SUBSECRETARÍA DE EDUCACIÓN SUPERIOR
DIRECCIÓN GENERAL DE EDUCACIÓN
SUPERIOR TECNOLÓGICA
INSTITUTO TECNOLÓGICO DE IGUALA

SECRETARÍA DE
EDUCACIÓN PÚBLICA

SEP

Instituto Tecnológico de Iguala

Programación de sistemas

- Introducción al diseño de los lenguajes de programación
- condiciones que determinan la funcionalidad del lenguaje

Ingeniería en sistemas computacionales

- **Ing. María del Carmen Aristegui Peralta**
- **Hugo armando Ramírez Cleto**

Iguala Gro. 23 de septiembre 2010

➢ Introducción al diseño de los lenguajes de programación

❖ Comunicación humana

La comunicación humana implica un sistema complejo de códigos interdependientes:

- Verbales: compuesto por el código oral y el escrito.
- No verbales: compuestos por mímica, mirada, movimientos, ropa, aspecto personal...
- Señales paralingüísticas: Volumen y tono de la voz, pausa y silencio.

❖ Prevención y detección de errores

Tener una serie de defensas tal que si un error no es detectado por uno, este probablemente sea detectado por otro.
Los errores deben ser detectados por el compilador, si un mecanismo no es capaz de detectar un error es necesario implementar otro que lo detecte, pero nunca ignorarlo.

A continuación se presentan prevención y tolerancia de errores y fallos

❖ Prevención de errores

Control sobre los apuntadores a NULL.
Prevención y tolerancia de fallos

• Hay dos formas de aumentar la fiabilidad de un sistema:
Prevención de fallos: Se trata de evitar que se introduzcan fallos en el sistema antes de que entre en funcionamiento

Prevención de fallos
Se realiza en dos etapas:

• Evitación de fallos: Se trata de impedir que se introduzcan fallos durante la construcción del sistema
• Eliminación de fallos: Consiste en encontrar y eliminar los fallos que se producen en el sistema una vez construido

Tolerancia de fallos: Se trata de conseguir que el sistema continúe funcionando aunque produzcan fallos
En ambos casos el objetivo es desarrollar sistemas con modos de fallo bien definidos.

❖ Detección de errores

- Por el entorno de ejecución

hardware (p.ej.. instrucción ilegal)

núcleo o sistema operativo (p.ej. puntero nulo)

- Por el software de aplicación

Duplicación (redundancia con dos versiones)
Comprobaciones de tiempo
Inversión de funciones
Códigos detectores de error
Validación de estado
Validación estructural

❖ Usabilidad

Es la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto. La usabilidad también puede referirse al estudio de los principios que hay tras la eficacia percibida de un objeto.

En interacción persona-ordenador, la usabilidad se refiere a la claridad y la elegancia con que se diseña la interacción con un programa de ordenador o un sitio web. El término también se usa a menudo en el contexto de productos como la electrónica de consumo o en áreas de comunicación, y en objetos que transmiten conocimiento (por ejemplo, un libro de recetas o un documento de ayuda en línea). También puede referirse al diseño eficiente de objetos mecánicos como, por ejemplo, un manubrio o un martillo.

❖ Beneficios de la usabilidad

Entre los principales beneficios se encuentran:

- Reducción de los costes de aprendizaje.
- Disminución de los costes de asistencia y ayuda al usuario.
- Optimización de los costes de diseño, rediseño y mantenimiento.
- Aumento de la tasa de conversión de visitantes a clientes de un sitio web.
- Mejora la imagen y el prestigio.
- Mejora la calidad de vida de los usuarios, ya que reduce su estrés, incrementa la satisfacción y la productividad.

❖ Eficiencia

Capacidad para el aprovechamiento óptimo de los recursos que emplea. Los lenguajes OOP arrastraron en un principio la reputación de ser ineficaces. Esto se debía en gran medida a que los primeros lenguajes (como Smalltalk) eran interpretados y no compilados.

La existencia de compiladores permite a los desarrolladores ganar rapidez. Actualmente, usando un buen lenguaje orientado a objetos como C++, Java, etc. Junto con las librerías apropiadas para la realización de un programa, puede que se ejecute más rápidamente que el mismo programa compilado con un lenguaje procedural

❖ Compilabilidad

El programa escrito en un lenguaje de programación (comprensible por el ser humano, aunque se suelen corresponder con lenguajes formales descritos por gramáticas independientes del contexto) no puede ejecutarlo directamente una computadora. La opción más común es compilar el programa obteniendo un módulo objeto, aunque también puede ejecutarse a través de un intérprete informático.
El código fuente del programa se debe someter a un proceso de traducción para convertirse en lenguaje máquina, interpretable por el procesador. A este proceso se le llama *compilación*.

❖ Independencia de la maquina

Los programas Windows son independientes de la máquina en la que se ejecutan (o al menos deberían serlo), el acceso a los dispositivos físicos se hace a través de interfaces, y nunca se accede directamente a ellos. Esta es una de las principales ventajas para el programador, ya que no hay que preocuparse por el modelo de tarjeta gráfica o de impresora, la aplicación funcionará con todas, y será el sistema operativo el que se encargue de que así sea.

A la hora de explotar un gran número de bases de datos de diferentes editores nos encontramos ante un doble problema. Por una parte, la citada falta de homogeneidad de los sistemas informáticos de los usuarios; por otra parte, cada una de las bases de datos suele tener su propio programa de consulta, de modo que nos encontramos multitud de programas diferentes que deberán conocer los usuarios. Dichos programas están en su mayor parte diseñados para ordenadores tipo PC con sistema operativo Windows. Teniendo en cuenta todo lo anterior, podemos ver que el sistema ha de permitir acceder a multitud de máquinas diferentes a una serie de programas en muchos casos incompatibles con ellas.

Los servidores de aplicaciones son la base de programas informáticos diseñados para ser ejecutados desde ordenadores personales a través de Navegadores de Internet convencionales.

Con ello se consigue independencia de la máquina (los programas funcionan en cualquier ordenador), independencia de ubicación (es posible utilizar los programas desde cualquier lugar) y una administración ligera y centralizada (mantenimiento cero de los programas de los ordenadores de los usuarios al residir éstos en el servidor).

❖ Simplicidad

Se basa en el entendimiento profundo del asunto que se quiere transmitir y en la capacidad de hacerlo de una forma clara y concisa

La simplicidad tiene algunas ventajas notables

- Proximidad (Approachability): Los diseños sencillos son más fáciles de entender y favorecen el uso inmediato y la exploración exhaustiva de los recursos del diseño.
- Reconocibilidad (Recognizability): Son más fácilmente reconocibles y asimilables ya que presentan menos información visual superflua.
- Inmediatez (Immediacy): Los diseños sencillos tienen un impacto mayor precisamente porque su facilidad de comprensión los hacen inmediatamente reconocibles con un esfuerzo consciente mínimo.
- Usabilidad: Por todo lo anterior suelen ser también los más fáciles de usar.

❖ Uniformidad

Es la representación de los objetos lleva implica tanto el análisis como el diseño y la codificación de los mismos.

❖ Ortogonalidad

Dos características de un lenguaje son ortogonales si pueden ser comprendidas y combinadas de forma independiente. Cuando las características del lenguaje son ortogonales, el lenguaje es más sencillo de comprender, porque hay menos situaciones excepcionales a memorizar. La Ortogonalidad ofrece la posibilidad de combinar características de todas las formas posibles (sin excepciones). La falta deDefinición de un lenguaje Ortogonalidad puede suponer la enumeración de situaciones excepcionales o la aparición de incoherencias. Un ejemplo de falta de Ortogonalidad es la limitación que impone Pascal para que una función devuelva determinados tipos de valores.

❖ Generalización

La generalización dice que algo similar también es correcto, pero es difícil de implementar. Hay que especializar para facilitar la implementación sin perder la utilidad del lenguaje

❖ Especialización

Hay que especializar para facilitar la implementación sin perder la utilidad del lenguaje

❖ CONDICIONES QUE DETERMINAN LA FUNCIONALIDAD DEL LENGUAJE

✓ Microestructuras

Una vez delimitadas las categorías conceptuales y sus esquemas categoriales, mostramos una definición terminográfica basada en la estructuración conceptual relacional de cada una de las categorías. Así, se lleva a cabo una segmentación de la información de la definición en función de las relaciones y atributos que caracterizan un esquema categorial determinado.

A continuación, para cada categoría se indica el término que va a ser definido y se detalla, entre corchetes, la arquitectura relacional base prototípica de cada una de las categorías conceptuales, así como los conceptos específicos que han sido activados por el término en cuestión, a la derecha. En el recuadro inferior se presenta la definición propiamente dicha, donde se ha formalizado en cada segmento oracional cada una de las relaciones y atributos conceptuales.

❖ Estructura de las expresiones

Una expresión es una serie de variables, operadores y llamadas a métodos (construida conforme a la sintaxis del lenguaje) que se evalúa a un único valor.
Esta expresión en particular se evalúa al valor de contador antes de que la operación ocurra.

El tipo de datos del valor devuelto por una expresión depende de los elementos utilizados en la expresión. La expresión contador ++ devuelve un entero porque ++ devuelve un valor del mismo tipo que su operando y contador es un entero. Otras expresiones devuelven valores booleanos, cadenas, etc.

❖ Estructuras de datos

Es una forma de organizar un conjunto de datos elementales con el objetivo de facilitar su manipulación. Un dato elemental es la mínima información que se tiene en un sistema.

Una estructura de datos define la organización e interrelación de éstos y un conjunto de operaciones que se pueden realizar sobre ellos. Las operaciones básicas son:

- Alta, adicionar un nuevo valor a la estructura.
- Baja, borrar un valor de la estructura.

- Búsqueda, encontrar un determinado valor en la estructura para realizar una operación con este valor, en forma secuencial o binario (siempre y cuando los datos estén ordenados).

Otras operaciones que se pueden realizar son:

- Ordenamiento, de los elementos pertenecientes a la estructura.
- Apareo, dadas dos estructuras originar una nueva ordenada y que contenga a las apareadas.

❖ Estructuras de control

Todas las estructuras de control tienen un único punto de entrada y un único punto de salida. Las estructuras de control se puede clasificar en: secuenciales, iterativas y de control avanzadas. Esto es una de las cosas que permite que la programación se rija por los principios de la programación estructurada.

Los lenguajes de programación modernos tienen estructuras de control similares. Básicamente lo que varía entre las estructuras de control de los diferentes lenguajes es su sintaxis, cada lenguaje tiene una sintaxis propia para expresar la estructura.

❖ Tipos de estructura de control

Tales características permiten desarrollar de forma muy flexible todo tipo de algoritmos aún cuando sólo existen tres tipos fundamentales de estructuras de control:

- Secuencial.
- Alternativa.
- Repetitiva.

❖ Estructura de un compilador

La estructura de un compilador, esta dividida en cuatro grandes módulos, cada uno independiente del otro, se podría decir que un compilador esta formado por cuatros módulos mas a su vez.

- **Preprocesador**, es el encargado de transformar el código fuente de entrada original en el código fuente puro.

- **Compilación** que recibe el código fuente puro, este es él modulo principal de un compilador, pues si ocurriera algún error en esta etapa el compilador no podría avanzar.

- **Ensamblado**, este modulo no es ni más mi menos que otro compilador pues recibe un código fuente de entrada escrito en ensamblador, y produce otro código de salida, llamado código binario no enlazado.

- **Enlazado** del código de fuente de entrada (código maquina relocalizable) con las librerías que necesita, como así también de proveer al código de las rutinas necesarias para poder ejecutarse y cargarse a la hora de llamarlo para su ejecución, modifica las direcciones relocalizables y ubica los datos en las posiciones apropiadas de la memoria.

❖ Estructuras para entrada y salida

Entrada: Son todos aquellos que permiten la entrada de datos a un computador. Entre estos encontramos: el teclado, el ratón, el escáner, el micrófono, la cámara web, el capturador de huella y firma digitales o lápices ópticos, etc.
Salida: Son todos aquellos que permiten mostrar la información procesada por el computador. Entre estos encontramos: la pantalla, la impresora, los altavoces, etc.
Se supone que los datos que se utilizan como referencia para crear el conjunto de entrada-salida representan de manera significativa al sistema que se pretende identificar, lo cual está vinculado a la estructura de identificación que se seleccione. Existen diversas combinaciones para configurar los datos que constituirán la base para la creación de una estructura de identificación. Algunos de los factores a tener en consideración son:

- Incorporación de retraso a los datos representativos de las variables Existencia de retardo en el sistema o proceso objeto de la medición.
- Si se desea predecir variables en el futuro (d).- Si se considerará el error en la estimación como variable de entrada.
- Tipo de estructura del modelo seleccionada: paralelo, serie-paralelo o un híbrido entre ambas.

Bibliografía

1. http://www.monografias.com/trabajos14/progorie/progorie.shtml
2. http://es.wikipedia.org/wiki/Estructura_de_datos
3. http://es.wikipedia.org/wiki/Estructuras_de_control
4. http://es.wikipedia.org/wiki/Usabilidad
5. http://html.rincondelvago.com/comunicacion-humana.html
6. http://www.infovis.net/printMag.php?num=35&lang=1
7. http://es.wikipedia.org/wiki/Ortogonalidad_%28Computaci%C3%B3n%29