Diseño de un LP

* Conceptos como valores, tipos de dato, variables, almacenamiento, ligaduras, alcance y abstracción de procedimientos son tan fundamentales que se encuentran en casi todos los lenguajes de programación.
* Cada lenguaje nuevo de programación debe apoyar la abstracción de datos de alguna forma, la falta de esta es una seria deficiencia en un lenguaje actual.
* Un lenguaje nuevo debería admitir unidades genéricas, polimorfismo por inclusión/herencia o paramétrico (conceptos asociados fundamentalmente con lenguajes imperativos, orientados a objetos y funcionales).
* Es importante que el diseñador tenga en claro para que va a ser utilizado el lenguaje.

Simplicidad

* Propiedad del diseño determinada por el número de componentes básicos, la minimalidad de los conceptos y las consistencias de sus estructuras.
* Siempre debe ser un objetivo del diseño del lenguaje.
* Va a medir que esos conceptos requeridos y la integridad de sus instrucciones o de sus estructuras sean lo suficientemente simples como para ser entendidas por el programador.

Expresividad

* Habilidad que posee un lenguaje para reflejar el significado deseado por el programador lo mas naturalmente posible.
* El programa debe poder modelar con naturalidad la solución del problema y su dominio de aplicación.
* La expresividad de un lenguaje se ve fuertemente afectada por la elección de que símbolo va a denotar cada operación.
* Puede entrar en conflicto con la simplicidad.

Seguridad

* Propiedad del diseño que desalienta la posibilidad de escribir programas con errores.
* Las características que aumentan la seguridad cargan al programador con tareas de especificar todas las declaraciones posibles dentro del programa mientras esta escribiendo ese código.
* Puede reducir la expresividad y la flexibilidad del lenguaje.
* Tipos de datos, chequeo de tipos, declaraciones de variables, eliminación del uso y la aritmética de punteros, la recolección de basura son características del lenguaje que **aumentan la seguridad**.
* Uso de variables globales, parámetros pasados por referencia, uso de alias **disminuyen la seguridad.**

Regularidad

* Cualidad del lenguaje que refleja la adecuada combinación de los conceptos que lo caracterizan al diseñar un lenguaje
* Para lograr la máxima potencialidad con un numero determinado de conceptos, el programador debe poder combinar estas características de manera regular sin restricciones innecesarias ni interacciones raras entre conceptos.
* Puede dividirse en tres conceptos diferentes:
  + Generalidad: se logra eliminando casos especiales en la disponibilidad y uso de los constructores; y combinando constructores sumamente relacionados en uno mucho mas general.
  + Ortogonalidad: atributo del lenguaje de ser capaz de combinar varias características que posee el lenguaje entre si en todas las combinaciones posibles de forma tal que todas tengan significado no generando resultados inesperados.
  + Uniformidad: cosas similares deben verse de forma similar y tener significado similar o a la inversa.
* Existen otros conceptos que van a influir en la regularidad:
  + Integridad de los tipos de datos: todos los tipos de datos en el lenguaje deben tener el mismo status. Ej. los parámetros y los resultados de la función no deben restringirse solo a tipos primitivos.
  + Abstracción: facilita al diseñador la abstracción sobre categorías sintácticas distintas de las categorías usuales. Ej procedimiento abstraen sobre sentencias e instrucciones.
  + Unidades genéricas: abstraen sobre declaraciones.
  + Tipos parametrizados: abstraen sobre tipos.
* La generalidad y la ortogonalidad pueden hacer muchos mas complejo el lenguaje de programación.

Robustez

* Habilidad que tiene el lenguaje de tratar eventos no deseados durante la ejecución.
* Si el sistema es robusto, es predecible hasta en situaciones anómalas.
* No es lo mismo que seguridad, estas son características que brinda el lenguaje para que al escribir el código ya el programador este escribiendo un código mas seguro, en cambio la robustez es un principio que se cumple cuando el programa se esta ejecutando, no cuando el programador lo esta escribiendo.
* Para que un lenguaje sea robusto debe incluir el manejo de excepciones, inclusión de verificación en el exceso de los limites de memoria o rangos de los subíndices, estas son todas situaciones que se pueden presentar en ejecución que escapan la escritura del código y que el lenguaje debe proveer los mecanismos para poder controlarlos y que el sistema siga funcionando.
* Robustez obligatoria en dominio de aplicación critica.

Flexibilidad

* Facilidad que permite que un mismo programa pueda realizar variaciones sobre una misma entidad sin necesidad de realizar modificaciones.
* Los tiempos de ligadura en un lenguaje imperativo son una característica que afecta directamente la flexibilidad de un lenguaje, los lenguajes interpretados facilitan esta característica ya que la ligadura ocurre en tiempo de ejecución, lo que lo hace mas flexible, si se produciría en tiempo de compilación seria mas seguro.
* En contraposición con la seguridad.
* El diseño de un lenguaje de programación que pretenda cumplir con este principio, inevitablemente tendrá que reducir o no podrá cumplir con el principio de seguridad.

Eficiencia

* Principio que refleja la relación entre los recursos utilizados y los objetivos conseguidos.
* Cada lenguaje debe ser capaz de hacer una implementación aceptablemente eficiente, el grado de aceptabilidad depende del dominio de aplicación para lo que el lenguaje fue diseñado.
* Costo-Beneficio: el programador debe decidir si los beneficios de cada concepto superan el costo.
* Eficiencia de la implementación.
* Eficiencia del código: diseño del lenguaje de programación debe ser tal que un traductor pueda generar código ejecutable eficiente. Ej optimización de código en el proceso de compilación.
* Eficiencia de la traducción: el diseño del lenguaje debe permitir que el código fuente se traduzca con rapidez y con un traductor de tamaño razonable. Esto se logra con una definición formal de la sintaxis y con que el lenguaje exija que todas la variables se definan antes de ser utilizadas.
* Eficiencia de la programación: estrechamente relacionada con la facilidad de escritura del lenguaje y su expresividad, con la potencia y la generalidad de los mecanismos de abstracción y con una sintaxis que tiene que ser concisa y la no declaración de variables antes de su uso.
* La eficiencia de un lenguaje esta fuertemente influenciada por la selección de los conceptos o de las características del lenguaje.
* Algunos conceptos como el tipado dinámico, el polimorfismo paramétrico, la orientación a objetos y la recolección de basura son conceptos costosos en el momento de la ejecución.

Reusabilidad

* Propiedad que permite que un componente de software pueda ser utilizado en otras aplicaciones.
* Conceptos asociados para cumplir con este principio:
  + Abstracción de datos.
  + Abstracción de procesos.
  + Unidades genéricas.
  + Polimorfismo paramétrico.
  + Polimorfismo por inclusión.
* También se puede analizar el alcance de las variables globales, las constantes simbólicas, la flexibilidad en las clases.

Extensibilidad

* Posibilidad de cambios y adiciones respecto de las especificaciones iniciales del lenguaje.
* Refleja hasta que punto un lenguaje de programación tiene mecanismos generales para que el usuario agregue nuevas funciones a un lenguaje.
* La idea es que el programador elija un conjunto de características que conformaran el lenguaje núcleo y proporcionar características via bibliotecas estándares y cuales características no especificar de ningún modo.
* Esto se manifiesta con la capacidad que tiene el lenguaje de permitir operadores y nuevas abstracciones de datos definidas por el programador.
* Pocos lenguajes son extensibles. A estos se los caracteriza por:
  + Definir nuevos tipos de datos.
  + Añadir funciones a una librería.
  + Añadir palabras claves y constructores.
  + Herencia.
  + Modularidad.
* Es de los principios mas importantes en el diseño de un lenguaje de programación en la actualidad.
* La simplicidad sin la extensibilidad no seria un principio a cumplir.

Portabilidad

* Habilidad para compilar y ejecutar programas en diferentes plataformas, sin realizar cambios en su código fuente.
* La definición del lenguaje de programación debe tener una alta independencia del hardware y debe estar estandarizado por algún organismo, esto facilitaría que los lenguajes puedan transportarse desde un entorno de programación a un entorno de ejecución diferente.

Estandarizacion

* Todos los lenguajes de programación tienen un ciclo de vida.
* Cada nuevo lenguaje que se diseña pasa por una serie de etapas desde su diseño inicial hasta que se comienza a utilizar de manera rutinaria por los programadores y su estandarizacion (si se solicita).
* Como determinar la validez de un código?:
  + Manual de referencia del lenguaje.
  + Podemos escribir un programa en ese lenguaje y ver que sucede.
  + Leer la definición estándar del lenguaje. (No ligado como los otros dos a una implementación particular).
* Todas las implementaciones deben alinearse con el estándar, para así evitar comportamientos inadecuados y garantizar una uniformidad del lenguaje.
* Tipos de estándares:
  + Patentados: definiciones elaboradas por las compañías que desarrollo el lenguaje y es su propietaria.
  + Por consenso: documentos elaborados por organizaciones con base a un acuerdo entre participantes pertinentes.
* Cada país cuenta con su organismo estandarizador, el desarrollo de las normas sigue un proceso similar en cada uno de ellos. En un momento dado un grupo decide que un lenguaje requiere un estándar, el organismo normativo de ese país correspondiente organiza un grupo de trabajo de voluntarios que va a desarrollar esa norma para ese lenguaje. Cuando ese grupo llega a un consenso sobre la norma se somete a votación de todos los interesados, se resuelven los desacuerdos y se produce el estándar del lenguaje.
* Para utilizar los estándares de una manera eficaz es necesario muchas veces revisar algunas cuestiones:
  + Oportunidad: Cuando estandarizar un lenguaje?, puede haber tres etapas:
    - Tardía: cuando se estandariza hay demasiadas implementaciones incompatibles entre si, lo que causo muchos problemas. Ej Fortran.
    - Temprana: se estandariza antes de haber una implementación o un uso masivo. No esta claro si el lenguaje va a funcionar o no. Ej ADA
    - Lo ideal es estandarizar cuando ya haya un masivo uso del lenguajes y no haya varias implementaciones.
  + Conformidad: Que significa que un programa adhiere a un estándar y que un compilador compila un estándar?
    - Si existe un estándar para un lenguaje se suele hablar de conformidad con respecto al estándar
    - Un programa es **conforme** si solo utiliza características definidas en el estándar.
    - Un **compilador conformable** es uno que toma un programa conforme y produce un programa ejecutable que genera la salida correcta. Si un compilador o traductor agrega características adicionales, cualquier programa que las utilice **no es conformable** y el estándar no dice cuales serán los resultados obtenidos.
  + Obsolescencia: Cuando envejece un estándar y como se modifica?, Se desarrollaron los conceptos de:
    - Características Obsolescentes: es candidata a ser descartada en la próxima versión del estándar. Se advierte a los programadores
    - Características desaprobadas: se puede volver obsolescente en el próximo estándar.