Trabajo Práctico Número 1
Teoría de la Computación

Ulises C. Ramirez

22 Abril 2018

1 ¿Qué es un lenguaje de programación?

1.1 Abstracciones en los lenguajes de programación

Abstracción	de Datos	de Control
Básica	tipos atómicos, variables	Asignación, goto
Estructurada	tipos estructurados	bucles, condicionales, subprogramas
Unitaria	módulos, paquetes	

La **abstracción** en los lenguajes de programación hace referencia principalmente a dos cuestiones: una de ellas trata acerca de la **Abstracción de Datos** y la otra trata de la **Abstracción del Control** la cual trata de las propiedades en cuanto a la estrategia de ejecución de un programa en una situacion determinada (ej: Bucles, sentencias condicionales, etc). Ambos tipos de abstracciones tienen una subclasificación, esta es: básica, estructurada y unitaria.

1.1.1 Abstracción de Datos

- **Abstracción de datos básica**: ésta se refiere a la representación interna de los tipos de datos atómicos que ofrece el lenguaje junto a las operaciones estándar, otro tipo de abstracción simbólica es la referenciación en memoria de una variable.
- **Abstracción de datos estructurada**: éste hace referecia al mecanismo de abstracción para las colecciones de datos, por ejemplo, una estructura típica son los arreglos (**Array**, tambien llamados vector).
- **Abstracción de datos unitaria**: ésta hace referencia a una agrupación como única unidad de datos y operación sobre estos datos. Ésto vendría a significar la introducción de del concepto de **encapsulado de datos** u ocultación de la información. Éstas, además, se asocian a menudo con los también denominados **TDA** (Tipos de Datos Abstractos) cuyo concepto se abarcó con profundidad en la materia **Algoritmos y Estructuras de Datos**

1.1.2 Abstracción de Control

- **Abstracción de control básica**: ésta se refiere a sentencias individuales que permiten modificar el control de flujo de la ejecución de un programa (ej: **sentencia de asignación** o el **GOTO** de los lenguajes que permitian la transferencia de control de una sentencia a otra parte del programa).
- **Abstracción de control estructurada**: estas agrupan sentencias más simples para crear una estructura con un propósito comun que permite gobernar la ejecución de un programa (ej: bucles, sentencias condicionales

if). Otro mecanismo util para la estructuración de un programa es **Subprograma**, típico concepto que se puede ver aplicado en los **procedimientos** utilizados en Pascal o los **métodos** utilizados en Java.

- **Abstracción de control unitaria**: ésta permite agrupar una coleccion de subprogramas como una unidad en sí misma e independiente del programa. Así se logra aislar parte del programa cuyo funcionamiento no es necesario conocer en detalle, se mejora la comprensión del mismo.

2 Definición de Lenguaje de Programación

Podríamos seguir a Louden [2] para la definición de lo que es un lenguaje de programación: "*Un lenguaje de programación es un sistema notacional para describir compuaciones en una forma legible tanto para el ordenarod omo para el programador*".

3 Definición de Lenguaje

Los lenguajes de programacion deben describirse de manera formal, completa y precisa. Esta descripcin ha de ser, ademas, independiente de la mquina y de la implementación. Para ello se utilizan habitualmente estándares aceptados universalmente.

Los elementos fundamentales para la definición de un lenguaje de programación son los siguientes:

- El **léxico** o conjunto de "palabras" o unidades léxicas que son las cadenas de caracteres significativas del lenguaje, tambien denominados tokens. También se conocen como unidades léxicas a los **identificadores**, los simbolos especiales de **operadores** y los **símbolos de puntuación**.

- la **sintaxis** o estructura que conlleva la descripción de los diferentes componentes del lenguaje y de sus combinaciones posibles. Para ello se utilizan las **gramáticas libres de contexto**, un estándar aceptado universalmente.

- la **semántica** expresa los efecos de la ejecución en un contexto determinado. A veces esta definición interactúa con otros significados del lenguaje, esto hace que sea la parte más compleja en la definición del lenguaje. Entre los sistemas de notación para definiciones semánticas formales se encuentran **semántica operacional**, **semántica denotacional** y la **semántica axiomática**.

Semántica Operacional: el significado de una construcción es una descripción de su ejecución en una máquina hipotética.

Semántica denotacional: asigna objetos matemáticos a cada componente del lenguaje para que modele su significado.

Semántica axiomática: modela el significado con un conjunto de axiomas que describen a sus componentes junto con algún tipo de inferencia del significado.

4 Diseño del Lenguaje

El reto de un lenguaje es lograr la potencia, expresividad y comprensión que requiere la legibilidad del programador, conservando la precisión y simplicidad necesarias para su traduccion al lenguaje máquina.

La legibilidad de los programadores es proporcional a las capacidades de abstracción del lenguaje, como por ejemplo, se abstraen los datos y los controles ¡vistos anteriormente¿. *Es seguro decir que uno de los objetivos, si no Él objetivo de la abstraccion en el diseño de los lenguajes de programación es el manejo de la* ***complejidad***

Aspectos de diseño en los lenguajes:

4.1 Efificiencia

El diseño debe permitir al traductor la generacion de codigo **ejecutable eficiente**, también conocido como **optimizabilidad**. La eficiencia se organiza en tres principios: eficiencia de traducción, de implementación y de programación.

- La **eficiencia de traducción** estipula que el diseño del lenguaje debe permitir el desarrollo de un traductor eficiene y de un tamaño razonable. Por ejemplo: Pascal o C, por restricciones de **diseño** exigen que las variables se declaren antes de su uso, permitiendo así que el compilador de una sola pasada. Logrando ser eficiente en el uso de los recursos a la hora de traducción.

- La **eficiencia de implementación** es la eficiencia con la que se puede escribir un traductor, que a su vez depende de la complejidad del lenguaje.

- la **eficiencia de la programación** está relacionada con la rapidez y la facilidad para escribir programas o **capacidad expresiva** del lenguaje. Esta capacidad expresiva se refiere a la facilidad para escribir procesos complejos de forma que el programador relacione de manera sencilla su idea con el código. Este es un aspecto relacionado con la potencia y la generalidad de los mecanismos de abstracción y la sintaxis.

4.2 Regularidad

Este principio hace referencia al comportamiento de las características del lenguaje. Se subdivide en tres propiedades: la generalidad, la ortogonalidad y la uniformidad, si se viola una de ellas, el lenguaje ya se puede clasificar como irregular.

- la **generalidad** se consige cuando el uso y la disponibilidad de los constructores no están sujetas a casos especiales y cuando el lenguaje incluye

solo a los constructores necesarios y el resto se obtienen por combinacion de constructores realacionados.

- la **ortogonalidad** (también llamada dependencia) ocurre cuando los constructores del lenguaje pueden admitir combinaciones significativas en ellas, la interaccion de los constructores o con el contexto, no provocan restrcciones ni comportamientos inesperados.

- la **uniformidad** se refiere a que *lo similar se ve similar y lo diferente, diferente* lo que implica la consistencia entre la apariencia y el comportamiento de los constructores.

4.3 Principios adicionales

En los siguientes apartados se presentan cuestiones adicionales a la legibilidad, eficiencia y regularidad del diseõ de lenguajes de programación, que ayudan a definir un buen diseño o a elegir mejor el lenguaje para un escenario concreto.

4.3.1 Simplicidad

Este se refiere a que cada concepto del lenguaje se represente de una forma única y legible y semánticamente que contiene el menos número posible de conceptos y estructuras con reglas sencillas de combinacion.

4.3.2 Expresividad

Es la facilidad con la que un lenguaje de programación permite expresar procesos y estructuras complejas. Uno de los mecanismos más expresivos es la recursividad. Una expresividad muy alta entra en conflicto con la simplicidad ya que el ambiente de ejecución es complejo.

4.3.3 Extensibilidad

Es la propiedad asociada con la posibilidad de añadir nuevas características a un lenguaje, como nuevos tipos de datos o nuevas funciones a la biblioteca. O tambien añadir palabras clave y constructores al traductor.

Otro aspecto importante de la extensibilidad es la **modularidad**, es decir, la capacidad de disponer de bibliotecas y agregar nuevas. Esta también se corresponde con la posibilidad de dividir un programa en partes independientes (denominados módulos o paquetes) que puedan enlazarse para su ejecucion.

4.3.4 Capacidad de restricción

Ésta se refiere a la posibilidad de que un programador utilice solo un subconjuto de constructores mínimo y por lo tanto solo necesite un conocimiento parcial del lenguaje. Esto ofrece dos venajas: el programador no necesita aprender todo el lenguaje, y por otra parte, el traductor puede implementar solo un subconjunto

determinado porque la implementación para todo el lenguaje sea muy costosa e innecesaria.

Un aspecto relacionado a la capacidad de restricción es la **eficiencia**: porque un programa no utilice ciertas características del lenguaje, su ejecución no debe ser mas ineficiente.

Otro aspecto a tener en cuenta que esta relacionado con la capacidad de restricción es el **desarrollo incremental** del mismo. Por ejemplo, Java ha evolucionado con respecto a su versión inicial, y en cada nueva versión se han incluido características que no estaban en versiones anteriores.

4.3.5 Consistencia entre la notación y las convenciones

Un lenguaje debe incorporar notaciones y cualquier otra característica que ya se hayan convertido en estándares como lo son el concepto de programa, funciones y variable. El que estos aspectos queden perfectamente reconocibles, facilita a los programadores experimentados el uso del lenguaje, de igual manera podemos mencionar las notaciones relacionadas a las matemáticas como los operadores aritméticos (+, -, etc).

4.3.6 Portabilidad

Esta se consigue si la definición del lenguaje de programación es independiente de una máquina en particular. Normalmente, los lenguajes interpretados o aquellos cuya ejecución se delega en una máquina virtual lo son.

4.3.7 Seguridad

Este pretende evitar los errores de programación, y permitir su descubrimiento. por lo tanto la seguridad está muy relacionada con la fiabilidad y la presición. Sin embargo, la seguridad compromete a la capacidad expresiva del lenguaje, pues se apoya en que el programador especifíque todo lo que sea posible en el código para evitar errores.

4.3.8 Interoperabilidad

Esta se refiere a la facilidad que tienen diferente tipos de ordenadores, redes, sistemas operativos, aplicaciones o sistemas en general para trabajar conjuntamente de manera efectiva, sin comunicaciones previas, para intercambiar información útil y con sentido.

Hay **interoperabilidad semántica** cuando los sistemas intercambian mensajes entre sí interpretando el significado y el contexto de los datos, **interoperabilidad sintáctica**, cuando un sistema lee datos de otro, mediante una representación compatible (metalenguajes como XML), e **interoperabilidad estructural** cuando los sistemas pueden comunicarse e interactuar en ambientes heterogéneos.

Para que esto sea posible entre aplicaciones desarrolladas en diferentes lenguajes de programación y ejecutadas en cualquier plataforma, se debe establecer la

representación de la información a intercambiar y además utilizar un protocolo de comunicación. Existen estándares para intercambiar datos entre aplicaciones.

5 Paradigmas

5.1 Paradigmas de programación

Los paradigmas de programación mencionados a continuación fueron consultados desde lo citado en el aprartado ***Referencias***.

5.1.1 Programación orientada a objetos [2]

Este se basa en la idea de que un objeto se puede describir como una colección de posiciones de memoria junto con todas las operaciones que pueden cambiar los valores de dichas posiciones. Estas entidades OBJETOS se agrupan en clases que representan a todos los que tienen las mismas propiedades. Estas clases se definen mediante declaraciones parecidas a las de los objetos estructurados en C o Pascal, estos objetos se crean a partir de la instanciación de la clase.

5.1.2 Programación funcional [2]

La computación en el paradigma funcional se fundamenta en la evaluación de funciones o en la aplicación de funciones a valores conocidos, por lo que tambien se denominan **lengujes iperativos**. El mecanismo básico es la evaluación de **funciones**, con las siguientes características:

- Transferencia de valores como parámetros de las funciones que se evalúan.
- La generación de resultados en forma de valores devueltos por las funciones.

5.1.3 Programación lógica [2]

En éste paradigma un programa está formado por un conjunto de sentencias que describen lo que es *verdad* o *conocido* con respecto a un problema, en vez de indicar la secuencia de psos que llevan al resultado. No necesita abstracciones de control condicionales ni de ciclos ya que el control lo aporta el modelo de inferencia lógica que subyace.

Referencias

[1] Fernando López Ostenero, Ana María García Serrano *Teoría de los Lenguajes de Programación*. Editorial Universitaria Ramón Areces.

[2] Louden, K. Enneth C *Lenguajes de Programación principios y práctica. Segunda edición*. Thomson, 2004.

[3] Michael L. Scott *Programming Language Pragmatics*. Department of Computer Science, University of Rochester.