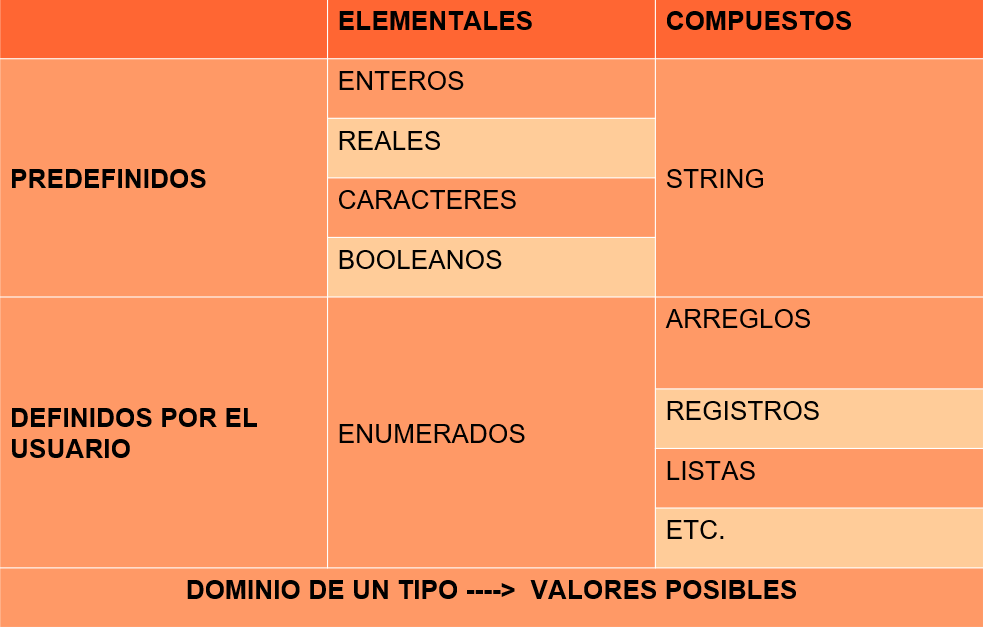
Clase 7 – Tipos de datos

Tipo de dato: se puede definir como un conjunto de valores y un conjunto de operaciones que se pueden utilizar para manipularlos.

HISTORIA



.Cualquier lenguaje esta equipado con un conjunto finito de tipos predefinidos, que normalmente reflejan el comportamiento del hw subyacente. Ademas, los lenguajes permiten al programador especificar agrupaciones de objetos de datos elementales y de forma recursiva, agregaciones de agregados. Esto se logra mediante la prestacion de una serie de constructores que permiten definir lo que se denomina tipo de dato definido por el usuario.

Tipos predefinidos:

.La ventaja de estos son: invisibilidad de la representacion, verificacion estatica, desambiguar operadores y permiten un control de precision.

.numeros (enteros y reales), caracteres y boolean.

.Que un conjunto de valores de un tipo sea definido por la implementacion del lenguaje significa que sera seleccionado por el compilador, mientras que si el tipo es definido por el lenguaje sera definido en su definicion.

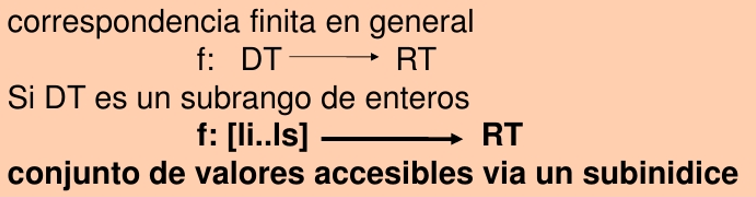
Tipos definidos por el usuario:

.La legibilidad depende de la eleccion apropiada de los nombres. Estos solo se cambien en la definicion (modificabilidad) y se usa la cantidad de veces necesarias (factorizacion).

Compuestos – constructores:

.Producto cartesiano: Entiendo que hace referencia al uso del producto cartesiano para la creacion de un nuevo tipo, juntando elementos de distintos tipos o algo asi. Es raro.

.Correspondencia finita: es una funcion de un conjunto finito de valores de un tipo de dominio DT en valores de un tipo del dominio RT.



.Union y union discriminada: la union/ union discriminada de dos o mas tipos define un tipo como la disjuncion de los tipos dados. Esto permite manipular diferentes tipos en distinto momento de la ejecucion, ademas nos permite tener un chequeo dinamico. La declaracion es muy similar a la del producto cartesiano. La diferencia es que sus campos son mutuamente excluyentes, los valores del tipo definido deben ser tratados en forma distinta si son de un tipo o del otro (repasar).

Union discriminada agrega un discriminante para indicar la opcion elegida. Ej: si tenemos la union discriminada de dos conjuntos S y T, y aplicamos el discriminante a un elemento e perteneciente a la union discriminada devolvera S o T.

El elemento e debe manipularse de acuero al valor del discriminante. En estos dos casos el chequeo de tipos debe hacerse en ejecucion. La union discriminada se puede manejar en forma segura consultando el discriminante antes de utilizar el valor del elemento.

Problemas:

.El discriminante y las variantes pueden manejarse independientemente uno de otros.

.La implementacion del lenguaje puede ignorar los chequeos.

.Puede omitirse el discriminante, con lo caul aunque se quisiera no se puede chequear.

.Recursion: Un tipo de dato recursivo T se define como una estructura que puede contener componentes del tipo T.

.Este define datos agrupados cuyo tamaño puede crecer arbitrariamente y cuya estructura puede ser arbitrariamente compleja.

.Los lenguajes de programacion convencionales soportan la implementacion de los tipos de datos recursivos a traves de los punteros.

.Un **puntero** es una referencia a un objeto

.Una **variable puntero** es una variable cuyo r-valor es una referencia a un objeto.

Punteros:

.Estructuras de tamaño arbitrario, con un numero de items no determinado: los punteros permiten conectar juntos muchos items sin tener un nombre explicito para todos ellos (recursion).

.Relaciones multiples entre los items: los punteros permiten que el dato sea puesto en varias estructuras sin necesidad de duplicarlo.

.Acceso a bajo nivel: los punteros estan cerca de la maquina.

.Los valores que pueden tomar son direcciones de memoria o valor nulo (no asinado)/ direccion no valida.

.Las operaciones que se pueden llevar a cabo son la asignacion de valor (generalmente asociado a la alocacion de la variable apuntada) y referencias a su valor (como direccion), operaciones entre punteros al valor de la variable apuntada (desreferenciacion implicita).

.Los punteros son un mecanismo muy potente para definir estructuras de datos recursivas, pero por acceder a bajo nivel pueden hacer inseguros a los programas que los usan.

Inseguridad de los punteros:

1.Violacion de tipos

2.Referencias sueltas – referencias dangling: si este objeto no esta alocado se dice que el puntero es peligroso (dangling). Una referencia suelta es un puntero que contiene una direccion de una variable dinamica que fue desalocada. Si luego se usa el puntero producira error.

3.Punteros no inicializados: Genera el peligro de acceder de forma descontrolada a posiciones de memoria. Algunos lenguajes solucionan esto asignandole un valor nulo.

4.Punteros y uniones discriminadas: En el caso de C, este es el mismo efecto que causa la aritmetica de punteros. Para resolver este problema asociado con los punteros, Java elimina la nocion de puntero explicito completamente.

5.Alias: no lo entendi.

6.Liberacion de memoria (objetos perdidos): Las variables puntero se alocan como cualquier otra variable en la pila de registros de activacion .

.Los objetos (apuntados) que se alocan a traves de la primitiva new son alocados en la heap, la memoria (heap) podria rapidamente agotarse a menos que de alguna forma se devuelva el almacenamiento alocado liberado y si los objetos en el heap dejan de ser accesibles esa memoria podria liberarse (por el garbage).

.Un objeto se dice accesible si alguna variable en la pila lo apunta directa o indirectamente y un objeto es basura si no es accesible.

Mecanismos para desalocar memoria:

.La liberacion de memoria puede ser de dos formas explicita (realizada por el programador con dispose/delete) o implicita (independiente de la aplicación realizada por el garbage collector).

.El reconocimiento de la basura recae en el programador, quien notifica al sistema cuando un objeto ya no se usa. Esto no garantiza que no haya otro puntero que apunte a esta direccion definida como basura, este puntero se transfora en dangling (puntero suelto). **Este error es dificil de chequear y la mayoria de los lenguajes no lo implementan por que es costoso**.

TEMA GARBAGE COLLECTOR NO ENTRA EN EL PARCIAL POR LO QUE SE PUBLICO.

Tipo de dato abstracto - TADs:

.Abstraccion: es el mecanismo que tenemos las personas para manejar la complejidad. Abstraer es representar algo descubriendo sus caracteristicas esenciales y suprimiendo las que no lo son. El principio basico de la abstraccion es la **informacion oculta**.

TAD = Representacion (datos) + operaciones (funciones y procedimientos).

.Los tipos de datos son abstracciones y el proceso de construir nuevos tipos se llama **abstraccion de datos**.

.Los nuevos tipos de datos definidos por el usuario se llamana **tipos abstractos de datos**.

Un Tipo abstracto de dato (TAD) es el que satisface:

.Encapsulamiento: la representacion del tipo y las operaciones permitidas para los objetos del tipo se describen en una unica unidad sintactica. *Refleja las abstracciones descubiertas en el diseño*

.Ocultamiento de la informacion: la representacion de los objetos y la implementacion del tipo permanencen ocultos. *Refleja los niveles de abstraccion. Modificabilidad*

.Las unidades de programacion de lenguajes que pueden implementar un TAD reciben distintos nombres:

.Simula-67: proporciona una estructura sintactica que permite que las operaciones y la representacion puedan especificarse en una unica unidad sintactica (class). Pero no satisface el ocultamiento de la informacion.

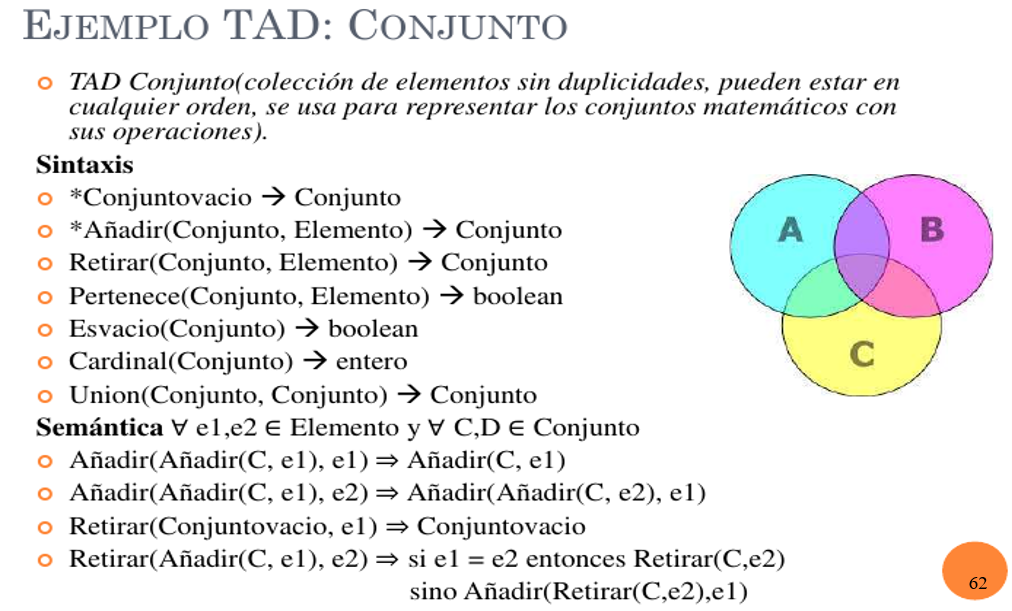
.Modula-2 modulo.

.ADA paquete

.C++ y Java clase.

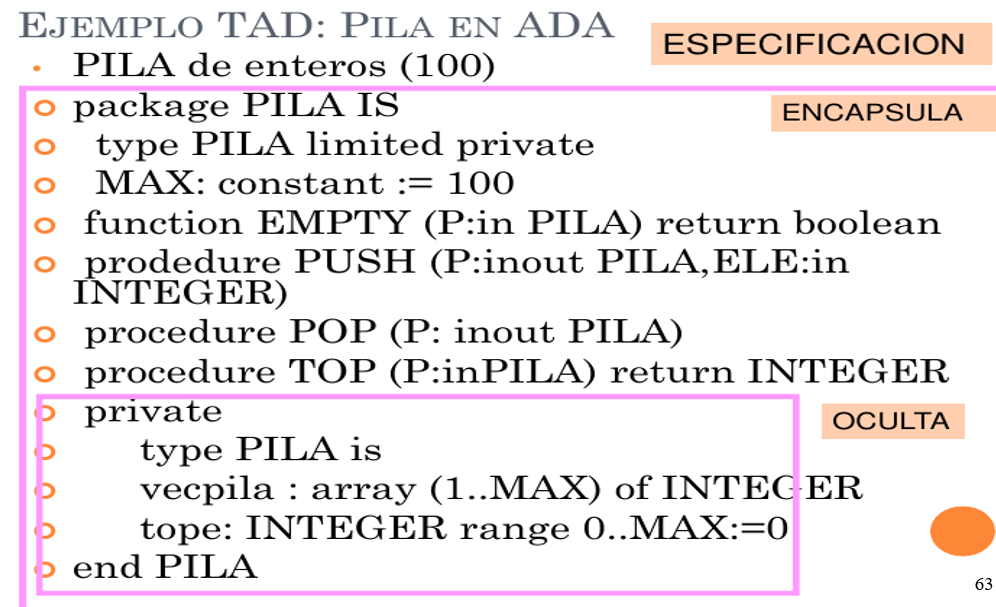
Especificacion de un TAD: la especificacion formal proporciona un conjunto de axiomas que describen el comportamiento de todas las operaciones. Ha de incluir una parte de sintaxis y una parte de semantica.

Por ej:



.Hay operaciones definidas por si mismas que se consideran constructores del TAD. Normalmente, se elige como constructor la operación que inicializa.

.Entiendo que en el ejemplo siguiente de la pila en ADA queda publico osea los metodos basicamente como en JAVA que uno tiene que definir los metodos abstractos, pero “queda oculto” en la clase la definicion de la pila que seria la **parte oculta.**



TAD – Clases: En terminos practicos, una clase es un tipo definido por el usuario. Una clase contiene especificacion de los datos que describen un objeto junto con la descripcion de las acciones que un objeto conoce. Atributos + metodos. Ademas, agrega un segundo nivel de abstraccion que consiste en agrupar las clases en jerarquias de clases. De forma que la clase hereda todas las propiedades de la superclase.

Sistema de tipos:

Definicion: es un conjunto de reglas usadas por un lenguaje para estructurar y organizar sus tipos. El objetivo de un sistema de tipos es escribir programas seguros.

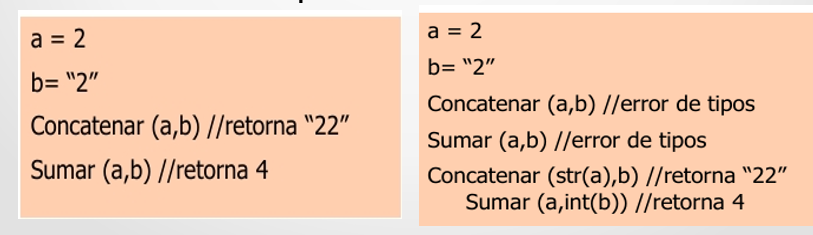
.Conocer el sistema de tipos de un lenguaje nos permite conocer de una mejor forma los aspectos semanticos del lenguaje.

.Este provee mecanismos de expresion: tales como expresar wtipos intrinsecos o definir tipos nuevos y asociar los tipos definidos con construcciones del lenguaje.

.Define reglas de resolucion: tales como la equivalencia de tipos (dos valores tienen el mismo tipo?), compatibilidad de tipos (puedo usar el tipo en este contexto?) y la interferencia de tipos (cual tipo se deduce del contexto?)

.Mientras mas flexible sea el lenguaje mas complejo el sistema. Seguridad VS Flexibilidad.

Tipado fuerte – tipado debil: se dice que el sistema de tipos es fuerte cuando especifica restricciones sobre como las operaciones que involucran valores de diferentes tipos pueden operarse. Lo contrario establece un sistema debil de tipos.



Sistema de tipos – especificacion:

.Tipo y tiempo de chequeo:

.Tipos de ligadura: Tipado estatico o tipado dinamico (lo ya visto)

Lenguaje fuertemente tipado: si el lenguaje es fuertemente tipado el compilador puede garantizar la ausencia de errores de tipo en los programas. Un lenguaje se dice fuertemente tipado si el sistema de tipos impone restricciones que aseguran que no se produciran errores de tipo en ejecucion. **Un lenguaje se dice fuertemente tipado si todos los errores de tipo se detectan**. Existen lenguajes fuertemente tipado con tipado dinamico (python) y con tipado estatico (gobstone), ademas como ejemplo C es debilmente tipado y tiene tipado estatico.

.Reglas de equivalencia y conversion:

.Tipo compatible: reglas semanticas que determinan si el tipo de un objeto es valido en un contexto particular.

.Un lenguaje debe definir en que contexto el tipo Q es compatible con el tipo T. Si el sistema de tipos define la compatibilidad.

.Equivalencia por nombre: dos variables son del mismo tipo si y solo si estan declaradas juntas o si estan declaradas con el mismo nombre de tipo.

.Equivalencia por estructura: dos variables son del mismo tipo si los componentes de su tipo son iguales.

.Un tipo es compatible con otro si es equivalente y se puede convertir.

.Coercion: significa convertir un valor de un tipo a otro. Reglas del lenguaje de acuerdo al tipo de los operandos y a la jerarquia.

.Widening (ensanchar): cada valor del dominio tiene su correspondiente valor en el rango. (Entero a real). Pascal solo widening de entero a real.

.Narrowing (estrechar): cada valor del dominio puede no tener su correspondiente valor en el rango. En este caso algunos lenguajes producen un mensaje avisando la perdida de informacion. (Real a entero). En C depende del contexto y utiliza un sistema de coersion simple.

.Clausula de casting: conversiones explicitas, se fuerza a que se convierta.

.Reglas de inferencia de tipo:

.La interferencia de tipos permite que el tipo de una entidad declarada se “infiera” en lugar de ser declarado. La inferencia puede realizarse de acuerdo al tipo de:

.Un operador predefinido: fun f1(n,m) = (n mod m = 0)

.Un operando: fun f2(n) = (n\*2)

.Un argumento: fun f3(n) = n\*n

.El tipo del resultado: fun f4(n) = (n\*n)

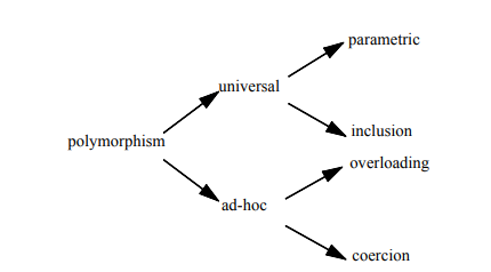
.Ejemplos.

Nivel de polimorfismo del lenguaje:

.Un lenguaje se dice mono-morfico si cada entidad se liga a un unico tipo (estaticos). En un lenguaje de programacion mono-morfico la funcion disjuntos(Ej pag 83) debera implementarse para cada tipo particular del conjunto.

.Un lenguaje se dice polimorfico si las entidades pueden estar ligadas a mas de un tipo, las variables polimorficas pueden tomar valores de diferentes tipos, las operaciones polimorficas son funciones que aceptan operandos de varios tipos y los tipos polimorficos tienen operaciones polimorficas.

.Todos los lenguajes practicos tienen cierto grado de polimorfismo. En consecuencia, las preguntas importantes a responder son ¿Qué diferentes tipos (o grados) de polimorfismo se identifican? Y ¿hasta donde podemos llegar?



.El **polimorfismo ad-hoc** permite que una funcion se aplique a distintos tipos con un comportamiento sustancialmente diferente en cada caso.

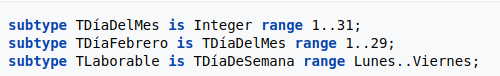
.El termino **sobrecarga** se utiliza para referirse a conjuntos de abstracciones diferentes que estan ligadas al mismo simbolo o identificador.

.La **coercion** permite que un operador que espera un operando de un determinado tipo T puede aplicarse de manera segura sobre un operando de un tipo diferente al esperado.

.El **polimorfismo universal** permite que una unica operación se aplique uniformemente sobre un conjunto de tipos relacionados.

.Si la uniformidad de la estructura de tipos esta dada a traves de parametros, hablamos de **polimorfismo parametrico**. Un tipo parametrizado es un tipo que tiene otros tipos como parametros. Ejemplo, Tipo lista con Tipo T como parametro.

.El **polimorfismo por inclusion** es otra forma de polimorfismo universal que permite modelar subtipos y herencia. Si un tipo se define como un conjunto de valores y un conjunto de operaciones, un subtipo T’ de un tipo T puede definirse como un subconjunto de los valores de T y el mismo conjunto de operaciones.



.El mecanismo de herencia permite definir una nueva clase derivada a partir de una clase base ya existente. Esta podria agregar atributos y comportamiento.