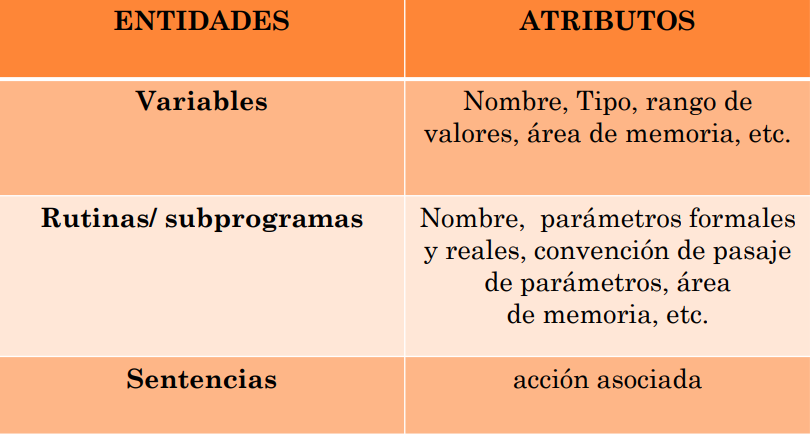
Clase 3 – Semantica Operacional

Semantica operacional: Esta permite describir el significado preciso de un programa y verificar el resultado final de la ejecucion de un programa. Es fundamental para diversos aspectos del proceso de desarrollo de sw, como el diseño de lenguajes de programacion, la verificacion de programas y la comprension de como se ejecutan los programas en un nivel mas bajo.



Repositor descriptor: Lugar donde se almacena la info de los atributos.

.El asociar cada entidad a sus atributos es un concepto central en la definicion de la semantica de los lenguajes de programacion.

Concepto de ligadura (binding): La ligadura es el momento en el que el atributo se asocia con un valor determinado. El momento de la ligadura cuando toma el valor se define (binding time) y es distinto en los lenguajes. Tipos de ligadura:

.Ligadura estatica: Se establece antes de la ejecucion y no se puede modificar.

.Ligadura dinamica: Se establece durante la ejecucion y si puede modificarse durante la ejecucion de acuerdo a una regla especifica del lenguaje. Como excepcion estan las constantes.

Los atributos pueden ligarse en distintos momentos:

1.definicion del lenguaje: La forma de las sentencias, la estructura del programa y los nombres de los tipos predefinidos.

2.implementacion del lenguaje: Set de valores y su representacion numerica ademas de sus operaciones

3.compilacion: asignacion/redefinicion del tipo a las variables

4.ejecucion: las variables se enlazan con sus valores y se enlazan con su lugar de almacenamiento.

Variables:

Atributos de una variable:

Nombre: String de caracteres que se usa para referenciar a la variable (id)

Alcance: es el rango de instrucciones en el que se conoce el nombre, es visible, y puede ser referenciada

Tipo: es el tipo de variables definidas, tiene asociadas rango de valores y conjunto de operaciones permitidas

L-valor: es el lugar de memoria asociado con la variable, esta asociado al tiempo de vida(las var se alocan y desalocan)

R-valor: es el valor codificado almacenado en la ubicación de la variable.

Una variable es una 5-TUPLA.

1.NOMBRE:

.El nombre es introducido por una sentencia de declaracion.

.La longitud maxima según el lenguaje (este se define en la etapa de definicion del lenguaje).

.los caracteres aceptados en el nombre (conectores).

.Si son sensitivos a mayusculas.

Palabra clave: son palabras propias del lenguaje, tienen un signficado especial solo en contextos particulares y se pueden utilizar como id en otros contextos.

Palabra reservada: es aquella palabra clave que no puedo utilizar para asignar a un id, esto depende de cada lenguaje. Ej: en Java const no tiene signficado pero no puede ser usada como id.

2. ALCANCE:

.El alcance se puede asociar a su visibilidad. Las instrucciones del programa pueden manipular las var a traves de su nombre dentro de su alcance. Afuera de ese alcance son invisibles.

.Los lenguajes usan diferentas reglas para ligar el nombre de una var a su alcance:

1.ligadura por alcance estatico: llamado alcance lexico. Se define el alcance en terminos de la estructura lexica del programa. Puede ligarse estaticamente a una declaracion de variables examinando el texto del programa, sin necesidad de ejecutarlo. La mayoria de los lenguajes adoptan reglas de ligadura de alcance estatico.

2.ligadura por alcance dinamico: define el alcance del nombre de la variable en donde es conocida en terminos de la ejecucion del programa. Cada declaracion de variable extiende su efecto sobre todas las instrucciones ejecutadas posteriormente, hasta que una nueva declaracion de la variable (con el mismo nombre) sea encontrada durante la ejecucion.

Despues de los ejemplos: Entiendo que el alcance estatico toma los valores “que conoce” osea en un caso donde tenes varias variables con un mismo nombre y tenes un variable definida en el main y el resto en metodos, al hacer una recursion si en el “metodo final” osea el metodo que realiza el calculo no tiene declarada esa variable utiliza la variable declarada en el main “porque la conoce” en cambio con el alcance dinamico si en el main invoco a un metodo y este invoca a otro sucesivamente el metodo final va a usar la variable del metodo “mas cercano” onda si paso por ese metodo com que ya lo conoce por decir de algun modo entonces al conocerlo usa el valor de esa var y no la del main dado que la del metodo es “mas cercana”. Con la definicion se entiende igual.

.ADA tiene una reglas especiales con el tema del alcance estatico. Pag 30.

.En Python: El tipo de una variable se infiere automaticamente en funcion del valor que se le asigna. Al ser fuertemente tipado no permite operaciones sobre tipos distintos.

Clasificacion de variables por su alcance:

1. Global: Son todas las referencias a variables creadas en el programa principal.

2. Local: Son todas las referencias a variables creadas dentro de una unidad (programa o subprograma).

3. No local: Son todas las referencias a variables que se utilizan dentro de un subprograma pero que no han sido creadas en la unidad. Son externas al subprograma.

.Es importante conocer las reglas de cada lenguaje.

3. TIPO: Se define el tipo de una variable como la especificacion del conjunto de valores posibles que se pueden asociar a la variable y del conjunto de operaciones permitidas sobre ellas (crear, acceder, modificar).

.Antes de referenciar una variable hay que ligarle un tipo. Una var de un tipo dado es una instancia de ese tipo.

.Ligar el tipo(conjunto de valores posibles y conjunto de operaciones asociadas) ayuda a proteger a las var de operaciones no permitidas, realizar chequeos de tipos y verificar el uso correcto de las var(cada lenguaje tiene sus reglas de combinaciones de tipos). Ademas ayuda a que el compilador o interprete detecte errores en forma temprana y a dar confiabilidad del codigo.

/\* ¿HACE FALTA SABER LAS ESPECIFICACIONES DE CADA TIPO? \*/

1. Tipos predefinidos: Son los tipos base que estan descriptos en la definicion del lenguaje (int, char, etc). Cada uno tiene valores y operaciones asociadas. Los valores se ligan en la implementacion a representacion de maquina según la arquitectura.

2. Tipos definidos por el usuario: Permiten al programador con la declaracion de tipos definir nuevos tipos a partir de los tipos base y constructores predefinidos. Esa declaracion establece el binding en tiempo de traduccion y heredan todas las operaciones de la representacion de la estructura de datos base. Ademas permite al programador crear abstracciones, encapsular la logica y datos, reutilizar codigo y mejorar la claridad y legibilidad del codigo. Son fundamentales para el desarrollo de programas complejos y para mantener un codigo organizado y mantenible.

.Hay que saber que hace cada lenguaje, que definen en su etapa de diseño y como lo hacen en la etapa de implementacion.

3. Tipos de Datos Abstractos (TAD): Son abstractos porque quien lo utiliza no necesita conocer los detalles de la representacion interna ni como estan implementadas las operaciones. Hay ocultamiento.

.No todos los lenguajes soportan este tipo.

.Se les debe asignar un nombre que lo identifique.

.Se realiza la asociacion del nuevo tipo con un set de operaciones que pueden ser usadas sobre sus instancias para manipular los objetos y las operaciones son descriptas como un set de rutinas que establece el programador y se especifican en la declaracion del nuevo tipo.

.Cada TAD define un conjunto de operaciones permitidas, pero oculta los detalles de implementacion interna. No hay una ligadura por defecto, el programador debe especificar la representacion y las operaciones.

.TAD comunes: Listas, colas, pilas, arboles, grafos, etc. (TAD se vera en mas detalle).

Momentos de ligadura Variable-Tipo:

.La ligadura entre la var y su tipo se especifica en la declaracion. Se liga en compilacion y no puede cambiarse en ejecucion. De esta forma hay proteccion. El chequeo de tipo tambien sera estatico.

1. Estatico: En traduccion/compilacion.

.Declarar previamente las var de un tipo hace que sean automaticamente protegidas de una operación ilegal. El compilador puede detectar “violaciones de la semantica estatica” relacionadas a var y sus tipos. Ademas realizar el “chequeo de tipos estatico” antes de la ejecucion contribuye a la deteccion temprana de errores y a mejorar la confiabilidad del programa.

La ligadura puede ser realizada en forma:

A. Explicita: La ligadura se establece mediante una sentencia de declaracion. La ventaja de esta reside en la claridad de los programas y en una mayor confiabilidad, porque cosas como errores ortograficos en nombres de var pueden detectarse en tiempo de traduccion/compilacion.

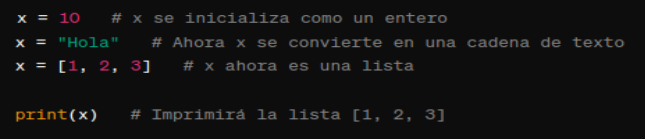
B. Implicito: Se deduce por “reglas propias del lenguaje”, esto ocurre sin que el programador tenga que especificar explicitamente el tipo de datos de la var. Lo define cada lenguaje.

C. Inferido: No requieren que el tipo de la var sea declarado. El tipo se deduce/infiere automaticamente de los tipos de sus componentes. Se basa en el contexto del codigo y en el valor asignado a la var de una asignacion. El enlace se realiza en etapa de traduccion. Si no puede inferir da error de compilacion.

.En general aplica en lenguajes funcionales

2. Dinamico: En ejecucion.

.El tipo se liga a la var en ejecucion y puede modificarse. Cambia cuando se le asigna un valor mediante una sentencia de asignacion (no declaracion).



Problemas:

.El costo de implementacion de la ligadura dinamica es mayor, dado que utiliza mucho tiempo de ejecucion en la comprobacion de tipos, el mantenimiento de tablas de descriptores asociado a cada var en el que se almacena el tipo actual y el cambio en el tamaño de la memoria asociada a la var.

.Ademas de necesitar un chequeo dinamico y tener menor legibilidad.

.En general aplica en lenguajes interpretados.

4. L-VALOR:

.Es la direccion del area de memoria ligada a la variable durante la ejecucion. Las instrucciones de un programa acceden a la variable por su L-valor.

.Las variables se alocan en un area de memoria. Ese area de memoria debe ser ligada a la variable en algun momento. En esa area de memoria se almacenara su valor al cual necesito llegar. Para ello uso el L-valor.

.El tiempo de vida es el periodo de tiempo en que existe la ligadura. Es el tiempo en que esta alocada la variable en memoria y en el cual el binding existe. Este va desde que se solicita hasta que se libera.

.Alocacion de memoria: es el momento en que se reserva la memoria para una variable.

Tipos de momentos de alocacion de memoria: (Este tipo dependera del lenguaje de programacion)

1. Estatico: se hace en compilacion, cuando se carga el programa en memoria en zona de datos y perdura hasta el fin de la ejecucion.

2. Dinamico: se hace en tiempo de ejecucion.

A. Automatica: cuando aparece una declaracion de una var en la ejecucion.

B. Explicita: requerida por el programador con una sentencia de creacion, a traves de algun constructor.

3. Persistente: son los objetos persistentes que existen en el entorno en el cual un programa es ejecutado, su tiempo de vida no tiene relacion con el tiempo de ejecucion del programa. Persisten mas alla de la memoria.

(Todo se vera mas adelante en detalle).

5. R-valor:

.Es el valor codificado almacenado en la ubicación asociada a la variable (L-valor). La codificacion se interpreta con el tipo de la variable.

Ejemplo: R-valor: 01110011

.Intepreta un nro entero si la variable es tipo int o interpreta una cadena si la variable es tipo char.

Objeto: (L-valor, R-valor)(direccion memoria, valor).

Momentos de ligadura – variable a valor:

.El valor (r-valor) puede cambiar durante la ejecucion con una asignacion. Exceptuando que sea una constante en este caso su valor se congela, estas no son permitidas por algunos lenguajes.

.Binding Dinamico: este varia según los lenguajes. Los lenguajes de programacion funcional y logica pueden vincular un valor mediante el proceso de evaluacion, pero una vez establecida la vinculacion, no se puede cambiar durante el tiempo de vida de la variable.

.Bindign time: tambien varia según los lenguajes.

.En Pascal el valor que proporciona una expersion debe evaluarse en tiempo de compilacion y en otros como C y ADA, el valor se puede dar como una expresion que involucra otras variables y constantes, en consencuencia, la ligadura solo puede establecerse en tiempo de ejecucion, cuando la var es creada.

Inicializacion de una variable: Los lenguajes y sus versiones implementan diversas estrategias de inicializacion:

1. Inicializacion por defecto: enteros se inicializan en 0, caracteres en blanco, y funciones en void, etc.

2. Inicializacion en la declaracion

Si no es inicializada.

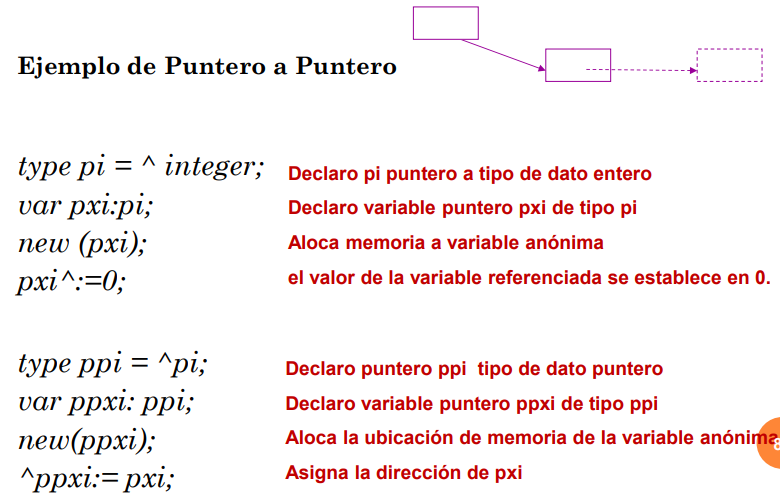
3. Estrategia ignorar el problema: Toma como valor inicial lo que hay en memoria (la cadena de bits asociados al area de almacenamiento), esto puede llevar a errores y requiere chequeos adicionales.

Consideraciones adicionales:

Variables anonimas y referencias – punteros:

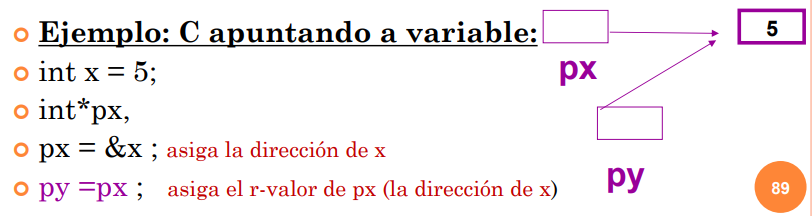
.Algunos lenguajes permiten que variables sin nombre sean accedidas por el r-valor de otra variable. Ese r-valor se denomina referencia o puntero a la variable. La referencia puede ser al r-valor de una var nombrada o al de una referenciada. El r-valor de una var sera la referencia al L-valor de otra var.

.Puntero: var que sirve para señalar la posicion de la memoria en la que se encuentra otro dato almacenado como valor, con la direccion de ese dato.

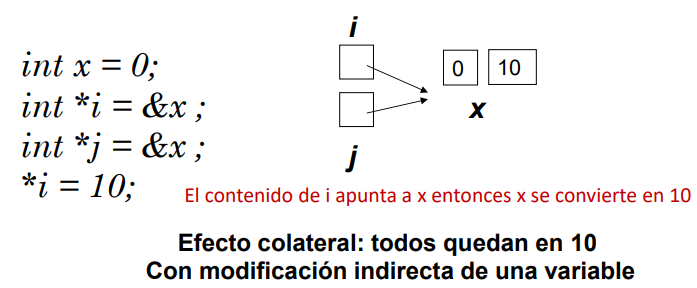


.Alias: se da si hay variables que comparten un objeto en el mismo entorno de referencia. Sus camino de acceso conducen al mismo objeto. Por lo tanto el objeto compartido modificado por uno se modifica para todos los caminos. Estos pueden traer problemas y es recomendable no usarlos.

Alias: distintos nombres -> 1 entidad



Ej de posible efecto colateral de usar Alias (sirve para entender como funciona):



Sobrecarga: 1 nombre -> distintas entidades

.Sobrecarga: es una caracteristica de algunos lenguajes de programacion que permiten definir comportamientos personalizados para operadores, metodos, funciones. Osea, un mismo nombre que realiza algo distinto según el contexto.

.Un nombre esta sobrecargado si en un momento referencia mas de una entidad. Esto debe estar permitido por el lenguaje y debe haber suficiente info para permitir establecer la ligadura univocamente.