



Universidad Tecnológica Nacional
FACULTAD REGIONAL CORDOBA

PARADIGMAS DE PROGRAMACION

Unidad VI

Elementos constitutivos de Lenguajes y Paradigmas



Objetivo

- Que el alumno reconozca los elementos constitutivos, mecanismos y formas de implementación de las características de los lenguajes de programación dentro de los paradigmas de programación correspondientes.

Contenidos Abordados

- Conceptos lógicos y transversales.
- Abstracción.
- Tipos de datos.
- Mecanismos de control de flujo.



Introducción: Conceptos lógicos y transversales.

Abstracción:

- Es una técnica para crear, comprender o manejar sistemas complejos.
 - En cada nivel de detalle cierta información se muestra y cierta información se omite.
 - Ejemplo: Diferentes escalas en mapas.
- Mediante la abstracción creamos **MODELOS** de la realidad.
- Se aplica a todos los ámbitos, no sólo a la Informática.

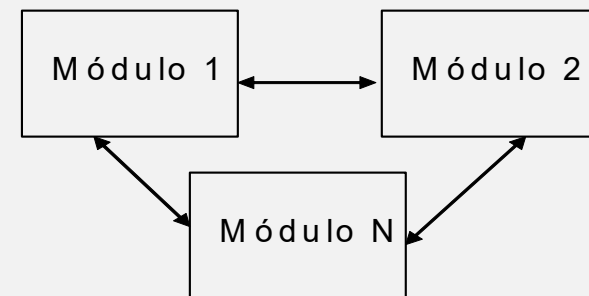
Introducción: Conceptos lógicos y transversales.

Modularización :

- Estrategia de programación que permite organizar la funcionalidad de un sistema complejo en pequeñas partes,
- Se organiza en unidades más pequeñas de software que se responsabilizan de tareas específicas y que interactúan entre ellas.

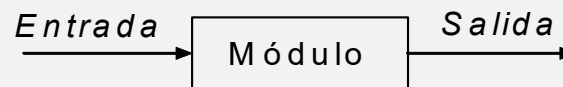
Módulos
Funciones
Métodos
Bloques
Entidades, etc.

Rutinas
Procedimientos
Predicados
Subprogramas

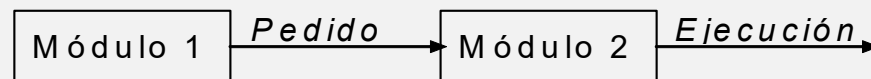


Introducción: Conceptos lógicos y transversales.

- **Encapsulamiento** : Cada parte o módulo no conoce el funcionamiento interno de los demás módulos, solo su Interfaz, se lo conoce también como ocultación de información o “caja negra”.



- **Delegación** : Se refiere a la distribución de responsabilidades entre las diferentes unidades de software, consiste en la invocación desde un módulo a otro módulo. El que pide explicita el que recibe ejecuta.



Introducción: Conceptos lógicos y transversales.

Proceduralidad:

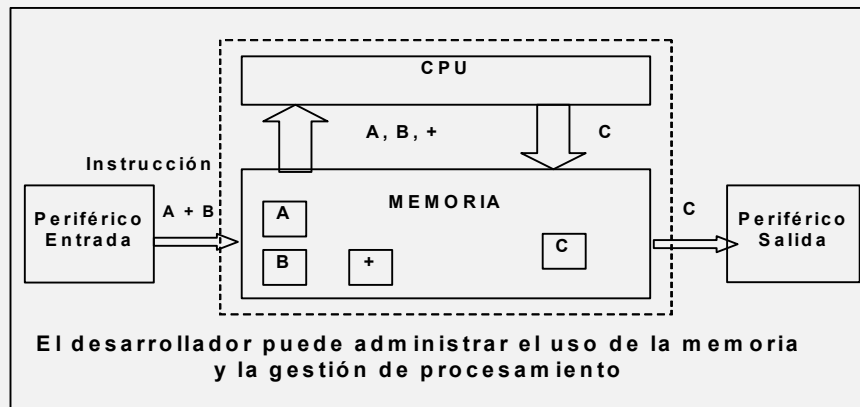
- Indican explícitamente secuencia de ejecución.
- Detallan un conjunto de sentencias.
- Ordenadas mediante estructuras de control (decisiones, iteraciones y secuencias) conforman “algoritmos”.
- Opuesta a la declaratividad.

Declaratividad:

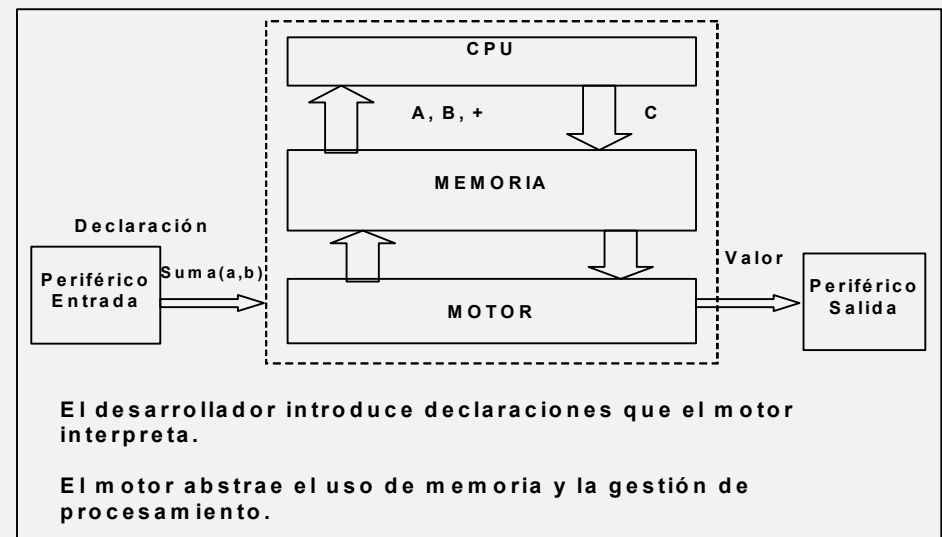
- Separación de la lógica y el control.
- Conjunto de declaraciones, que pueden ser proposiciones, condiciones, restricciones, afirmaciones, o ecuaciones, que caracterizan al problema y describen su solución.
- El sistema utiliza mecanismos internos de control, comúnmente llamado “motores”.

Introducción: Conceptos lógicos y transversales.

MODELO BASADO EN INSTRUCCIONES



MODELO BASADO EN DECLARACIONES



Introducción: Conceptos lógicos y transversales.

Tipo de Datos

- Es un conjunto de valores y de operaciones asociados a los datos.
- Permiten la agrupación o clasificación del gran volumen y variedad de valores que es necesario representar y operar en un sistema.
- Los paradigmas y lenguajes poseen sistemas de tipos de datos, que especifican los tipos de dato, con sus valores y operaciones asociadas.

Introducción: Conceptos lógicos y transversales.

Estructuras de Datos

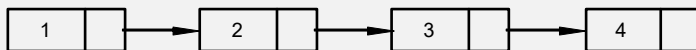
- Son tipos de datos compuestos que se pueden procesar en su conjunto como una unidad o se pueden descomponer en sus partes y tratarlas en forma independiente.
- Poseen características que las definen, tales como:
 - La disposición de los elementos: Lineales y no lineales.
 - La forma de creación y utilización: estáticas y dinámicas.
 - El medio de almacenamiento: temporales y permanentes.

LINEALES

ARREGLO (VECTOR)

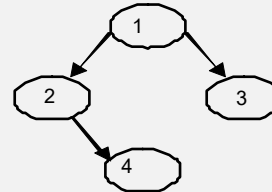


LISTA ENLAZADA

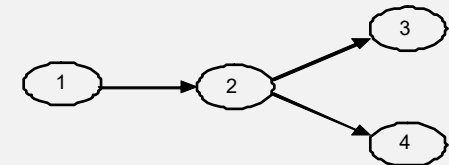


NO
LINEALES

ARBOL



GRAFO





Introducción: Conceptos lógicos y transversales.

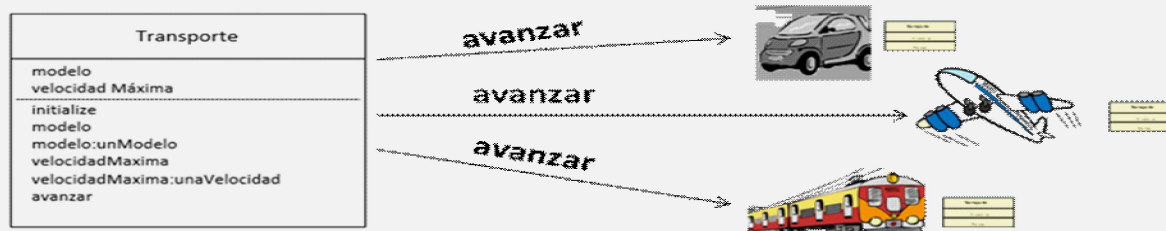
Polimorfismo y Software Genérico

- Permite construir piezas de software genéricas que trabajen indistintamente con diferentes tipos de entidades, para que las mismas puedan ser intercambiadas.
- Hay polimorfismo cuando, ante la existencia de dos o más bloques de software con una misma interfaz, otro bloque de software cualquiera puede trabajar indistintamente con ellos.

Introducción: Conceptos lógicos y transversales.

Polimorfismo de Subtipado:

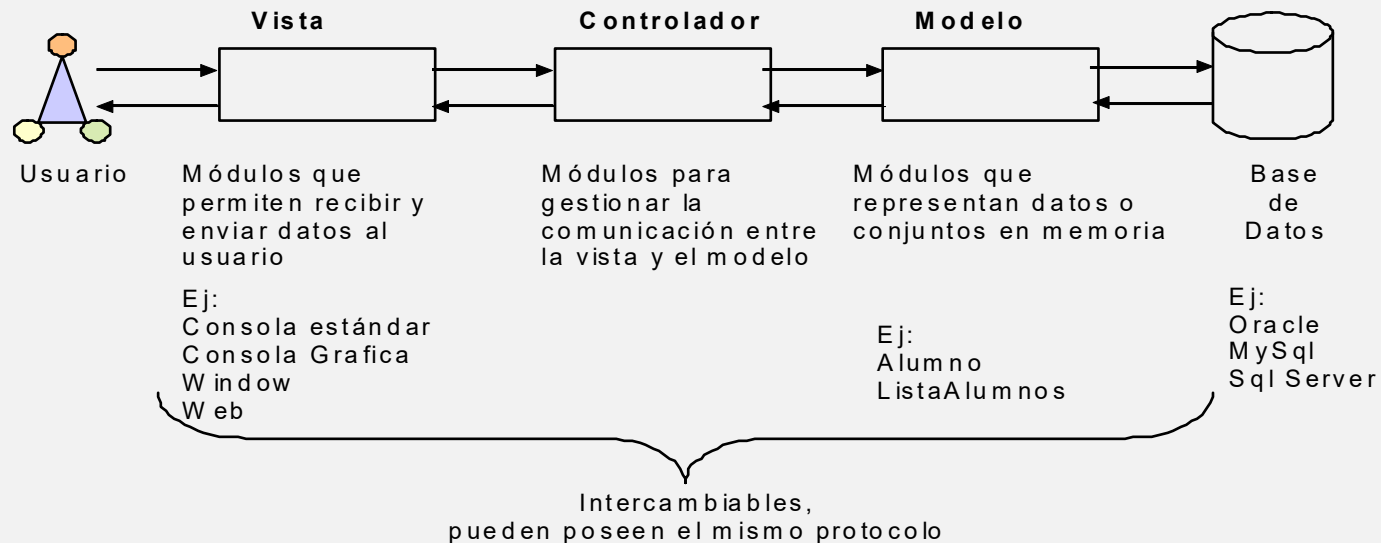
- Es la capacidad de referirse a objetos de clases distintas en una jerarquía utilizando el mismo mensaje para realizar la misma operación, pero de manera diferente.
- Facilita el almacenamiento y transporte de objetos de distintos tipos. Si varios objetos de distintos tipos están relacionados a través de un ancestro en común, con los mismos mensajes se los pueden manipular a través de éste.



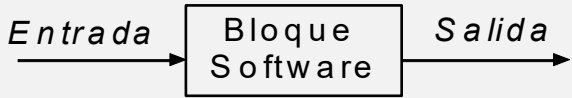
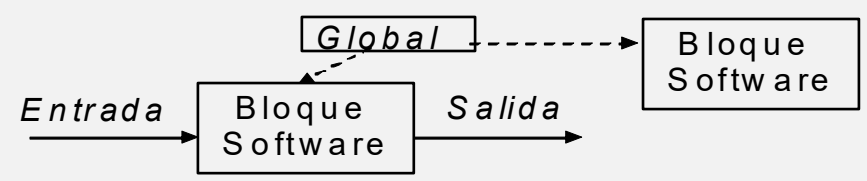
Introducción: Conceptos lógicos y transversales.

Ejemplo de Modularización y código genérico

Patrón de Diseño MVC (Modelo-Vista-Controlador)



Introducción: Conceptos lógicos y transversales.

Transparencia Referencial	Efecto de Lado
<p>El valor de una expresión depende únicamente del valor de sus componentes, se evalúa el mismo bloque de software, sin importar el comportamiento interno de dicho bloque.</p>  <pre>Entrada → [Bloque Software] → Salida</pre> <p>Definición:</p> <pre>int sumaUno(int x) { return (x + 1); }</pre> <p>Invocación:</p> <pre>sumaUno(6) = 7; sumaUno(4) = 5;</pre>	<p>El resultado de la evaluación de un bloque de software depende de otros valores o condiciones del ambiente más allá de sus parámetros.</p>  <pre>Entrada → [Bloque Software] → Salida</pre> <p>Definición:</p> <pre>int global = 0; int sumaValor(int x) { global = global + 1; return (x + global); }</pre> <p>Invocación:</p> <pre>sumaValor(6) = 7; sumaValor(4) = 6; global = 9; sumaValor(2) = 12;</pre>



Introducción: Conceptos lógicos y transversales.

Asignación:

- Consiste en cambiar la información representada por una variable, de forma que si se consulta su valor antes y después, de dicha operación se obtiene un resultado distinto.
- Esta operación es propia de los lenguajes de la vertiente imperativa.

Unificación:

- Es un mecanismo por el cual una variable que no tiene valor asume valor.
- Una vez unificada (ligada), no cambia su valor.
- Esta operación es propia de los lenguajes de la vertiente declarativa.



Introducción: Conceptos lógicos y transversales.

Modo de Evaluación

- Los parámetros que se utilizan en la invocación de un bloque de software pueden ser evaluados en diferentes momentos de acuerdo al modo de evaluación que utilice el lenguaje de programación.
- Los modos de evaluación pueden ser:
 - **Evaluación ansiosa** (antes de la invocación)
 - **Evaluación diferida** (lo hace quien ejecuta).



Introducción: Conceptos lógicos y transversales.

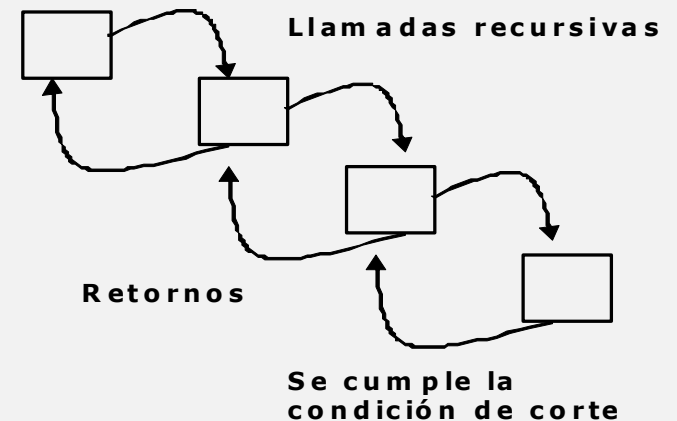
Orden Superior

- Los programas pueden tener como argumento Programas y producir como resultado otros Programas.
- Esta definición se ajusta según el paradigma:
 - En la programación imperativa los procedimientos son tratados como datos, pueden ser utilizados como parámetros y devueltos como resultados.
 - En la programación Funcional las funciones son tratadas como valores, posibilitando pasar funciones como parámetros de otras funciones o devolver funciones como valor de retorno.

Introducción: Conceptos lógicos y transversales.

Recursividad

- Es una técnica que se basa en definir relaciones en términos de ellas mismas.
- Un bloque de software recursivo se define con al menos un término recursivo, en el que se vuelve a invocar el bloque que se esta definiendo y algún término no recursivo como base para detener la recursividad.



Abstracción

- Operación intelectual que ignora selectivamente partes de un todo para facilitar su comprensión.
- En la resolución de problemas: implica ignorar detalles específicos buscando generalidades que ofrezcan una perspectiva distinta, más favorable a su resolución.
- El proceso de abstraerse progresivamente de los detalles, permite manejar **niveles de abstracción**.

Abstracción

- En el paradigma imperativo la principal abstracción requiere que se piense en términos de la estructura de la computadora, en como traducir un problema en instrucciones de computadora.
- Tanto en el paradigma Orientado a objetos como en los declarativos se permiten describir el problema en términos del problema en lugar de términos de la computadora.
- La abstracción se realiza:
 - En POO con objetos o entidades.
 - En P. Funcional con funciones matemáticas.
 - En P. Lógico con proposiciones lógicas.

Abstracción

- Por ejemplo, se necesita conocer la condición de regularidad de un alumno.
- En el paradigma orientado a objetos ¿Como sería el Proceso de abstracción?.



Alumno

Propiedades

legajo
asignatura
nota1
nota2

Operaciones

conocerLegajo....
conocerNota1
conocerNota2
conocerPromedio

Abstracción

Tipos de abstracción

- **Abstracción Procedimental:** Definimos un conjunto de operaciones (procedimiento) que se comporta como una operación.
- **Abstracción de Datos (TDA):** Tenemos un conjunto de datos y un conjunto de operaciones que caracterizan el comportamiento del conjunto. Las operaciones están vinculadas a los datos del tipo.
- **Abstracción de Iteración:** Abstracción que permite trabajar sobre colecciones de objetos sin tener que preocuparse por la forma concreta en que se organizan.

Tipos de Datos.

- **Teoría de Valores y Tipos:** Es el estudio de los valores o datos en el área de las ciencias de computación.
- **Valor:** Es cualquier entidad que puede ser evaluada, almacenada en una estructura de datos, pasada como argumento a una función o procedimiento, retornado como resultado de una función.
- **Tipo:** Es un conjunto de valores que tienen características comunes y exhiben un comportamiento uniforme bajo ciertas operaciones, por ejemplo:

<p>{false, true} es un tipo. (admiten operaciones de negación, conjunción y disyunción). {13,'e', true} no es un tipo</p>	<p>a + b</p> <ul style="list-style-type: none">• Es posible si son números (enteros, reales).• No siempre es posible con otros tipos:• Booleanos (+, or,).• String (“+”, “,”, “++”)
---	---

Tipos de Datos.

- Un sistema de tipos define como un lenguaje de programación clasifica los valores y expresiones en tipos, cómo se pueden manipular estos tipos y cómo interactúan.
- El rango del tipo de dato limita y la forma de su evaluación afecta en el "tipado" del lenguaje.
- Un lenguaje de programación puede asociar una operación concreta con diferentes algoritmos para cada tipo de dato en el caso del polimorfismo.
- La teoría de tipos de datos es el estudio de los sistemas de tipificación.

Clasificación de los Tipos.

- **Tipos Primitivos:** es aquel cuyos valores son atómicos y no pueden ser descompuestos en valores más simples.

Lenguaje	Sintaxis	Conjunto de Valores	Notación
Java, C++	boolean	{ false, true }	Valor- Verdad
Haskell	Bool	{ false, true }	Valor- Verdad
C++	int	{ ... -2, -1, 0, 1, 2 .. }	Entero
Haskell	Int, Integer	{ ... -2, -1, 0, 1, 2 ... }	Entero
Java, C++	float	{ ..,-1.0,.. 0.0, .. ,...1.0, ... }	Real

EJEMPLOS DE TIPOS PRIMITIVOS EN DISTINTOS LENGUAJES

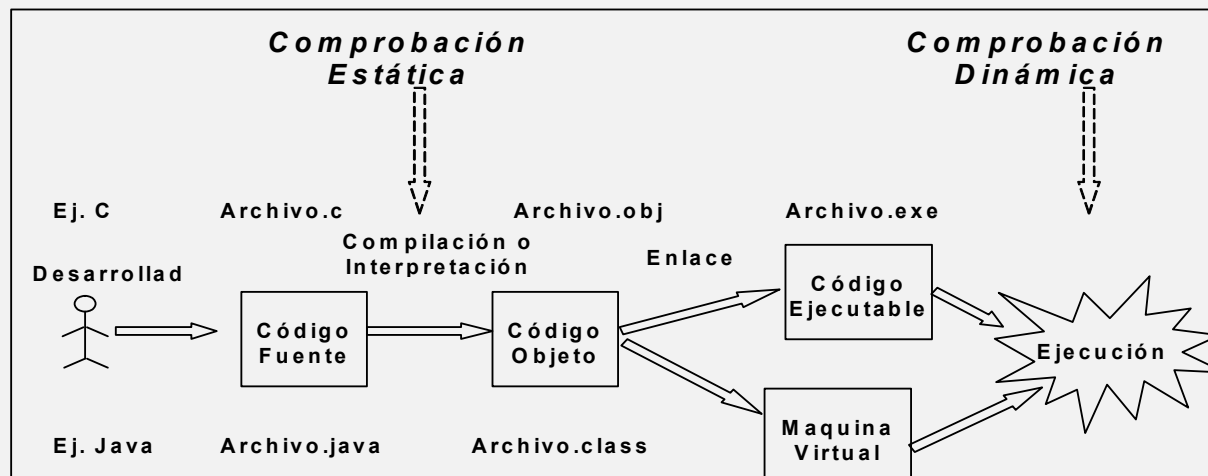
- **Tipos Compuestos:** un tipo compuesto o tipo estructurado de datos es un tipo cuyos valores son compuestos o estructurados a partir de otros valores simples. Los lenguajes soportan una variedad amplia de estructura de datos como tuplas, arreglos, uniones, listas, árboles etc.

Verificación de Tipos.

- Asegura que las operaciones de un programa se apliquen de manera apropiada.
- Su propósito es la prevención de errores.
- Por ejemplo:
 - Para hacer $a * b$, a y b deben ser verificados, para asegurarse de que son números y que la operación está definida para ellos.

Verificación de Tipos.

- En la implementación de los lenguajes esta verificación puede ser realizada en tiempo de compilación o en tiempo de ejecución.
 - **Comprobación Estática:** garantiza la detección de algunos errores de programación. Por ejemplo: Comprobación de Tipos, Flujos de Control, Unicidad, relacionadas con nombre.
 - **Comprobación Dinámica:** se realiza durante la ejecución del programa objeto.



Verificación de Tipos.

Lenguaje de Tipificación Estática

- Cada variable y parámetro tiene un tipo fijo que es elegido por el programador, así el tipo de cada expresión y cada operación puede ser deducido con la verificación de tipos en tiempo de compilación.
- Lenguajes : C, C++, Java y Haskell

Lenguaje de Tipificación Dinámica

- Solo los valores tienen tipos fijos. Una variable o parámetro no tiene un tipo designado, pero puede tomar valores de diferentes tipos en tiempo de ejecución.
- Lenguajes : Lisp, Smalltalk, Clipper, Perl, Python, etc.

Tipificación estática vs. dinámica

Concepto	Estática	Dinámica
Velocidad de compilación	Más Lenta	Más Rápida
Velocidad de ejecución	Más Rápida	Más Lenta
Espacio de memoria	Menor	Mayor (valor+tipo)
Seguridad	Mayor	Menor
Flexibilidad	Menor	Mayor (Reutilización de código)
Efectividad	Mayor	Menor
Costo	Menor	Mayor

Sistema de Tipos.

Ofrecen:

- **Seguridad:** puede permitir a un compilador detectar incoherencias en el significado o código probablemente invalido.
- **Optimización:** un sistema de tipado estático puede dar información muy útil a un compilador.
- **Documentación:** los tipos de datos pueden ser utilizados como una forma de documentación, porque pueden ilustrar la intención del programador.
- **Abstracción:** los tipos de datos permiten a los programadores pensar en los programas a un alto nivel, sin preocuparse con el bajo nivel de implementación.

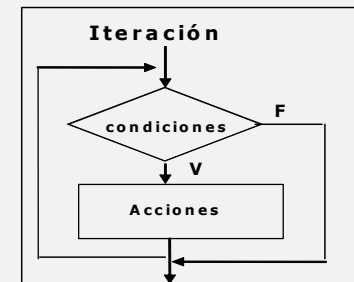
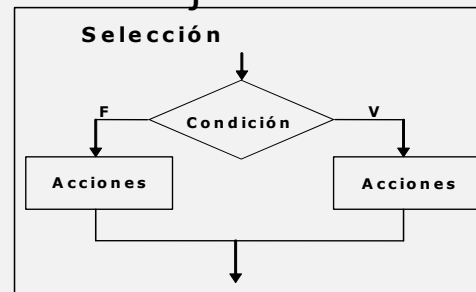
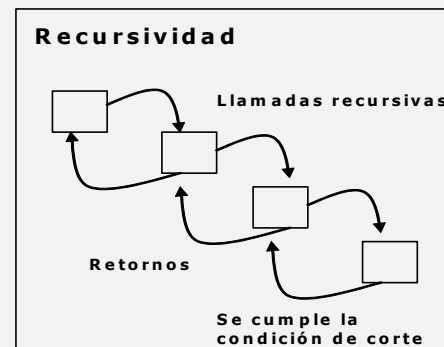
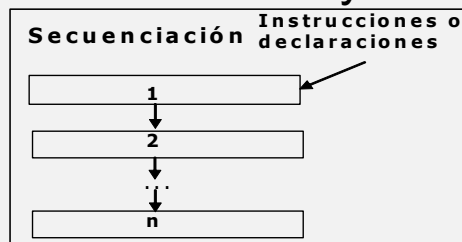
Conversión de Tipos.

- **Coerción:** es la conversión implícita o automática de un tipo de dato a otro, realizada por un lenguaje de programación.
 - Para calcular la raíz cuadrada usando `sqrt(5)`, el entero 5 es automáticamente convertido al valor real 5.0.
 - La expresión `2*3.142` se transforma automáticamente a `2.0*3.142`.
- **Conversión Explícita:** el programador se encarga de hacer las conversiones directamente en código (conocida como casting). La forma de escritura dependerá del lenguaje.
 - Por ejemplo en java:

```
int x = 3;  
x = (int) (1.5 + (double) (x/2));
```

Mecanismos de Control de Flujo.

- El Control de Flujo es fundamental para la mayoría de los modelos de cómputo, pues establece el orden en que debe ejecutarse el programa.
- El flujo de control hace referencia al orden en que las llamadas a funciones, instrucciones y declaraciones son ejecutadas o evaluadas.



Abstracción Procedimental

El mensaje de Smalltalk `do: unBloque` : es una iteración general que recorre cada elemento de la colección y ejecuta un bloque de código.

"Calcula el total de la suma de los números en numeros"

```
| numeros suma |  
numeros := #(1 2 3 4 5).  
suma := 0.  
numeros do: [ :num | suma := suma + num ].  
^ suma
```


Mecanismos de Control de Flujo.

Organización de los mecanismos de control

Secuenciación: orden específico, usualmente el de aparición en el programa.	Abstracción procedimental: una colección de construcciones de control se encapsula como una unidad, sujeta a parametrización.
Selección: se elije entre dos o más instrucciones según alguna condición a tiempo de ejecución.	No determinismo: el orden o escogencia de instrucciones y expresiones no es especificado deliberadamente.
Iteración: un fragmento de código se ejecuta de manera repetida bien sea un número de veces o hasta cumplirse determinada condición a tiempo de ejecución.	Concurrencia: dos o más fragmentos de programa se ejecutan/evalúan al mismo tiempo.
Recursión: una expresión se define en términos de si misma directa o indirectamente..	

Flujo de control en C y Java.

Selección

```
if (condicion)
{
    instrucciones
}
else
{
    instrucciones
}
```

```
switch(expresion) {
    case valor_1:
        instrucciones_1
        [break;]
    ...
    default:
        instrucciones
}
```

Flujo de control en C y Java.

Iteración:

```
while (condicion) {  
    instrucciones  
}
```

```
do {  
    instrucciones  
} while (condicion)
```

```
for (inicializacion; condicion; incremento) {  
    instrucciones  
}
```

Interrupción de flujo:

- **break**: permite salir de la estructura alternativa o múltiple.
- **continue**: permite saltar al principio de una ejecución repetitiva.
- **return** expr: retorna desde una función una expresión o no.

Flujo de control en Smalltak.

Las estructuras de control son abstracciones procedimentales, que se construyen a partir de mensajes enviados a objetos booleanos, bloques y números, y de la cooperación entre ellos.

- **Selección:** Se forma enviando mensajes a objetos booleanos y evaluando bloques si las condiciones se cumplen.

Ejemplo: `(a > b) ifTrue: [c := a + b] .`

- **Iteración:** hay cuatro tipos de abstracciones `timesRepeat`, `whileFalse`, `whileTrue` y `to:do`.

Ejemplo: Itera de 1 hasta 10, imprimiendo los sucesivos números en la ventana Transcript.

```
1 to: 10  
do: [ :x | Transcript show: x; cr].
```

Flujo de control en Haskell.

Haskell es un lenguaje de programación puramente funcional, en la que no decimos al computador que debe hacer sino mas bien decimos como son las cosas. Y la forma de expresión es por medio de funciones.

- Las selección se resuelve con declaraciones **guarded e if...then..else**.

```
if x > y           | x > y = x
  then x           | x <= y = y
  else y
```

- Utiliza la **recursividad**, (una alternativa para las iteraciones).

Flujo de control en Prolog.

- El Desarrollador no controla el flujo de control.
- El control se entiende a la forma en que el lenguaje busca las respuestas a los objetivos. El desarrollador sólo especifica los hechos y reglas que los relacionan y luego se formulan consultas que son los objetivos o metas a obtener.
- La selección se podría comparar con las reglas lógicas, que son afirmaciones condicionales, por ejemplo:
 - *Hecho o Afirmación incondicional*
Alumno (Juan)
 - *Regla o Afirmación condicional (será cierta si se cumplen las proposiciones dependientes)*
Regular (X) if Alumno (X) and AproboPrimerParcial (X) and AproboSegundoParcial (X)