Introducción a la Programación

Elementos de Programación

PROGRAMACIÓN

Hacer que las computadoras hagan lo que nosotros queremos que hagan.

Ámbitos de aplicación:

- **Control de procesos industriales, máquinas, herramientas.**
- **Electrodomésticos, comunicaciones.**
- **Gestión de negocios, oficina.**

Objetivo:
Definir los pasos a seguir para resolver un problema dado y conseguir un resultado esperado: CONSISTENTE, REPETIBLE Y CONFIABLE.

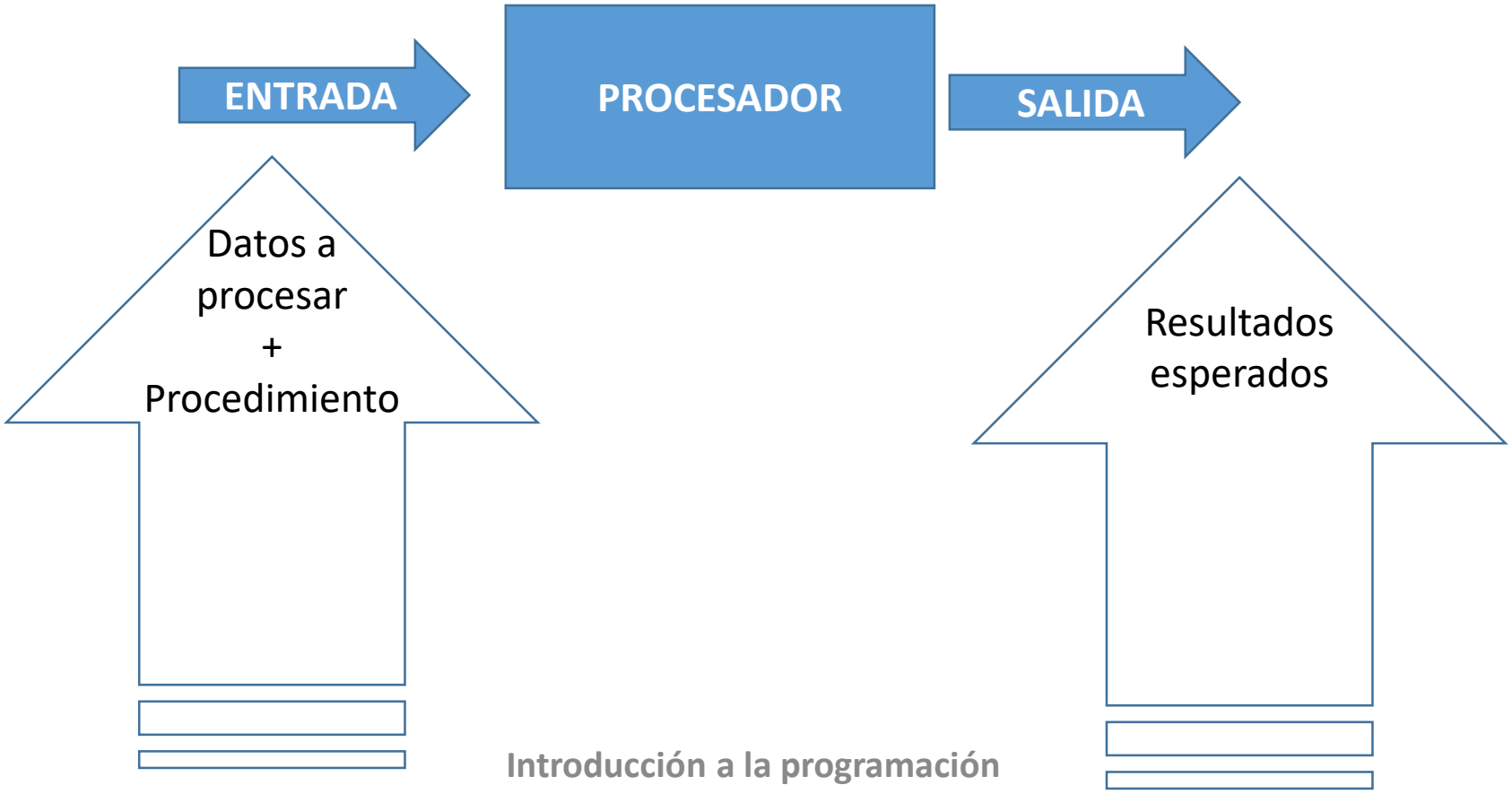
PROGRAMA

- **Son instrucciones ejecutables por el procesador que llevan a cabo un *algoritmo* que permiten resolver un problema dado.**

- **El algoritmo normalmente define SIEMPRE el mismo comportamiento para un conjunto de variables.**
- **El primer paso es encontrar el ALGORTIMO, es decir el mecanismo que – codificado en un cierto lenguaje- nos permita resolver el problema.**

- **Este mecanismo comprende 2 conceptos:**
  - **DATOS → Que representan "objetos" sobre los que operamos.**
  - **PROCEDIMIENTOS → Que son las reglas que definen cómo manipular los datos.**

PROCESADOR


ENTRADA
PROCESADOR
SALIDA
Datos a procesar + Procedimiento
Resultados esperados

PROCESADOR

- **Es quien "ejecuta" el programa y hace visible el resultado esperado del programa.**

MISMA ESTRUCTURA, distintos TAMAÑOS:

- **Marcapasos cardíaco, implantes inteligentes.**
- **Micros embebidos (PDA, celular, mobile, etc).**
- **Microprocesadores (Notebooks & PCS).**
- **Microprocesadores de alta gama (Midrange).**
- **Procesadores híbridos (Mainframe)**

EL LENGUAJE DEL PROCESADOR

SISTEMA BINARIO:

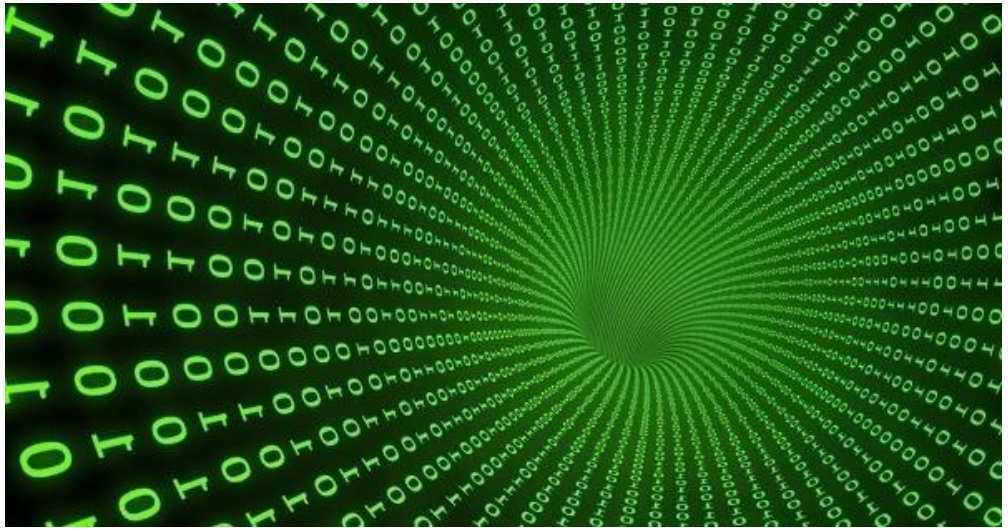
- **1101 1010 0100 1000 (DA68) → Activar motor #4.**

Originalmente especificado a través de diversos mecanismos:

- **Cintas/ tarjetas perforadoras (Jacquard)**
- **Llaves (On- off).**

Lenguajes:

- **Bajo nivel: Assembler → MOV A, X**
- **Alto nivel: COBOL, Java, Phyton → print PI * pow(x,2)/2**



CARACTERÍSTICAS

La mayoría de los lenguajes (para no decir todos) son capaces de realizar:

- **Operaciones aritméticas.**
- **Manipulación y análisis de texto.**
- **Loops: ejecutar grupo de instrucciones repetidamente.**
- **Condicionales: ejecutar grupo de instrucciones según criterio.**
- **Entrada/salida: recibir y entregar datos de y al mundo exterior.**
- **Funciones y procedimientos para manipular los datos.**
- **Utilizar librerías de funciones y procedimientos “prefabricados”.**
- **Organizar programa combinando estructuras simples para formar otras complejas.**

CÓDIGO FUENTE

- **Se conoce como "código fuente" (source code) al documento de texto que contiene las instrucciones en el lenguaje correspondiente:**

```
__author__ = "Mark Pilgrim (mark@diveintopython.org)"
__version__ = "$Revision: 1.2 $"
__date__ = "$Date: 2004/05/05 21:57:19 $"
__copyright__ = "Copyright (c) 2004 Mark Pilgrim"
__license__ = "Python"

def fibonacci(max):
    a, b = 0, 1
    while a < max:
        yield a
        a, b = b, a+b

for n in fibonacci(1000):
    print n,
```

EJECUCIÓN DEL CÓDIGO

El código fuente contiene el programa en un formato legible, en el lenguaje de alto nivel correspondiente (ej: COBOL, Java, Phyton)

Existen varios mecanismos para que el procesador EJECUTE las instrucciones del código:

- **INTERPRETADO: el código fuente se lee y ejecuta instrucción por instrucción, con poca ó ninguna optimización.**
- **COMPILADO: el código fuente es procesado por un compilador el cual genera:**
  - **Objeto: ejecutable (binario): optimizado por la plataforma de hardware y sistema operativo correspondiente.**
  - **Bytecodes: es un paso intermedio, que luego será interpretado por una máquina virtual VM.**

COMPILADOR VS. INTÉRPRETE

COMPILADOR

Genera un objeto binario:

- **Muy compacto.**
- **Optimizado.**
- **Específico para el sistema operativo/ procesador objetivo.**

INTÉRPRETE

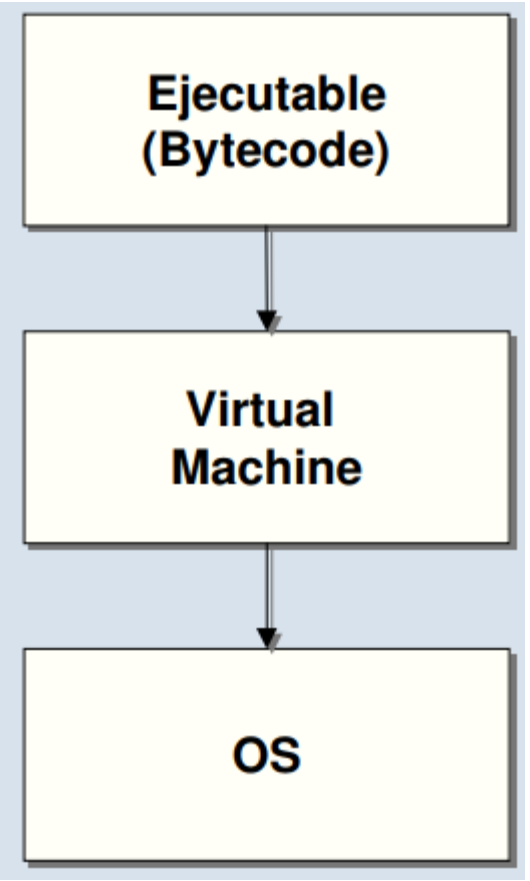
Ejecuta funciones paso a paso (script)

- **Muy flexible, se cambia el código y se corre.**
- **Se interpreta en el momento.**
- **No está optimizado.**

MÁQUINA VIRTUAL

Para que el ejecutable se independiente del sistema operativo (portable) se ejecuta en una MV.

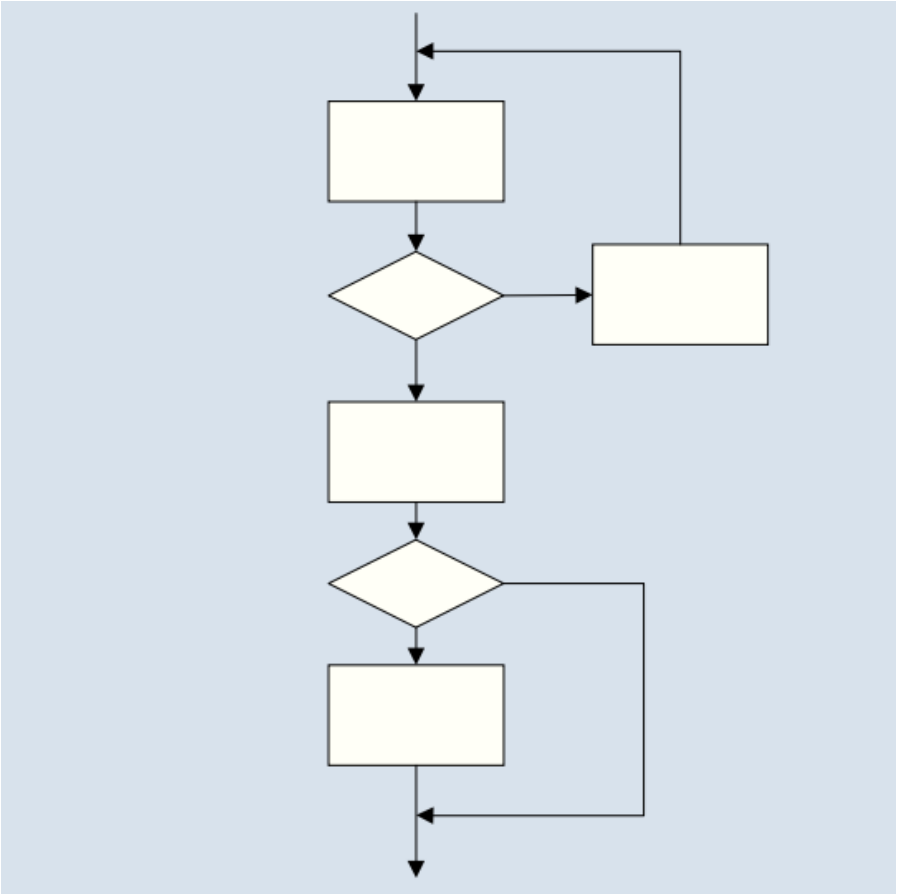
- **Ésta "máquina" interpreta el programa compilado (bytecode).**
- **Cada sistema operativo tiene una versión de la MV, pero todas son iguales de cara al programa compilado.**



CICLO DE VIDA DEL SOFTWARE

- ✓ **Diseñar el algoritmo de solución.**
- ✓ **Escribir el código fuente y generar el objetivo ejecutable (script/bytecode/exe).**
- ✓ **Ejecutar la solución y verificar el resultado.**
- ✓ **Debug.**

DISEÑAR EL ALGORITMO



ELEMENTOS DE LA PROGRAMACIÓN

Hay 5 elementos presentes en la mayoría de los lenguajes de programación:

- **Variables.**
- **Loops.**
- **Condicionales.**
- **Entrada/salida**
- **Subrutinas y funciones.**

VARIABLES

Es la forma que se presentan los datos, puede ir de algo tan simple como la edad, el nombre, hasta algo complejo como el conjunto de afiliados de una prepaga, con sus números de carnet, qué prestaciones utilizó cada uno, con que médico, etc.

```
nombre = "Juan Manuel"
apellido = "Fangio"
campeonatosObtenidos = 5
añosCampeon = [1951, 1954, 1955, 1956, 1957]
cv = [ "X", ["Ing.Civil", "Bachiller"],["Roggio"]..]
```

VARIABLES

Es la "memoria" del programa:

- **Saldo = 1000**
- **Saldo = saldo -200**
- **Saldo = saldo + 40**

SALDO
1000
800
840

Son posiciones de memoria donde se guardan:

- **Datos de entrada (parámetros)**
- **Resultados intermedios y datos de salida.**

El contenido de la variable se actualiza desde:

- **Parámetros externos**
- **Dispositivos de entrada (interfaz usuario)**
- **Lectura de medios magnéticos/ópticos (archivos, etc.)**
- **Cálculos y manipulación interna.**

TIPO DE VARIABLES

- a = “Esto es un texto”
- b = 12345
- c = 3.14159…
- d = False
- e = [0,0,1][0,1,0][1,0,0]
- f = x’0F14BE20200DEAF130’
- g = [“Lunes”, “Martes”, … ”Domingo”]

TIPO DE VARIABLES

- **Para algunos lenguajes (ej. JavaScript) cada variable puede contener distintos tipos de datos:**

```
- var nombre = "Duke"
- var edad = 12
- var importe = "123.456"
- var comodin = "Blink"
- comodin = 182
```

- **Para detectar posibles errores de forma temprana, algunos lenguajes como JAVA exigen que se declare el tipo de dato a contener:**

```
- string nombre = "Duke"
- integer edad = 12
- float importe = "123.456"    ERROR!!!
```

INICIACION DE LAS VARIABLES

- **Algunos lenguajes exigen "declarar la variable antes de utilizarla.**
- **En general, al declarar la variable se inicia con un valor vacío (espacio, 0, 0.00, ect.) Algunos lenguajes NO inician la variable → error al utilizarla.**
- **Es una buena práctica de programación asignarle un valor inicial a cada variable, evitando así sorpresas y garantizado la integridad del programa (→ no hay estados inválidos)**

OPERACIONES CON LAS VARIABLES

La mayoría de los lenguajes provee operadores aritméticos (+, -, *, /, etc) para las operaciones básicas

```
suma = 13;
suma = suma + 5;
print suma;
18
```

```
print 14 + 8 - 2 * (2 + 4)/ 5;
20
```

Nótese las construcción: DATO=DATO + 1

OTRAS: X+=2 X*=5 X++

LOOPS

Permiten ejecutar grupos de instrucciones del programa repetidas veces bajo ciertos criterios, característica fundamental de todos lenguaje.

CONSTRUCCIÓN "FOR"

```
for mes in [ "Enero", "Febrero", "Marzo" ]:print mes
Enero
Febrero
Marzo
```

CONSTRUCCIÓN "WHILE"

```
i=0
while ( i < 3 ) : print i; i += 1;
0
1
2
```

CONDICIONALES

Permiten ejecutar grupos de instrucciones según si se cumple un determinado criterio, característica fundamental de todo lenguaje.

Se basan en la comparación de variables:

```
age = 14
if (age < 20) : print "Purrete!";
Purrete!
```

Hay que diferenciar la comparación de variables:

- **Númericas (integer, float, etc.)**
- **Alfanúmericas (string)**

SINTAXIS: IF (EXPRESION) THEN… ELSE
CONDICIONES MÚLTIPLES: THEN, ELSE, IF, ELIF

ESTRUCTURAS DE LA PROGRAMACIÓN

Todo programa se puede construir utilizando 3 tipos de estructuras:

- ✓ **SECUENCIA : Las instrucciones se ejecutan en el orden que están escritas.**

- ✓ **DECISIÓN: Al entrar en este bloque, el valor de una variable determina que sub- bloque(s) se ejecutara(n) a continuación.**

- ✓ **ITERACIÓN: Al entrar en este bloque, el valor de una variable determina cuántas veces se ejecutará el mismo contenido.**

ENFOQUES

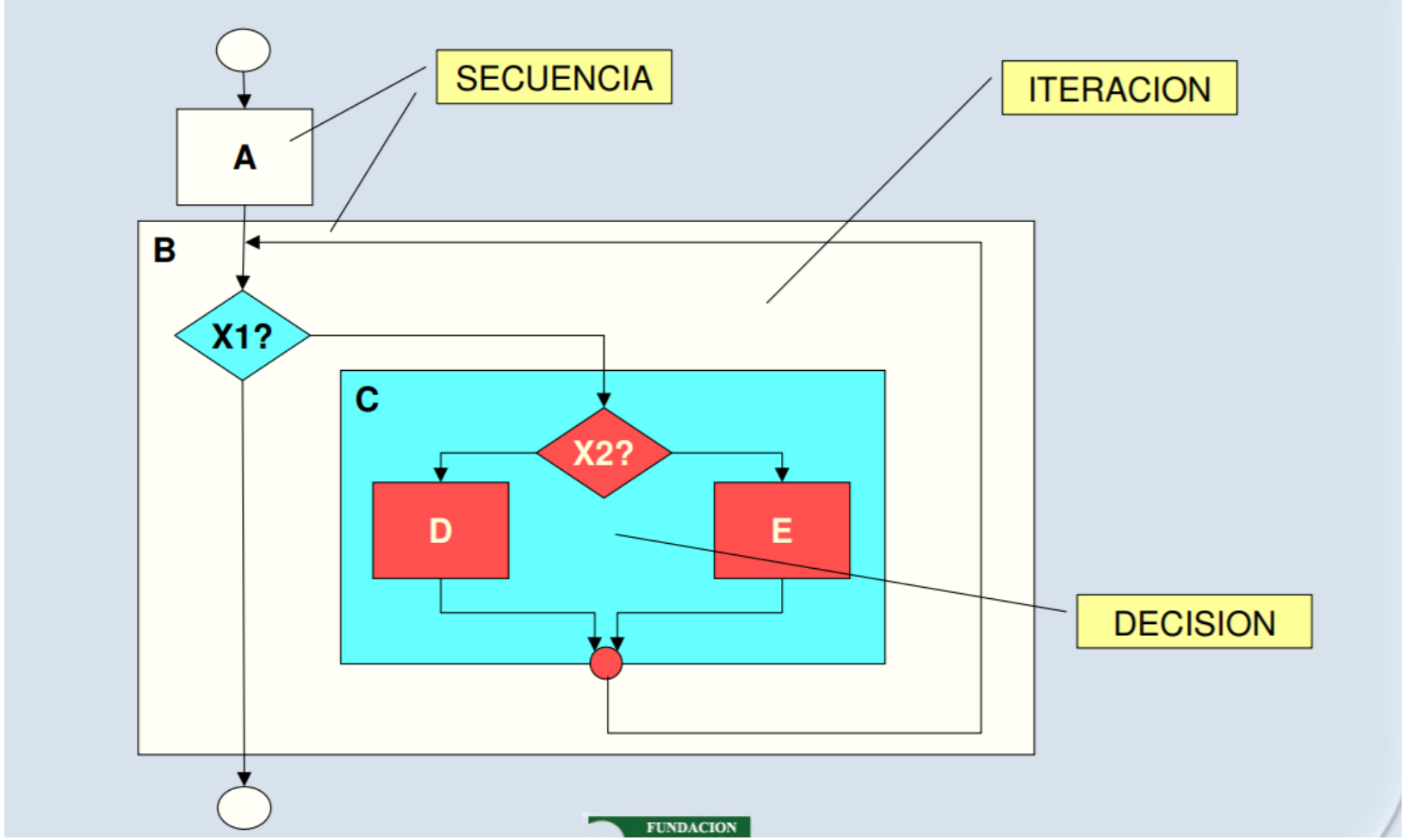
TÉCNICAS ESTRUCTURADAS

- Refinamiento progresivo.
- Dividir y conquistar.
- Reducir la complejidad utilizando una estructura de bloques más simples y manejables.

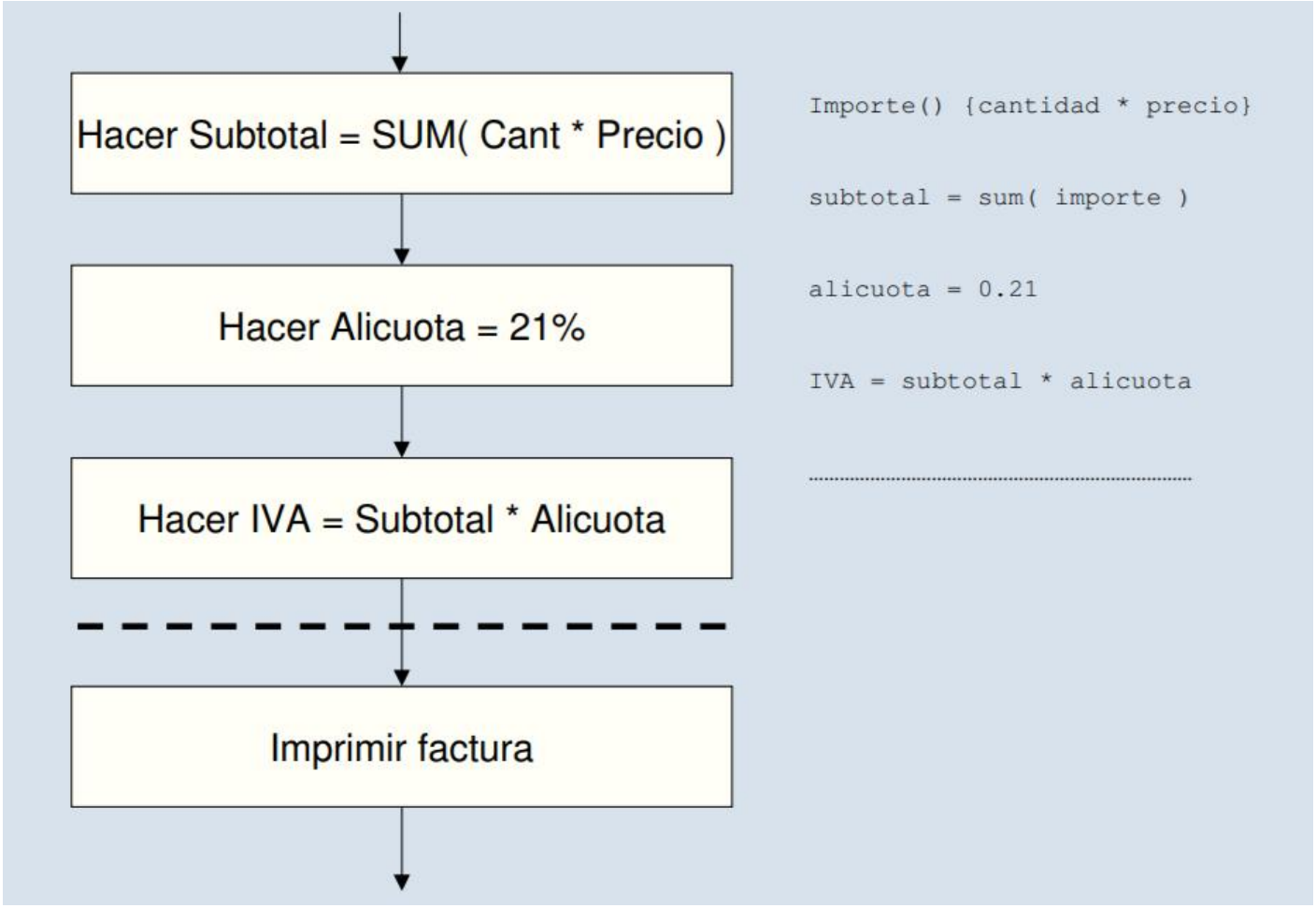
ORIENTADA A OBJETOS

- Modelar el problema basados en objetos reales.
- Asignarle a cada objeto un comportamiento.
- Resolver mediante colaboración e interacción entre objetos.

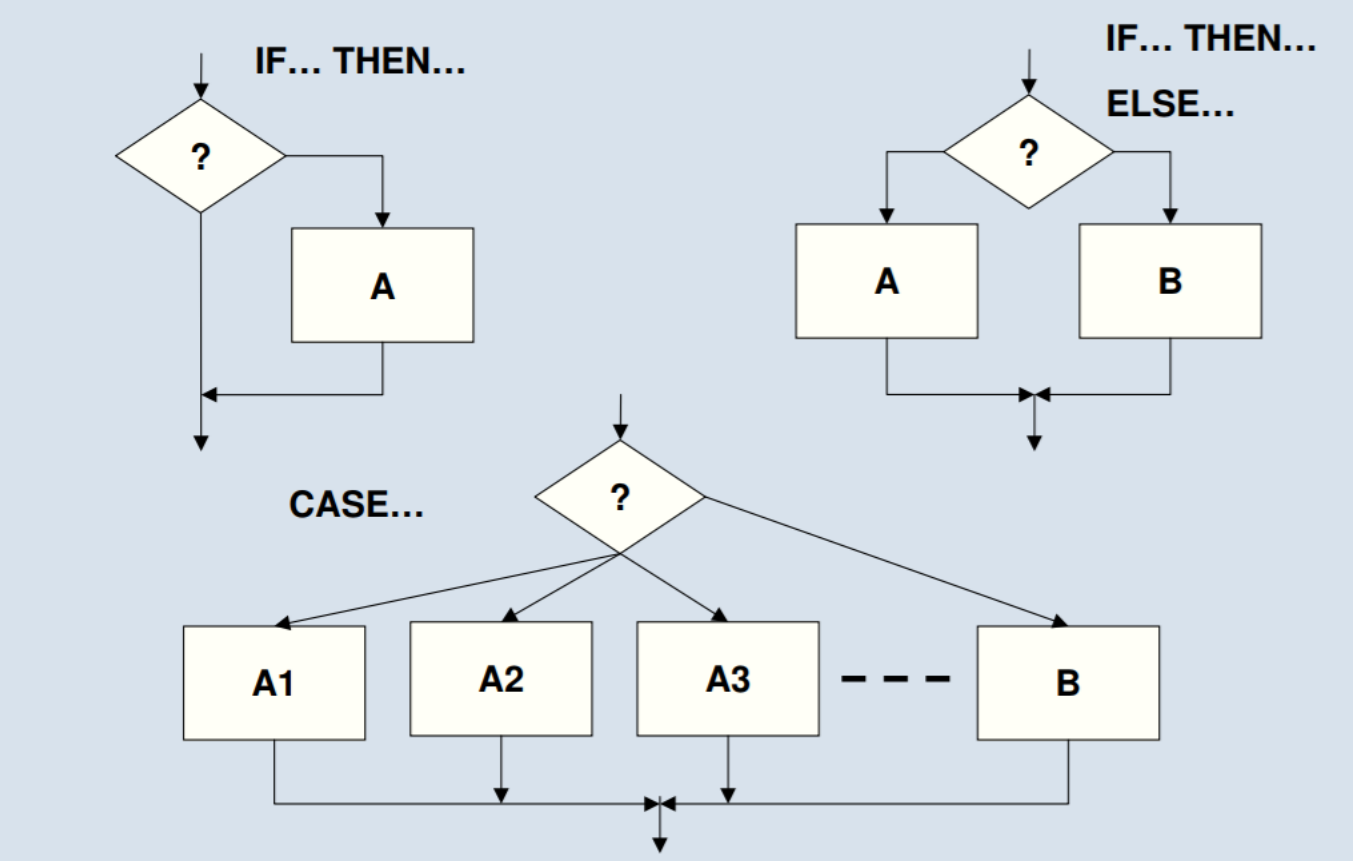
ESTRUCTURAS AÑADIDAS



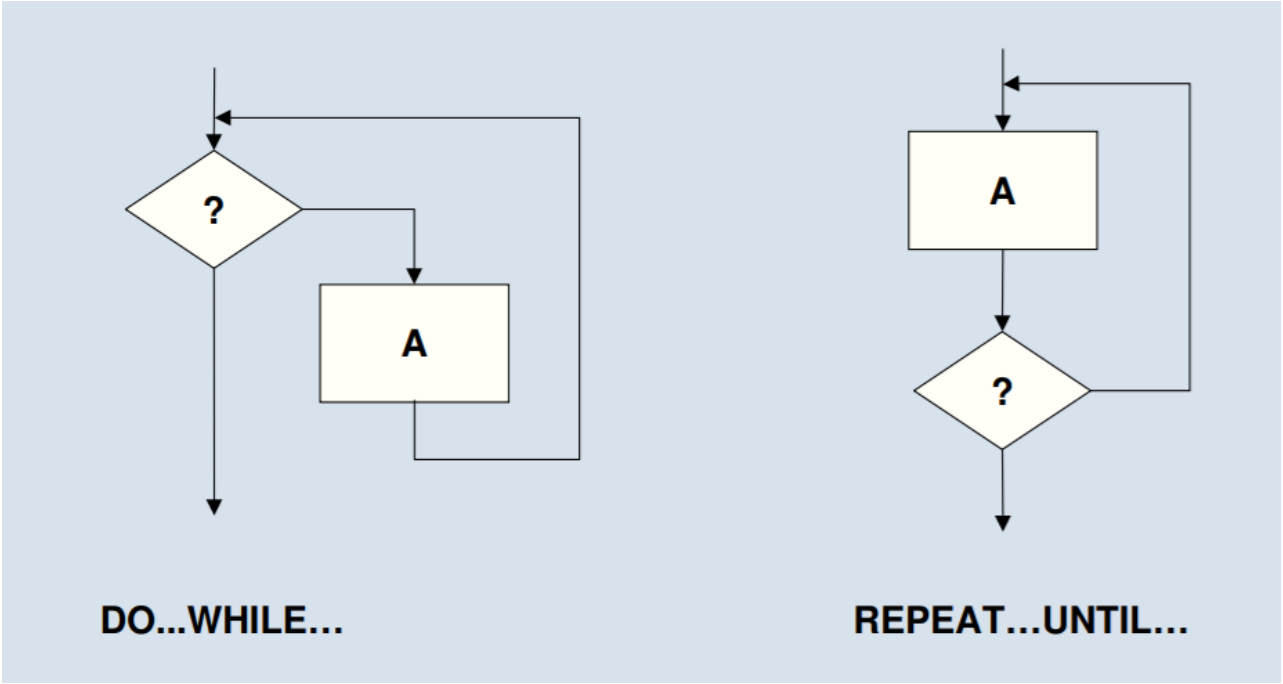
SECUENCIA


Hacer Subtotal = SUM(Cant * Precio)
Hacer Alicuota = 21%
Hacer IVA = Subtotal * Alicuota
Imprimir factura
Importe() {cantidad * precio}
subtotal = sum(importe)
alicuota = 0.21
IVA = subtotal * alicuota

DECISIÓN



ITERACIÓN



MANEJO DE ERRORES

Dependiendo el tipo del lenguaje los errores pueden detectarse

- **Como cualquier otra condición (ej: “IF ERROR THEN X”)**
- **Manejarse como una excepción: el control del programa pasa a una sección especial (el “error handler”).**

Esto último permite separar el manejo de errores y excepciones del flujo normal del programa.

- **El mantenimiento del programa se hace más fácil.**

INSTRUCCIONES & MÉTODOS

Es lo que "podemos hacer con un programa"

Lenguajes tradicionales: Instrucciones

- **READ RCONT01, WRITE TABLA**
- **PRINT TOTAL, PERFORM SALDO EOF, etc.**

Están dadas por el fabricante- conjunto limitado.

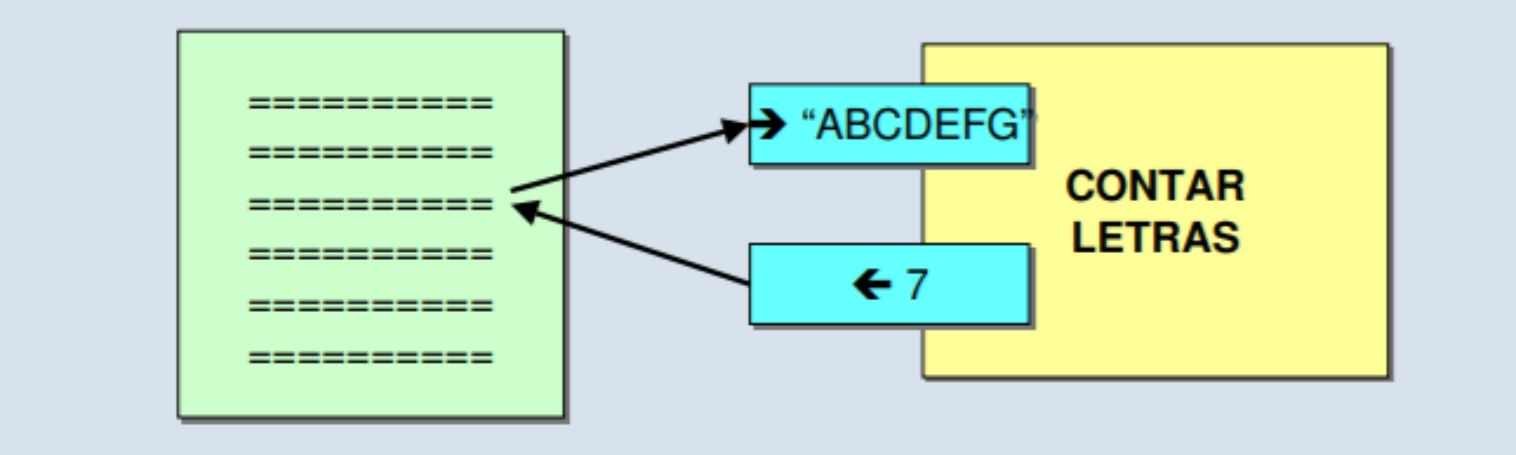
Lenguajes modernos: Métodos

- **Compra.calcularTotal(), document.print(), window.close()**
- **Cliente.setScoring(149), canvas.drawCircle(100,100,50, ROJO)**

No hay límite, es como extender el lenguaje

PROCEDIMIENTOS & FUNCIONES

- **Son bloques ó grupos de instrucciones que producen un resultado específico.**
- **Imprescindible para la buena organización del código.**
- **Idealmente trabajan con parámetros, no afectan variables globales (mala práctica!)**



PRÁCTICA

Revisar CÓDIGO FUENTE y reconocer

- **Métodos.**
- **Procedimientos.**
- **Bloques.**
- **Funciones.**

INTERFAZ USUARIO

- **La mayoría de las aplicaciones software tendrán que interactuar con una persona : EL USUARIO.**

- **El componente que maneja esta interacción es la llamada INTERFAZ del usuario: HOMBRE- MÁQUINA.**

DIPOSITIVOS

La interacción con el USUARIO del sistema puede realizarte a través de distintos mecanismos:

- **Monitor, impresora, fax, cajero de dinero, brazo robótico.**
- **Teclado, pantalla, táctil, botones, sensores, etc.**

Existen una amplia variedad de dispositivos para interactuar:

- **PC, Workstation, terminal “boba” (texto verde).**
- **Personal Digital Assistant PDA (Ej: Palm))**
- **Teléfonos (?) celulares.**
- **Paneles de control (botones + display + altavoz)**

INTERFAZ DEL USUARIO

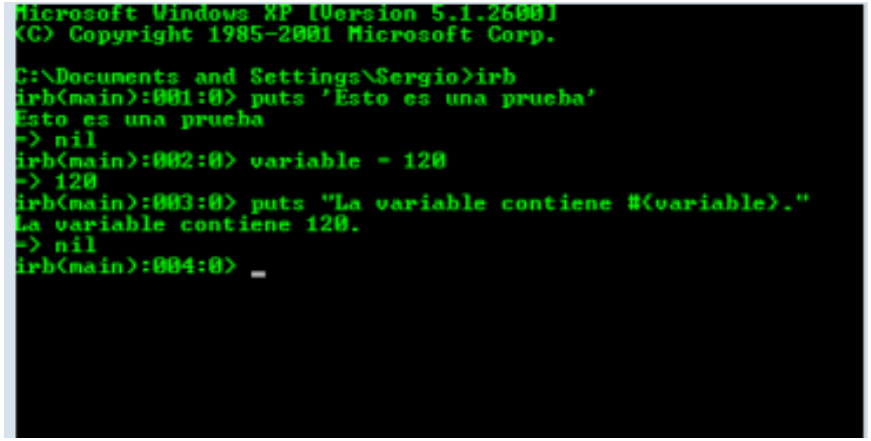
Es el componente más CRÍTICO en el diseño del sistema, porque es la "cara visible".

- **Determina qué puede hacer y qué NO puede hacer el sistema.**
- **Requiere balancear cuidadosamente:**
  - **Lo que el usuario quiere porque lo vio en otra aplicación.**
  - **Lo que el usuario quiere porque se imagina que sería útil.**
  - **Lo que el diseñador propone porque cree que es mejor.**
  - **Lo que la moda y el mercado imponen porque es más fashion.**
- **Es una especialidad en sí misma, puede ser encargada a un diseñador gráfico con experiencia.**

TIPOS DE INTERFAZ

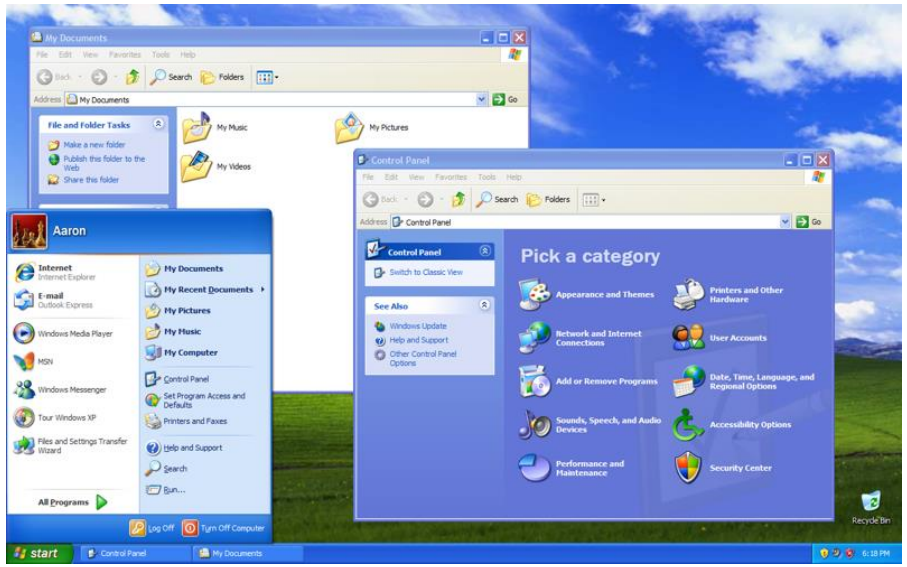
Para los dispositivos de pantalla podemos identificar dos tipos de interfaz

- **INTERFAZ TEXTUAL (Green screen ó terminal boba): Era el modo predominante antes de la difusión masiva de las PCs y las Macintosh.**
  - **Orientado a la linde de comando (ej:DOS): toda la interacción se produce en la última línea, el resto es historia previa.**
  - **Orientado a pantalla (ej: 3272/5250): la interacción se realiza enviando y recibiendo la pantalla completa (24*80) como un solo bloque.**



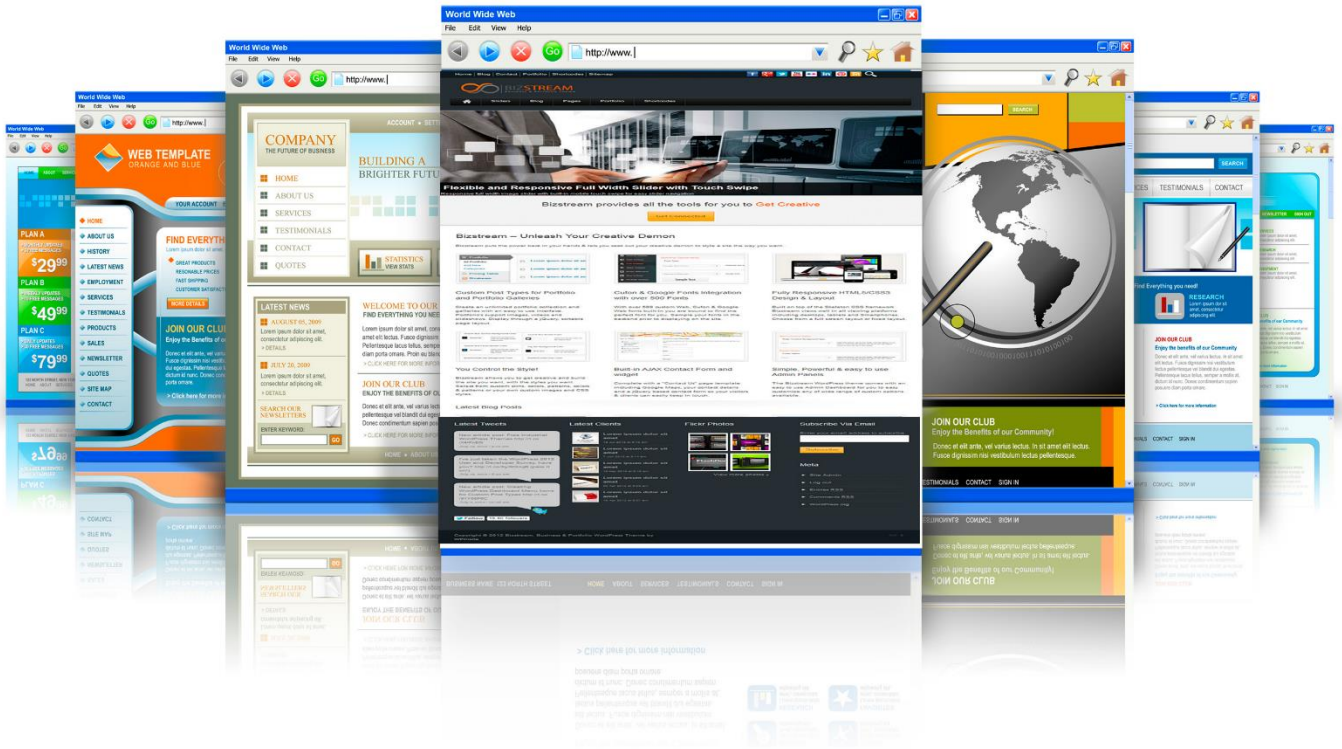
TIPOS DE INTERFAZ

- Interfaz gráfica (GUI): (ej: Windows)
  - Estándar.
  - Mayor riqueza visual.
  - Mayor productividad, facilidad de uso, ergonomía



TIPOS DE INTERFAZ

- Interfaz gráfica (WEB PAG)
  - Futuro cliente, liviano: cliente universal.
  - Fácil poı
  - Reinven



TRABAJANDO CON DATOS

¿QUÉ ES Y PARA QUÉ SE USA?

- Almacenamiento.
- Registros, tablas, archivos.
- Modelado de entidades.
- Entidades, atributos, relaciones.
- Tipo de relaciones.
- Subtipos y supertipos.
- Tablas.
- Claves.
- Clave primaria, claves alternativas.
- Diagrama de entidades y relaciones.
- XML.

ALMACENAMIENTO

Siempre necesitamos contar con datos

- **Datos de referencia permanentes (ej: cliente).**
- **Movimiento y transacciones (ej: pedido).**
- **Resultados de un proceso (ej: liquidaciones)**
- **Resultados intermedios, pasando de un programa a otro.**

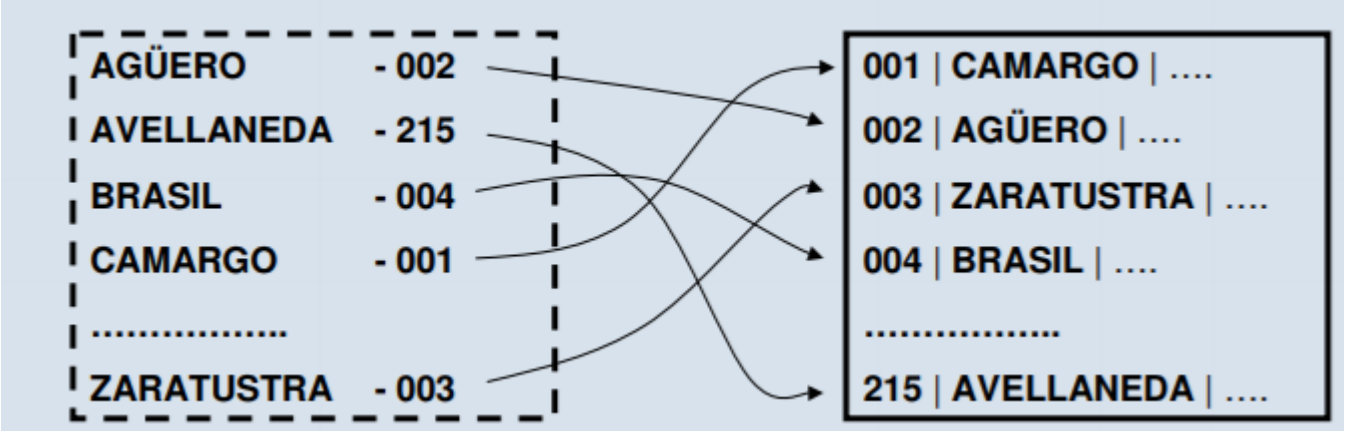
Coma la memoria de la máquina es volátil, necesitamos mecanismo de almacenamiento (medios magnéticos) y archivo (medios magnéticos/ ópticos)

Posibles tecnologías:

- **Archivos “planos”.**
- **Base de datos:**
  - **Jerárquica.**
  - **Relacional.**

ARCHIVOS “PLANOS”

- **Datos organizados en un solo nivel (plano).**
- **Acceso típicamente secuencial.**
- **Optimizado a través de índices.**



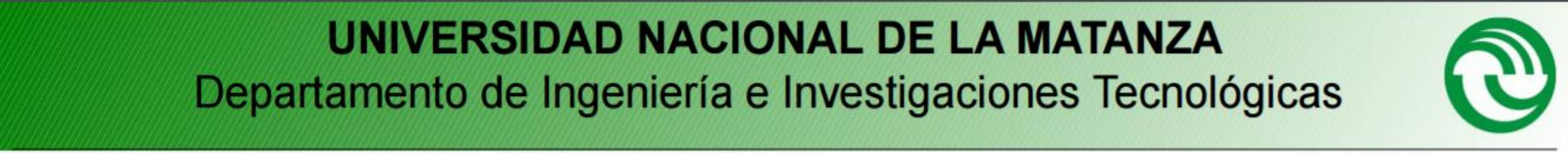
BASE DE DATOS (DBMS)

En realidad es un *motor* de base de datos.

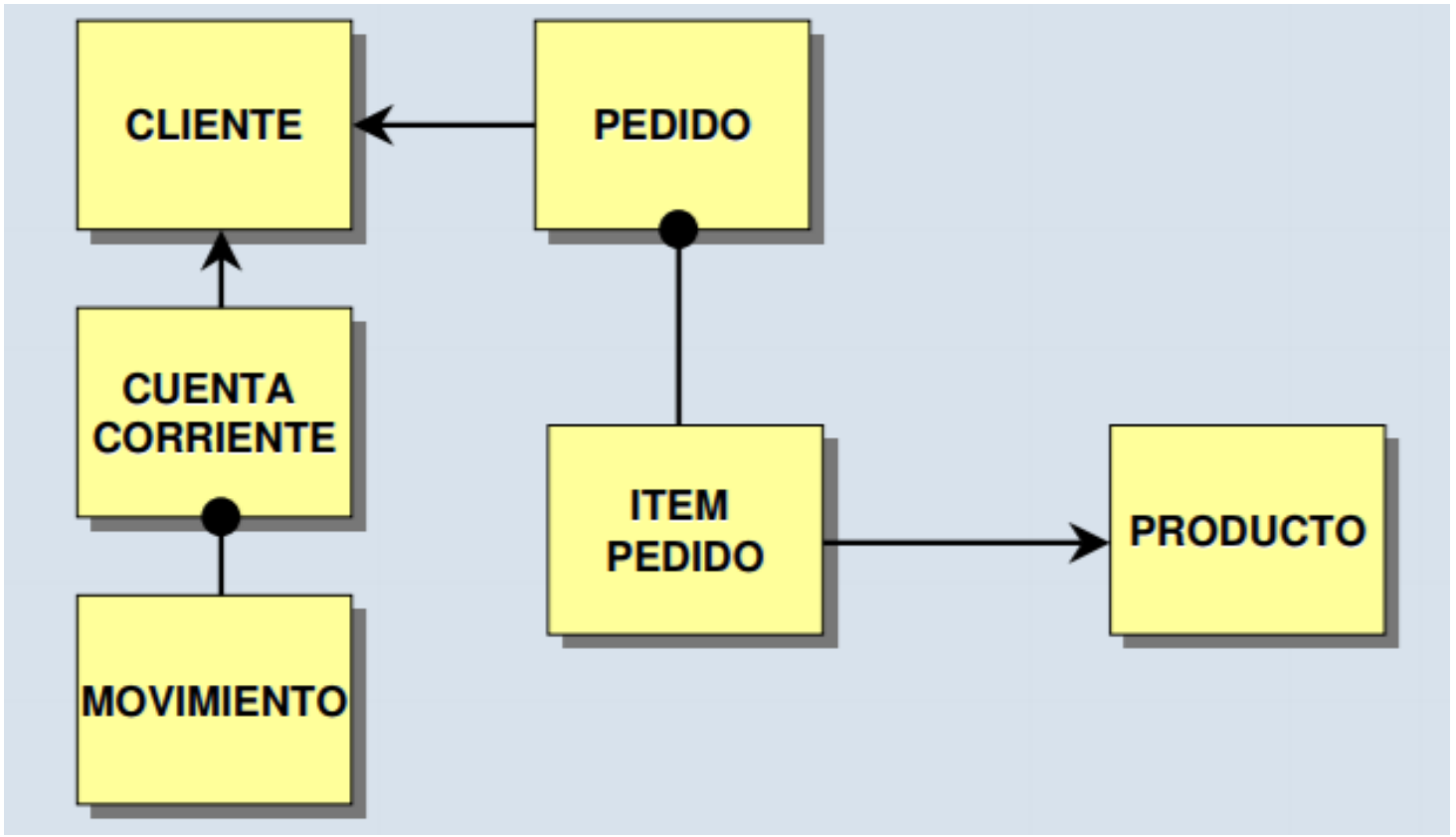
- **Maneja la operaciones de inserción, cambio, modificación y remoción de datos.**
- **Organiza los datos en tablas relacionales.**

DATOS ALMACENADOS

ARCHIVOS “PLANOS”	TABLAS RELACIONALES
Para acceder a los datos es necesario conocer la forma que fueron gravados	Incluye información sobre su contenido (metadata)
Los datos se acceden en forma secuencial, eventualmente a través de los índices.	Los datos se acceden luego de haber seleccionado los registro de interés.



UN MODELO (PARCIAL)



ENFOQUES DE PROGRAMACIÓN

¿CÓMO ENCARAR EL PROBLEMA?

Hay varias formas de estudiar un problema y diseñar su solución:

- **PROCEDURAL: el tradicional conjunto de instrucciones. Es el titiritero (programador) quien maneja los hilos.**

- **ORIENTADO A OBJETOS: Modela un universo de objetos con comportamiento propio interactuando entre sí. El programador define el comportamiento de cada objeto y deja que entre todos hagan su trabajo.**

ENFOQUE PROCEDURAL

Tanto el problema como la solución pueden descomponer en partes cada vez más siples.

La solución es una secuencia formada por tres tipos de bloques (programación estructurada):

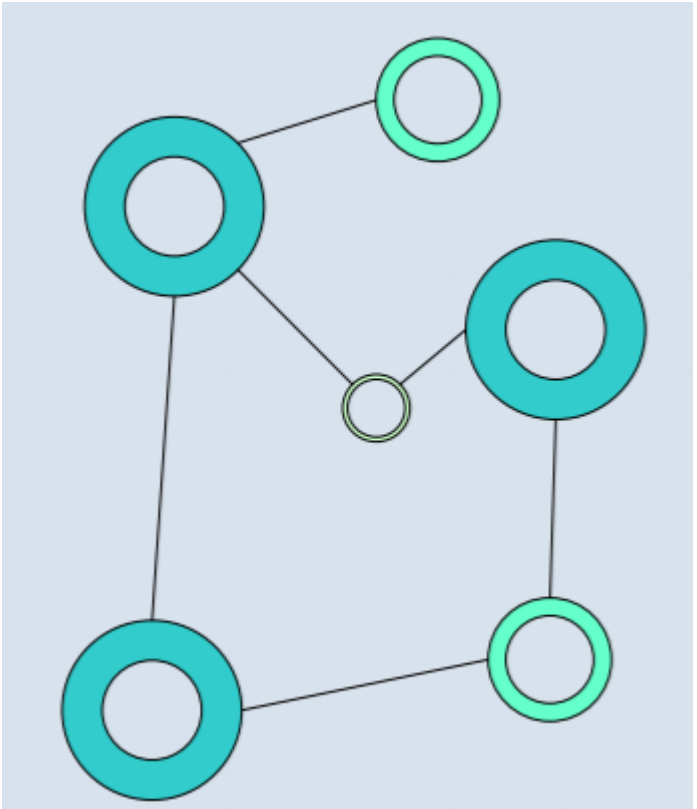
- **SECUENCIA: las instrucciones se ejecutan en el orden en que están escritas.**
- **DECISIÓN: al entrar en este bloque, el valor de una variable determina qué bloques (s) se ejecutarán a continuación.**
- **INTERACIÓN: al entrar en este bloque, el valor de la variable determina cuántas veces se ejecutará el contenido del mismo.**

ENFOQUE PROCEDURAL

- **Es el más arraigado de los enfoques.**
- **Es ideal para problemas sencillos, mecánicos, determinísticos.**
- **Su complejidad crece exponencialmente.**
- **Es difícil de apreciar en toda su extensión, se pierde el contexto.**
- **Difícil de particionar para trabajar en equipo.**

ORIENTACIÓN A OBJETOS

- **Objetos con un comportamiento y una responsabilidad específica colaborando entre sí.**
- **Se comunican mediante el envío de mensajes.**
- **Cada mensaje es un pedido para que le otro objeto haga algo o responda con otro objeto.**



ENFOQUE OO

- **Permite modelar mejor el mundo real: solución alineada con la realidad.**
- **Permite particionar el problema en forma natural, no arbitraria.**
- **Cada elemento es reutilizable, si está bien diseñado.**
- **Es mucho más complejo que el enfoque procedural.**
- **Ideal para problemas complejos, orgánicos (ej. Organizaciones).**

ELEMENTOS DE OO

- **OBJETOS: son componentes básicos.**
- **CLASES: Definen las características de cada objeto; instancia de clase.**
- **MENSAJE: Invoca un “método” en el objeto, con o sin datos (argumentos).**
- **INTERFAZ: Conjunto de métodos públicos, es el “contrato” entre un objeto y su entorno.**