ahora había secuencia y transferencia de control (condicional e incondicional) y venían ligadas a la arquitectura subyacente La programación estructurada propone utilizar:
 1. Secuencia
 2. Iteración
 3. Condicional
- ▶ En base a las estructuras de control surgen luego la ejecución simétrica (o concurrente) y manejo de excepciones

Estructuras de control

A nivel unidad: Diseño Top-Down

- ▶ Utilización de subprogramas para partir el problema en sub-problemas
- ▶ Funcionalidad de la unidad: - Para resolver sub-problemas
- Para resolver sub-problemas
- ▶ Esquemas de relación entre unidades
 - ▶ Jerárquico
 - ▶ Simétrico
 - ▶ Paralelo
 - ▶ Latente

Encapsulamiento



- ▶ Para que el lenguaje sea bueno en abstracción debe tener encapsulamiento y mecanismo para crear instancias
- ▶ Encapsulamiento sin abstracción = agrupamiento sin ocultamiento

Paradigma de los lenguajes de programación

Paradigma de Programación

- Conjunto coherente de métodos para resolver un problema
- Tiene principio básico
- Guía el proceso de desarrollo del software

Paradigma de Programación



Colección de patrones conceptuales (estructuras o reglas) que juntos modelan el proceso de diseño y que determinan en última instancia la estructura de un programas

- ▶ Un estilo de Programación
- ▶ Una manera de visualizar la ejecución
- ▶ Una forma de resolver problemas de programación
- ▶ Un enfoque, una perspectiva, una filosofía sobre cómo programar

Paradigma de los lenguajes de programación

Los 4 principales paradigmas

- ▶ Imperativo o procedural (ej. Pascal, Fortran, C)
- ▶ Orientado a Objetos (ej. Smalltalk, Java)
- ▶ Funcional (ej. Haskell, Lisp, ML)
- ▶ Lógico (ej. Prolog, F-Prolog)
- ▶ Otros: Multiparadigma (ej. Curry (Programación Lógico-Funcional))

Paradigma de los lenguajes de programación

Los 4 principales paradigmas

- ▶ Imperativo o procedural (ej. Pascal, Fortran, C)
- ▶ Orientado a Objetos (ej. Smalltalk, Java)
- ▶ Funcional (ej. Haskell, Lisp, ML)
- ▶ Lógico (ej. Prolog, F-Prolog)
- ▶ Otros: Multiparadigma (ej. Curry (Programación Lógico-Funcional))

Paradigmas de los lenguajes

Paradigma Imperativo

Un programa es una secuencia de **instrucciones** que indican el flujo de la ejecución.

Señal dada para que se realice
el cambio de estado del
autómata

Las principales características son:

1. ejecución secuencial de instrucciones
2. uso de variables representando valores de locaciones de memoria
3. uso de sentencias de asignación para cambiar los valores de las variables, permitiendo así al programa operar sobre las locaciones de memoria

Paradigmas de los lenguajes

Paradigma Imperativo

Se busca estructurar el control realizando una programación estructurada y modular con abstracción de datos para fomentar la reusabilidad y extendibilidad

Los programas se construyen siguiendo una aproximación:

- Top-down
 - Modular
- Solo sub-programas
 - Dividir para conquistar
 - Existen abstracciones algorítmicas

Abstracción a nivel instrucción

Agrupar instrucciones en unidades –
Procedure de Pascal

Abstracción de expresiones

Function de Pascal

Paradigmas de los lenguajes


Paradigma Orientado a Objetos

- Se caracteriza por reconocer las entidades del problema (similar a la abstracción de datos)
- Entidad = **Objetos**
- Comunicación por **mensajes**, diferente a la semántica de llamadas a procedimiento

Caracterizado por atributos y comportamiento
(de acuerdo a su propósito y habilidades)

Objetivos:

- Mejorar la **reusabilidad** del software
- Mejorar la **extendibilidad** del software

Para algunos  Quiebre total con lo anterior
Nuevo eslabón de la línea estructurada

Paradigmas de los lenguajes

Paradigma Lógico

Basado en lógica de primer orden (lenguaje preciso para expresar conocimiento)

Lenguaje representativo: ProLog

Los fundamentos del paradigma son:

- a) Deducir consecuencias a partir de premisas
- b) Estudiar o decidir el valor de verdad de una sentencia a partir del valor de verdad de otras
- c) Establecer la consistencia entre hechos y verificar la validez de argumentos

Paradigmas de los lenguajes

Paradigma Lógico

Características de los lenguajes lógicos

- a) Eliminación del control
- b) El concepto de variable es más matemático, son nombres que retienen valores
- c) Establecen “que” es lo que se debe hacer sin dar ninguna especificación sobre el “como” hacerlo

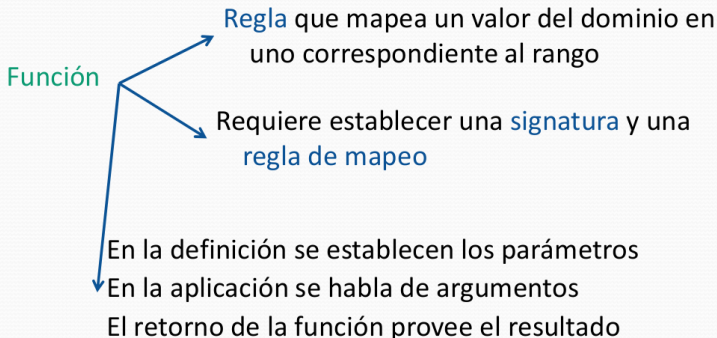
Características de los programas lógicos

- a) Conjuntos de axiomas que establecen relaciones
- b) Definen un conjunto de consecuencias que determinan el significado
- c) Son teoremas y la ejecución es una prueba automática

Paradigmas de los lenguajes

Paradigma Funcional

La esencia de esta metodología esta en componer funciones para definir otras más complejas.



Paradigmas de los lenguajes

Paradigma Funcional

Características de los lenguajes funcionales

- a) Define un conjunto de datos
- b) Provee un conjunto de funciones primitivas
- c) Provee un conjunto de formas funcionales
- d) Requiere de un operador de aplicación

Características de los programas funcionales

- a) Semántica basada en valores
- b) Transparencia referencial
- c) Regla de mapeo basada en combinación o composición
- d) Las funciones son “ciudadanos” de primer orden

Paradigmas de los lenguajes

Metodologías de Programación (paradigmas)

- ▶ 1950s y 1960s: aplicaciones simples, la preocupación es la eficiencia
- ▶ Fines de los 1960s: Eficiencia de las personas es importante: legibilidad, mejores estructuras de control
 - ▶ Programación Estructurada
 - ▶ Diseño Top-Down y refinamiento sucesivo
- ▶ Fines de los 1970s: de Orientación a Procesos a Orientados a Datos: mayor abstracción
- ▶ Los 1980s: Programación Orientada a Objetos
- ▶ Los 2000s: Ubíquo? Aspectos? Grid? Scripting? Entorno?

Estándarización de los lenguajes

El convenio suele estar reflejado en un documento (un libro) que se hace público y mediante el cual se determinan las reglas de interpretación correcta de los programas

- ▶ algunos lenguajes están definidos por un documento estandarizado en un organismo oficial como ISO (p.ej. C++ es el estándar ISO/IEC 14882, de 1998)
- ▶ en otros casos la descripción del lenguaje no está oficialmente estandarizada, el lenguaje se define por el documento de referencia que lo describe (pej. Java, descrito en este libro <http://java.sun.com/docs/books/jls/>)

Bibliografía

- ▶ Pratt, Terrance W., Programming Languages: Design and Implementation.
- ▶ Sebesta, Robert, Concepts of Programming Languages.
- ▶ Material de apoyo en pedco.

Desafíos

- ▶ Nombre y explique otro criterio por el cual puedan ser juzgado los lenguajes además de los discutidos en estas transparencias.
- ▶ Ventajas y desventajas de las características ortogonales en un lenguaje.
- ▶ Java utiliza una llave derecha para marcar el final de todas las sentencias compuestas. ¿Cuáles son los argumentos a favor y en contra de este diseño?