Universidad Zaragoza

Fundamentos de Informática

Lección 1. Conceptos Básicos de Programación

Universidad Zaragoza

Curso 2010-2011

José Ángel Bañares, Pedro Álvarez
20/09/2010. Dpto. Informática e Ingeniería de Sistemas.

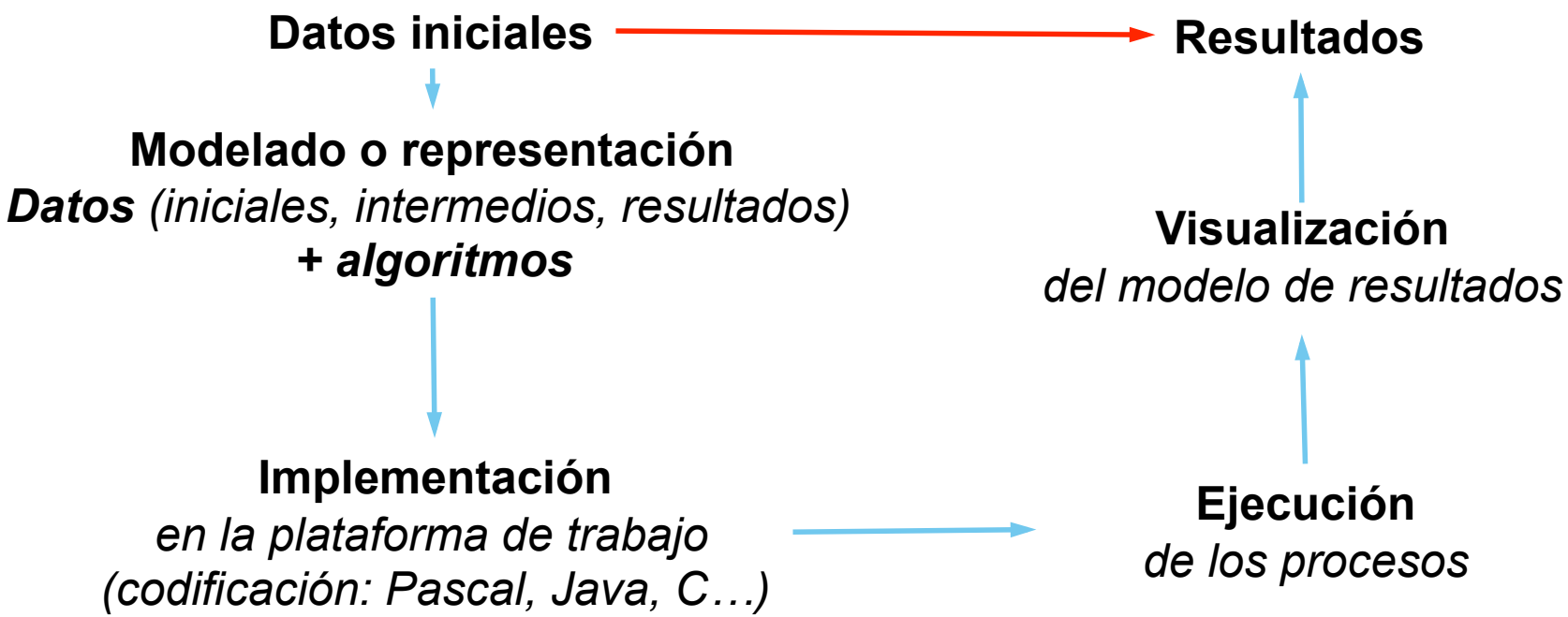
Universidad Zaragoza
1542

Conceptos Básicos de Programación

- Problemas y soluciones
- Objetivos de la programación
- Noción de algoritmo
- Noción de Programa
  - Estilo de programación
  - Lenguajes de Programación
  - Elementos de un Lenguaje de programación
- El computador (hardware)
- Propiedades de los algoritmos
- Entorno de Programación

Universidad Zaragoza
1542

Problemas y soluciones

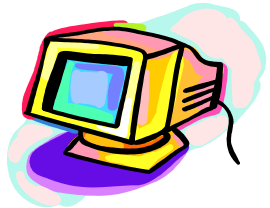


Universidad Zaragoza
1542

Objetivo de la programación

- Problemas que se tratan de resolver en Programación
  - Cómputo o cálculo
  - Gestión comercial
  - Control
  - Tratamiento de la señal
  - Lúdicos
  - Inteligencia Artificial
  - *E-bussines*
  - Etc.

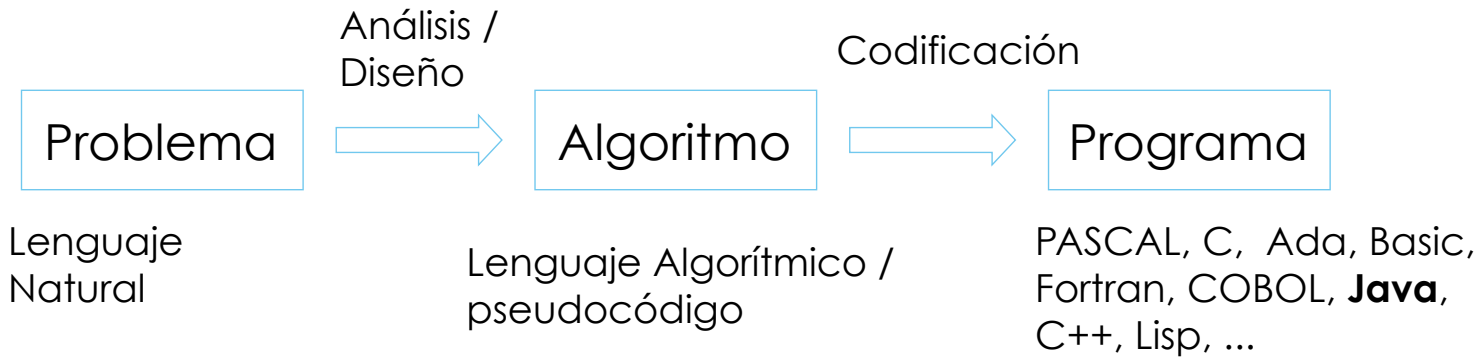
- Problemas cuya solución puede ser llevada a cabo por un COMPUTADOR
  - *Computador: Herramienta que permite tratar de forma automática problemas de* ***tratamiento de la información****.*
  - *Informática: Ciencia del tratamiento de la información.*



Universidad Zaragoza
1542

Objetivo de la programación: Construir programas

- Programa: texto formado por instrucciones para que una máquina las ejecute
- ¿Cómo transformar un problema en un programa ejecutable por un computador



Universidad Zaragoza
1542

Noción de algoritmo

- **Acción**: Es un acontecimiento producido por un actor (o ejecutante de una acción) que tiene lugar durante un periodo de tiempo finito y que tiene como consecuencia o resultado una nueva situación bien definida y previsible.

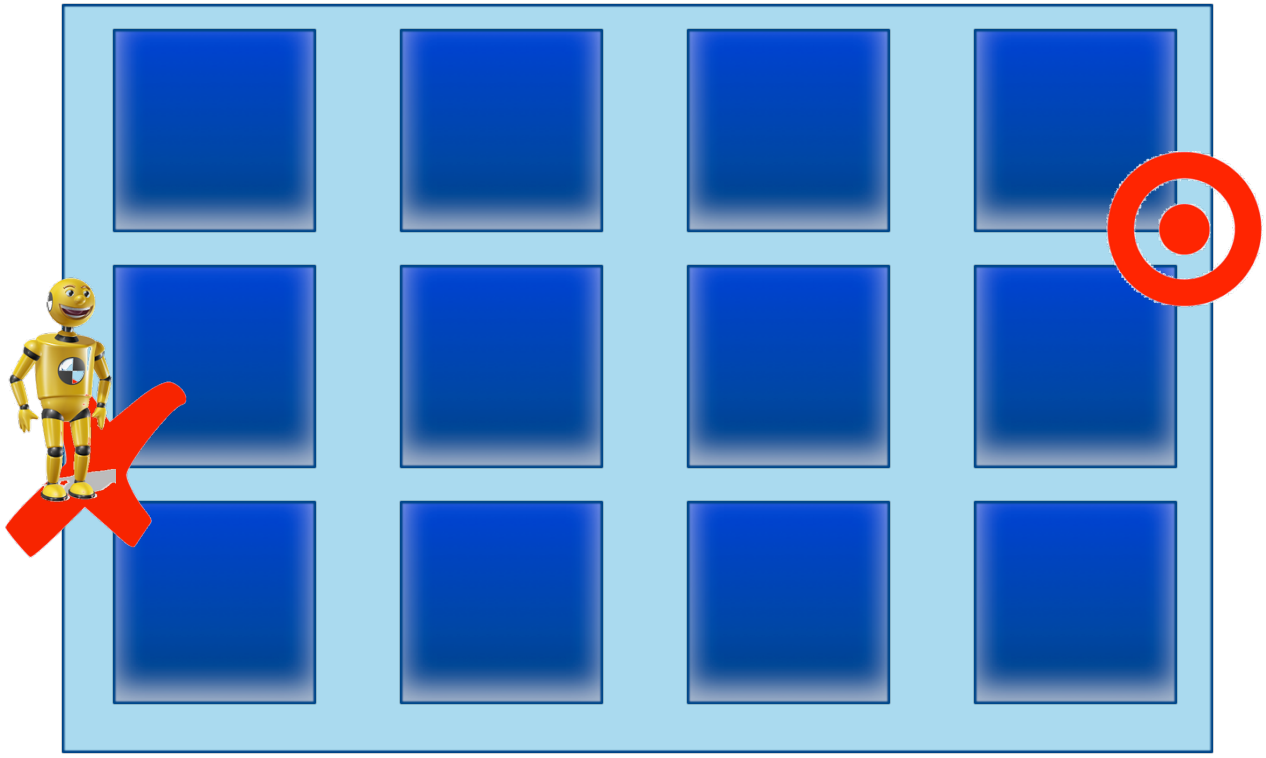
- **Estado**: Es el conjunto de objetos (con sus circunstancias) disponibles en un instante determinado.

- **Algoritmo** (Proceso): Es la descripción de una sucesión finita de acciones que permite transformar el entorno del estado inicial dado en el final deseado.

Universidad Zaragoza
1542

Noción de algoritmo

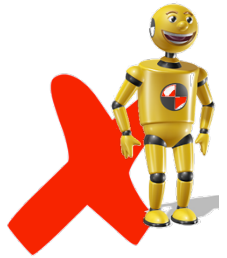
Problema



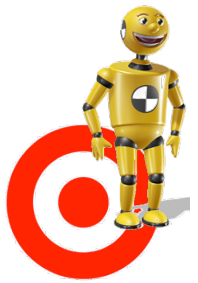
Acciones:

- Avanza
- Retrocede
- Izquierda
- Derecha

- Estado Inicial



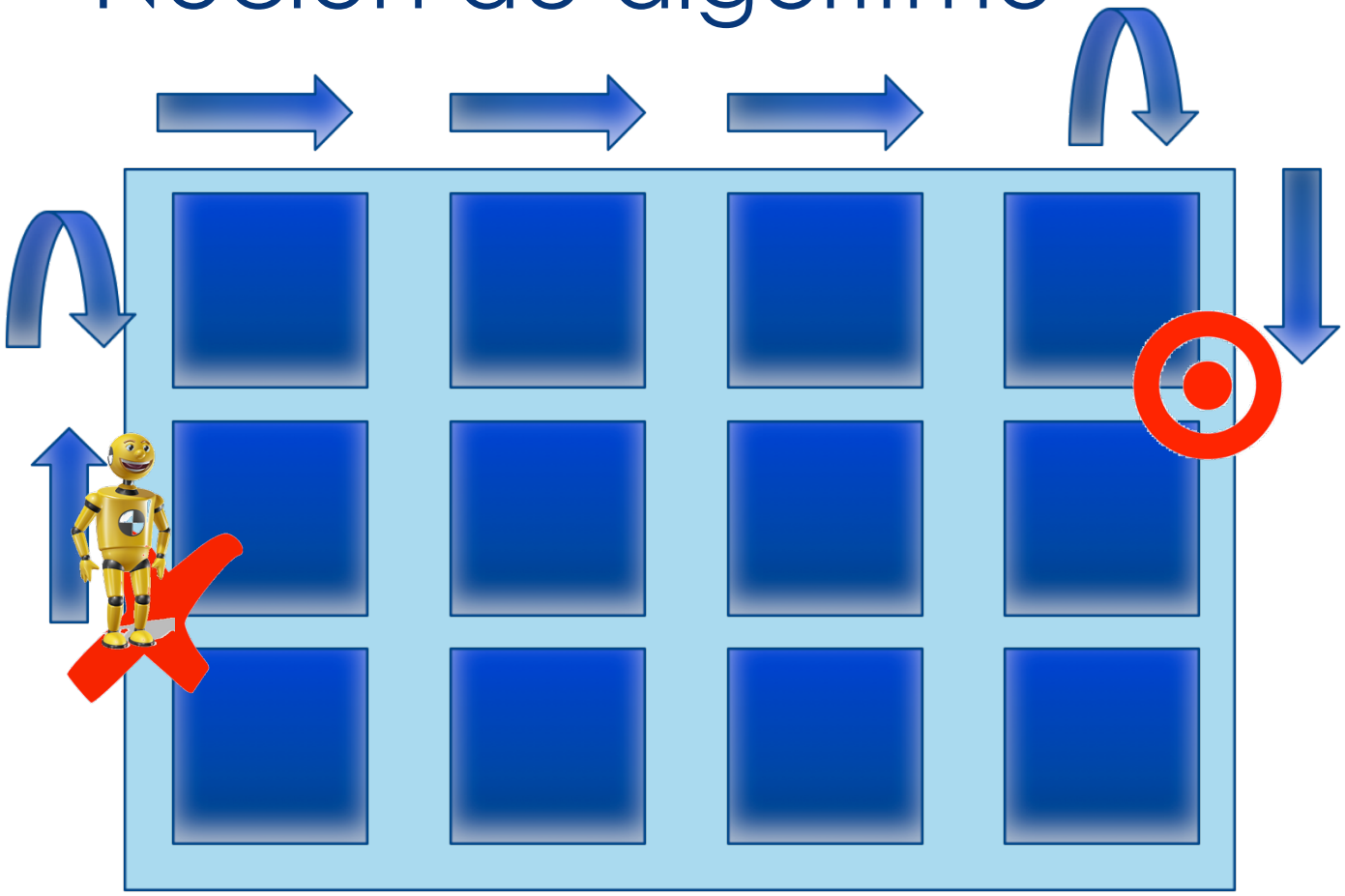
- Estado final



Universidad Zaragoza
1542

Noción de algoritmo

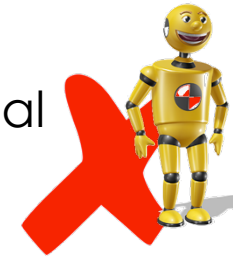
Análisis



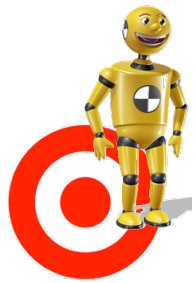
Acciones:

- Avanza
- Retrocede
- Gira

- Estado Inicial



- Estado final



Universidad Zaragoza
1542

Diseño de Programas

- Programas = Algoritmos + Estructuras de Datos[1]

  - La **abstracción de datos** es una metodología que nos permite representar un "objeto" quedándonos con los aspectos relevantes del problema y ocultando los de detalles de cómo se construye.
  - La **abstracción de acciones** es una metodología permite definir "acciones complejas" mediante la combinación de acciones simples
  - Los **Algoritmos** representan procesos que combinan acciones que manipulan las abstracciones de datos.
  - La evolución de los procesos descritos por los algoritmos vienen dirigidas por un **conjunto de reglas** expresadas en un **programa**.

[1] *Algorithms + Data Structures = Programs es un libro escrito por Niclaus Wirth*

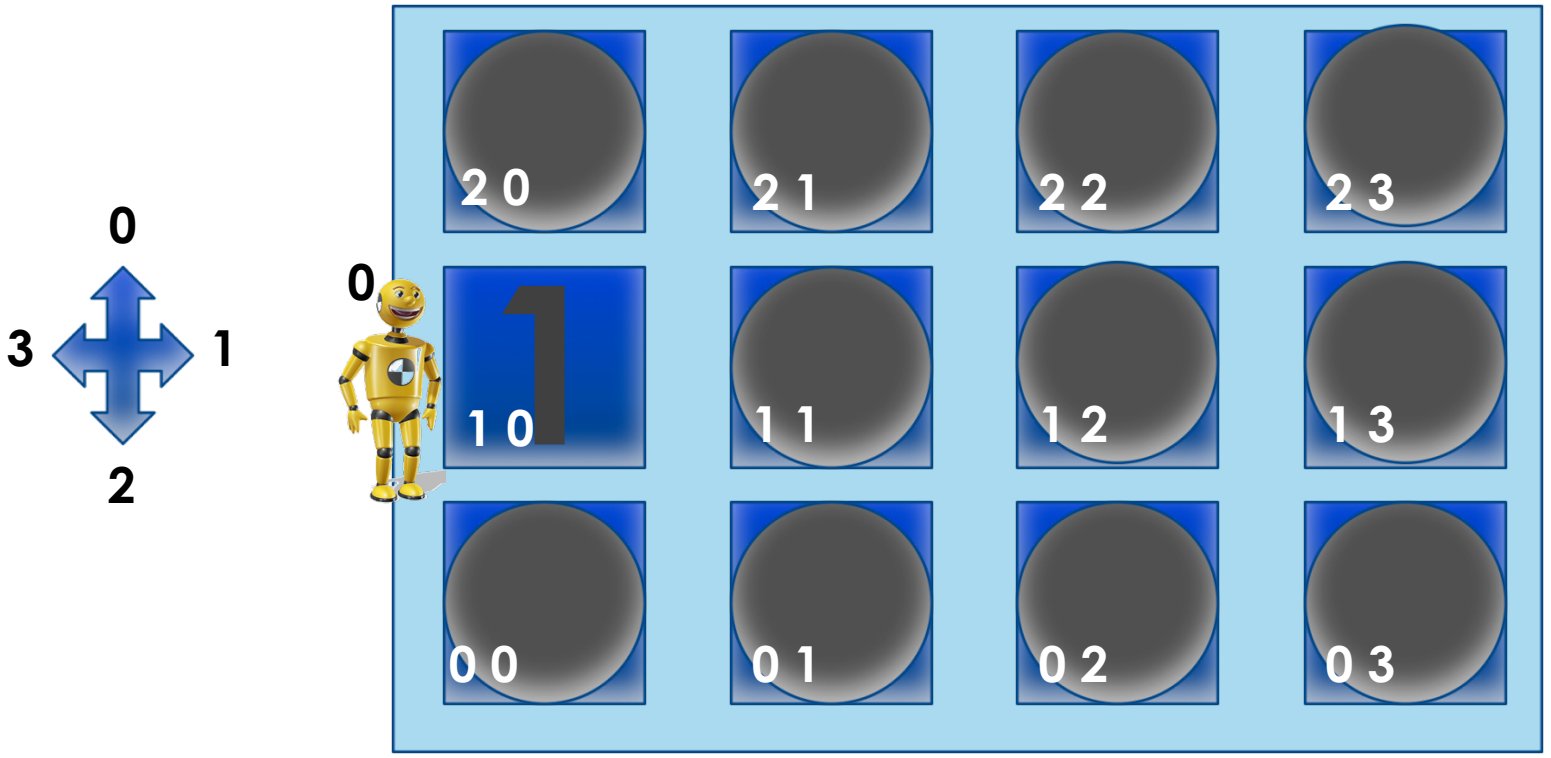
Universidad Zaragoza
1542

Abstracción (DRAE)

- **abstraer.(Del lat. *abstrahĕre).1. tr. Separar por medio de una operación intelectual las cualidades de un objeto** *para considerarlas aisladamente o para considerar el mismo objeto en su pura esencia o noción.*

Abstracción de datos

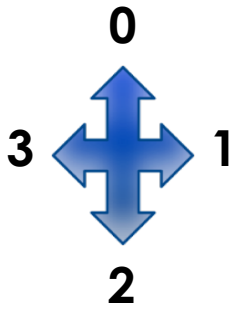
Diseño

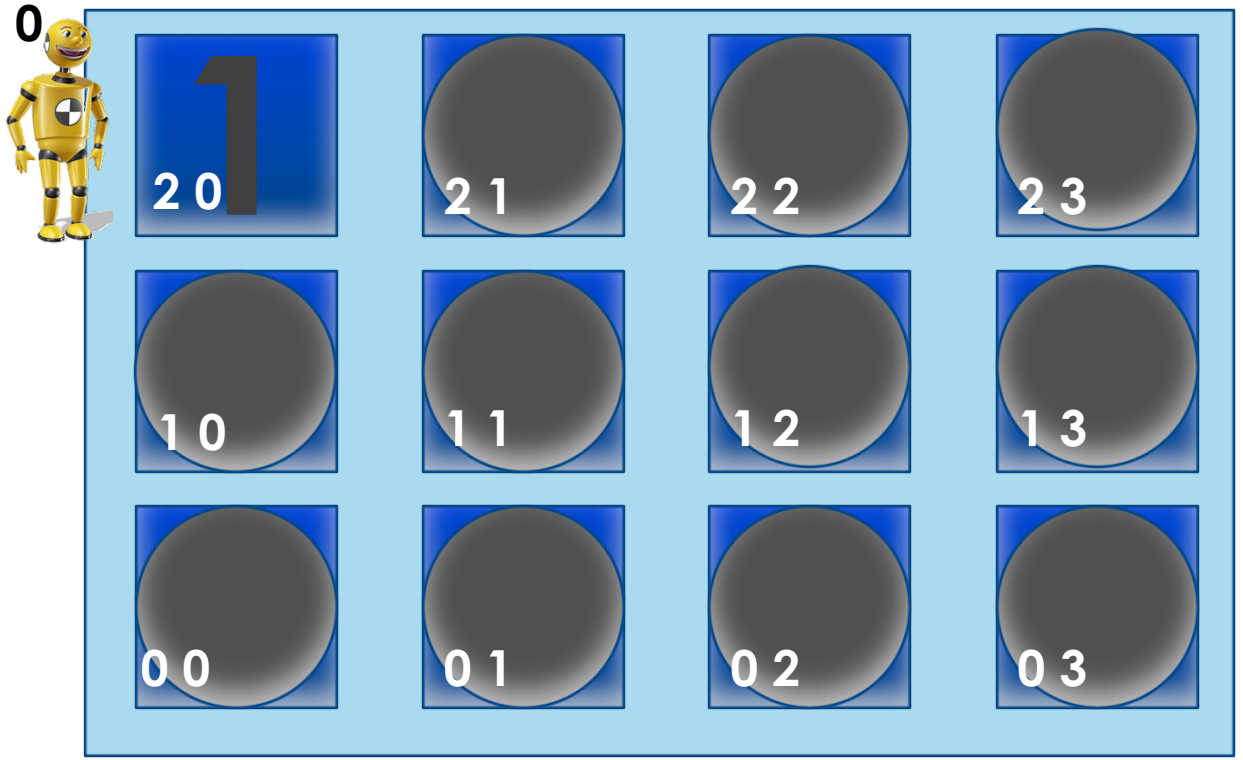


Universidad Zaragoza
1542

Abstracción de datos

Diseño

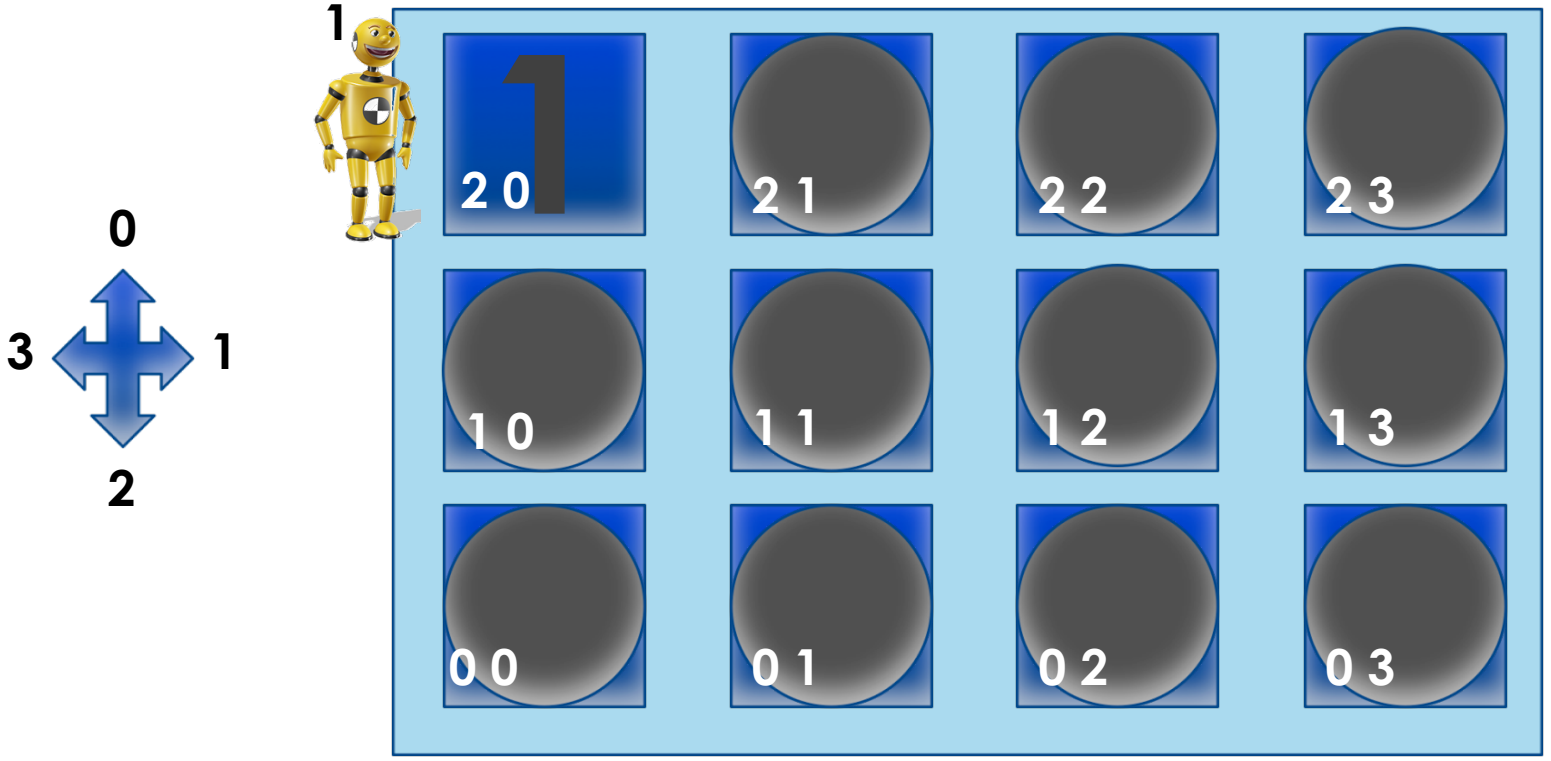




Universidad Zaragoza
1542

Abstracción de datos

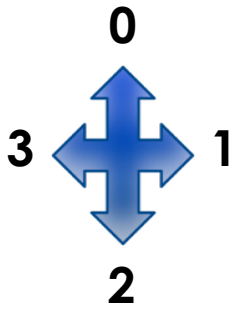
Diseño

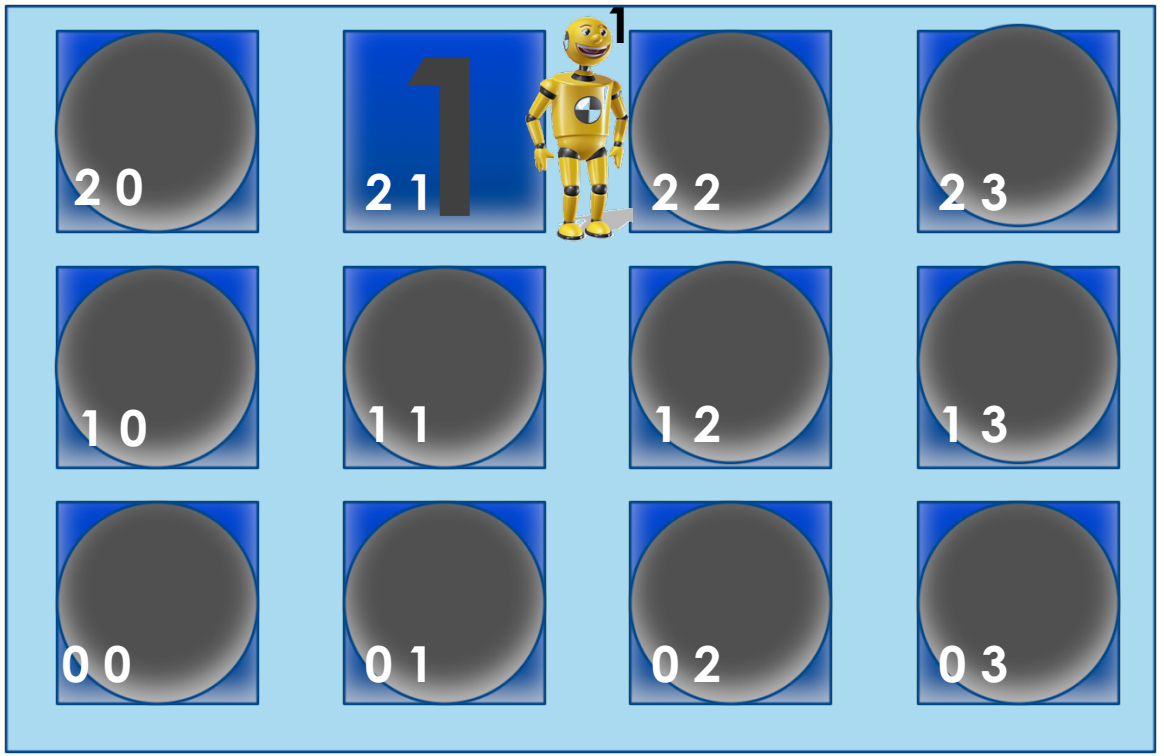


Universidad Zaragoza

Abstracción de datos

Diseño

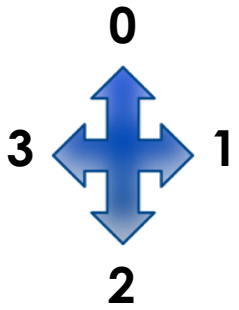


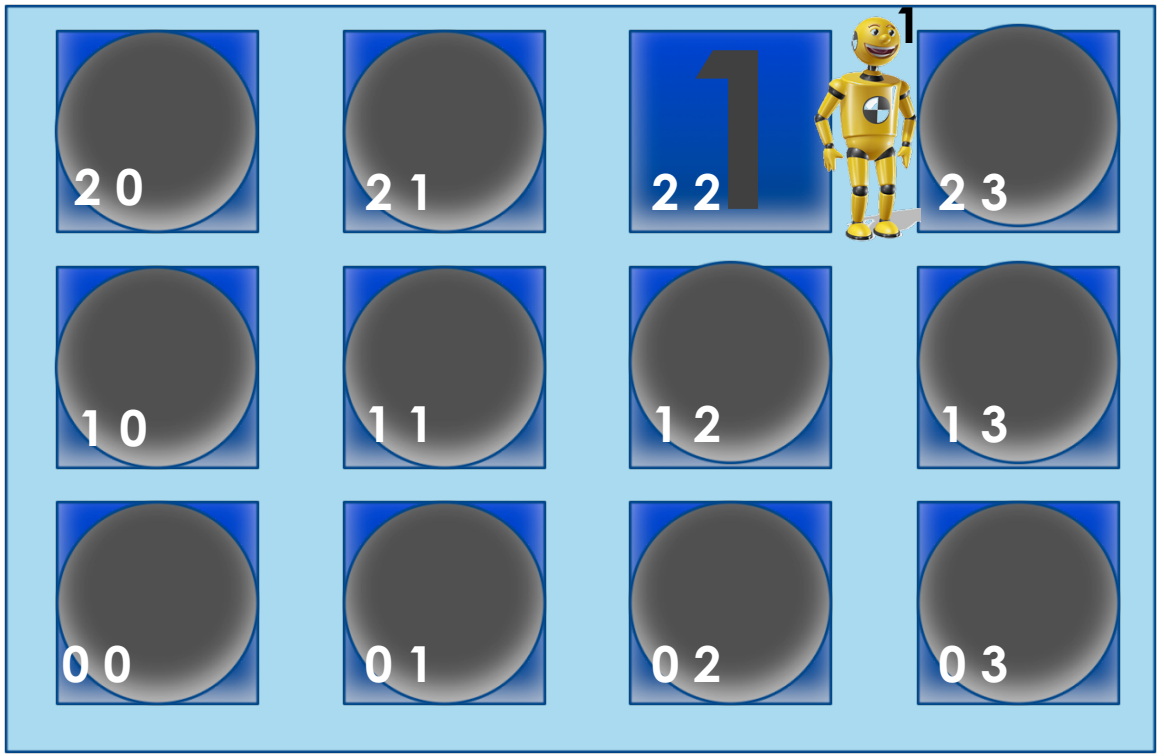


Universidad Zaragoza
1542

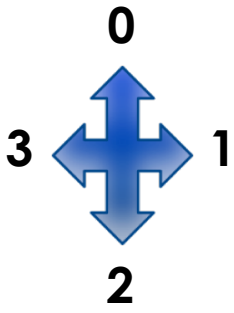
Abstracción de datos

Diseño





Universidad Zaragoza
1542



Abstracción de Acciones

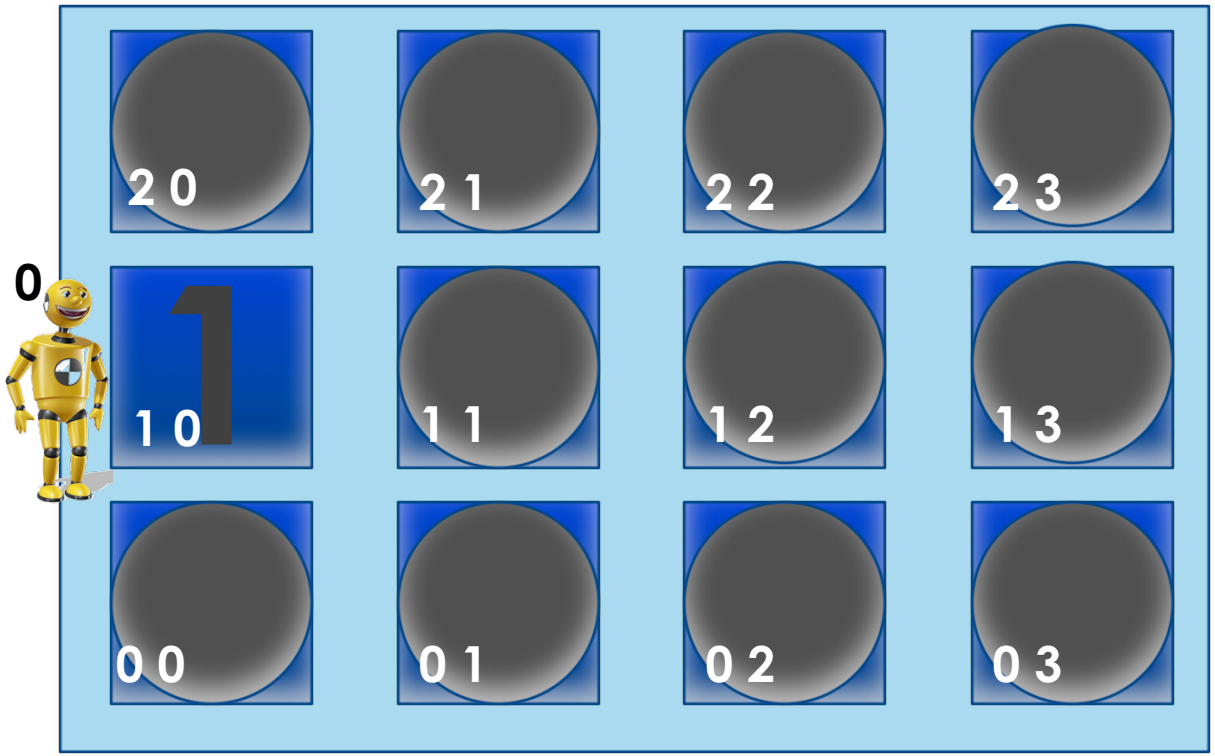
Diseño

Avanza

Si Orientación= 0
 y = y-1
Si Orientación = 1
 x = x+1
Si Orientación = 2
 y = y+1
Si Orientación = 3
 x =x-1

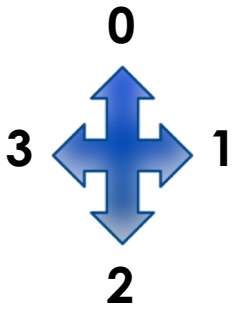
Gira

Orientación =
 Orientación+1 % 4



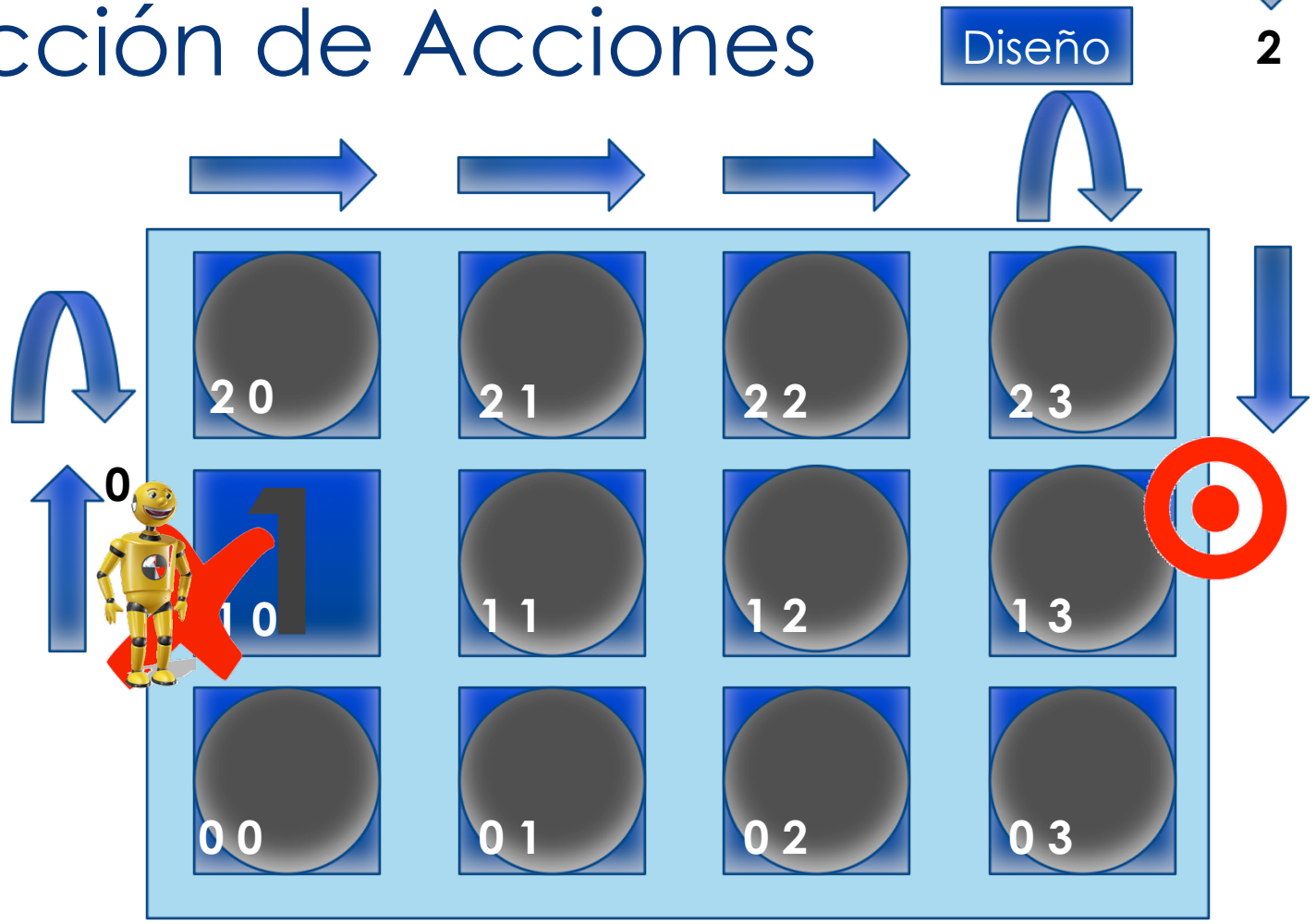
Universidad Zaragoza
1542

Abstracción de Acciones



Algoritmo

Avanza
Gira
Avanza
Avanza
Avanza
Gira
Avanza



Universidad Zaragoza
1542

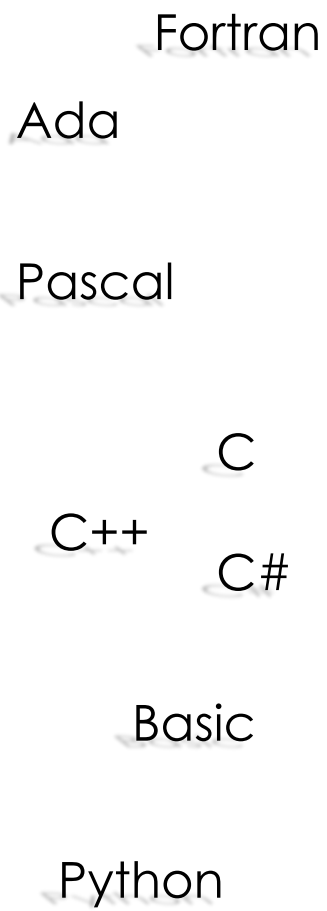
Programa

- Un lenguaje de programación sirve para organizar nuestras ideas relacionadas con un proceso (algoritmo)
- Todo lenguaje de programación tiene tres mecanismos para lograrlo[2]:
  - **Expresiones primitivas:** Representan los elementos más simples que el lenguaje es capaz de tratar
  - **Medios de combinación:** Por la que podemos componer elementos a partir de otros más simples.
  - **Medios de Abstracción:** Por la que los elementos compuestos pueden ser nombrados y manipulados como unidades.

Universidad Zaragoza
1542

Estilos de programación

- Programación Imperativa
  - Describe la programación en términos del **estado** del programa y **sentencias** que cambian dicho estado.
  - Los **programas imperativos** combinan instrucciones que indican al computador como realizar una tarea (**Algoritmo**)
  - **Programas = Algoritmo (receta) + Datos (ingredientes)**
- Influenciado por la arquitectura de la máquina
  - **variables**: Direcciones de memoria. Datos.
  - **Acción básica**: Asignación: Dar valor a una variable.
  - **Combinación básica de acciones**: Composición secuencia. Una instrucción detrás de otra.



Universidad Zaragoza
1542

Estilos de programación

- Programación declarativa
  - Describe la programación en términos del **estado** del programa y **reglas/acciones** que cambian dicho estado.
  - Los **programas declarativos** "encadenan" reglas desde el estado inicial al estado final.
  - **Programas = Acciones + Estado inicial + Estado final**
    - **¡No hay descripción del proceso! (No hay algoritmo)**

Clips

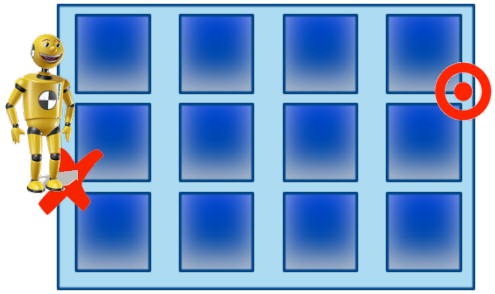
Ops5

Prolog

Jess

Universidad Zaragoza

Programación Declarativa



Se **exploran** posibles combinaciones de acciones que lleven al estado final.

Avanza
Gira
Avanza ***4 veces***
Gira
Avanza

¡No hay algoritmo!

Clips

Ops5

Prolog

Jess

Regla 1
Si Orientación= 0
Entonces y = y-1
Regla 2
Si Orientación = 1
Entonces x = x+1
Regla 3
Si Orientación = 2
Entonces y = y+1
Regla 4
Si Orientación = 3
Entonces x =x-1
Regla 5
Si Orientacion= ?Orientacion
Entonces Orientación = ?Orientación+1 % 4
Regla 6
Si Dummy en 1 3
Entonces Parar

Estado inicial
Dummy en 1 1, orientacion 0
Estado final
Dummy en 1 3

Universidad Zaragoza
1542

Programación funcional

ML
Miranda
Lisp
Matemática
Erlang

- No existencia de asignaciones de variables
- Carencia de construcciones estructuradas como la secuencia o la iteración

¡Vale! Hay muchos estilos de programación.
NOS centraremos en la programación Imperativa
    Programación estructurada
    Programación orientada a objeto

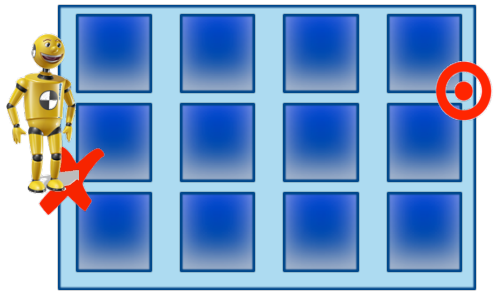
Universidad Zaragoza
1542

Programación Estructurada

Las reglas para un programa estructurado son:

- El programa tiene un **diseño modular**
- La **unidad de modularidad** son las abstracciones de **acción**
- Los módulos son **diseñados descendentemente**
- Cada módulo de programa se codifica usando tres estructuras de control (composición de acciones)
  - *Secuencia,*
  - *Selección*
  - *Iteración*

Programación Estructurada



Programa

Avanza
Gira
Avanza ***3 veces***
Gira
Avanza

Avanza

Orientación =
Orientación+1 % 4

Gira

Si Orientación= 0
y = y-1
Si Orientación = 1
x = x+1
Si Orientación = 2
y = y+1
Si Orientación = 3
x =x-1

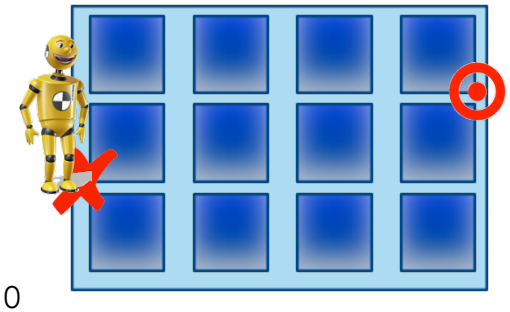
Universidad Zaragoza
1542

Programación Orientada a Objeto

- Las reglas para un programa orientado a objeto son:
  - El programa tiene un **diseño modular**
  - La **unidad de modularidad** son las abstracciones de datos. *Las **clases** de objeto.*
  - Cada módulo de programa se codifica especificando las acciones (**métodos**) que ofrece el objeto y ocultando las abstracciones de dato utilizadas.
    - Los servicios se programan utilizando las reglas de la programación estructurada.

Universidad Zaragoza
1542

Programación Orientada a Objeto



Si Orientación= 0
 y = y-1
Si Orientación = 1
 x = x+1
Si Orientación = 2
 y = y+1
Si Orientación = 3
 x =x-1

Orientación =
 Orientación+1 % 4

Avanza
Gira
Avanza ***3 veces***
Gira
Avanza

Programa

Dummy

Servicios
 Avanza
 Gira
 Mueve_objetivo

Mundo

Servicios
 Posición Dummies
 Dimensiones

Universidad Zaragoza
1542

Elementos de un Lenguaje

- Los elementos que definen un lenguaje son:
  - SIMBOLOS
    - Palabras clave = {IF, THEN, WHILE, …}
    - Caracteres = {'a',..'z','A'..'Z','#','?',…….. }
    - Dígitos = {0,..,9}
    - Otros símbolos = {'.' , ',' , ';' ,…}

Expresiones primitivas

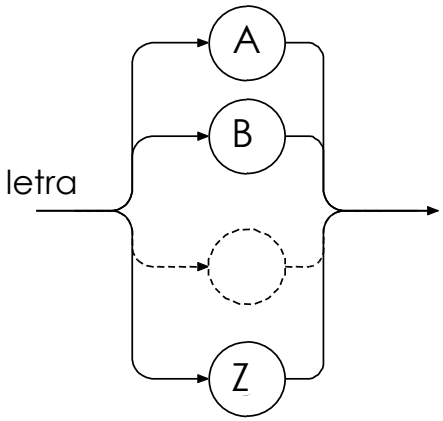
  - SINTAXIS
    - Conjunto de reglas sintácticas
    - Notaciones para expresar las reglas:
      - Backus-Naur
      - Grafo Sintáctico

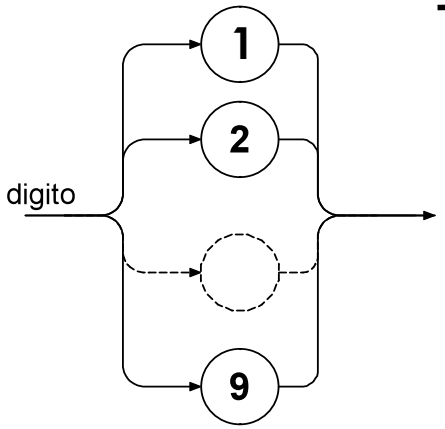
Medios de combinación

Universidad Zaragoza

Sintaxis

Grafos Sintácticos



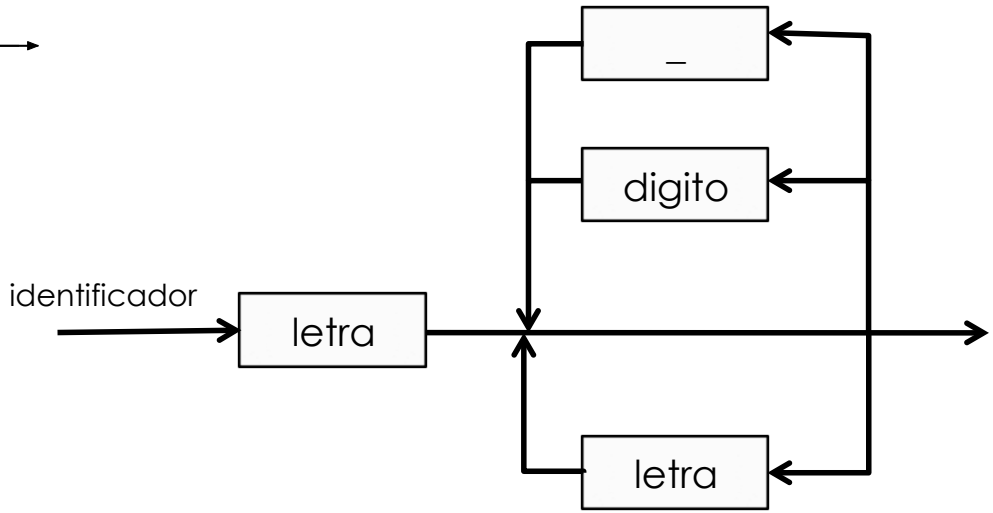


Backus-Naur

<letra>::= ‘A’ I ‘B’ I… I‘Z’

<digito>::= ‘0’ I ‘1’ I… I‘9’

<identificador>::= <letra> {<letra> I <digito> I’_’}

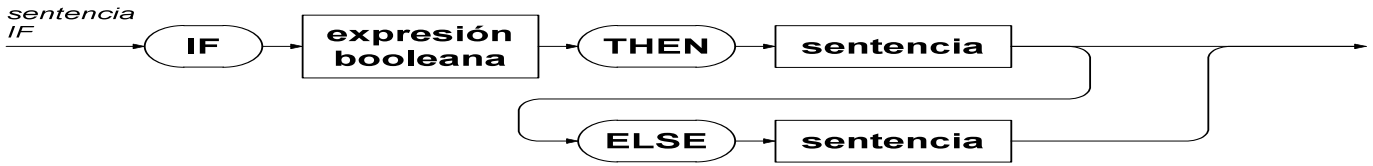


1 o más veces

Ejemplos identificadores válidos: P3Q NUMERO_PI
Ejemplo identificadores no válidos: 3PQ _numero

Universidad Zaragoza
1542

Sintaxis

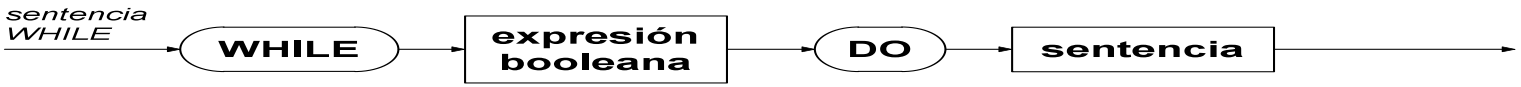


<Sentencia IF> ::=

IF <expresion booleana> THEN <sentencia>

[ELSE <sentencia>]

0 o 1 vez



<Sentencia WHILE> ::=

WHILE <expresion booleana> DO <sentencia>

Universidad Zaragoza
1542

Medios de abstracción


```
(defun suma (a b) (+ a b))
```

Definición de la función suma en Lisp

Definición de la función suma en Java

```
public int suma (int numero1, int numero2){
 int retsuma;
 retsuma = numero1 + numero2;
return retsuma;
}
```

Universidad Zaragoza
1542

Se da nombre a una acción

Medios de abstracción

Medios de abstracción

```
(defun suma (a b) (+ a b))
```

Definición de la función suma en Lisp

Definición de la función suma en Java

```
public int suma (int numero1, int numero2){
 int retsuma;
 retsuma = numero1 + numero2;
return retsuma;
}
```

Universidad Zaragoza
1542

Que agrupa acciones que manipulan datos

Medios de abstracción

Medios de abstracción

```
(defun suma(a b) (+ a b))
```

Definición de la función suma en Lisp

Definición de la función suma en Java

```
public int suma (int numero1, int numero2){
 int retsuma;
 retsuma = numero1 + numero2;
return retsuma;
}
```

Medios de abstracción

Definición del tipo Persona en Pascal

```
Persona= Record
           nombre:String[20];
           Fecha_nacimiento:tpFecha;
         end;
```

Definición del tipo Persona en Java

```
Public class Persona
{
    public String nombre;
    public Fecha_nacimiento:tpFecha
}
```

Universidad Zaragoza
1542

Se da nombre a un nuevo tipo de dato

Medios de abstracción

Definición del tipo Persona en Pascal

```
Persona= Record
           nombre:String[20];
           Fecha_nacimiento:tpFecha;
         end;
```

Definición del tipo Persona en Java

```
Public class Persona
{
    public String nombre;
    public Fecha_nacimiento:tpFecha
}
```

Universidad Zaragoza

Que agrupa otros datos (…y las acciones permitidas)

Medios de abstracción

Definición del tipo Persona en Pascal

```
Persona= Record
           nombre:String[20];
           Fecha_nacimiento:tpFecha;
         end;
```

Definición del tipo Persona en Java

```
Public class Persona
{
    private String nombre;
    private Fecha_nacimiento:tpFecha;
   public String getnombre {return "Nombre:" + nombre;}
}
```

Universidad Zaragoza
1542

Niveles de lenguajes de programación

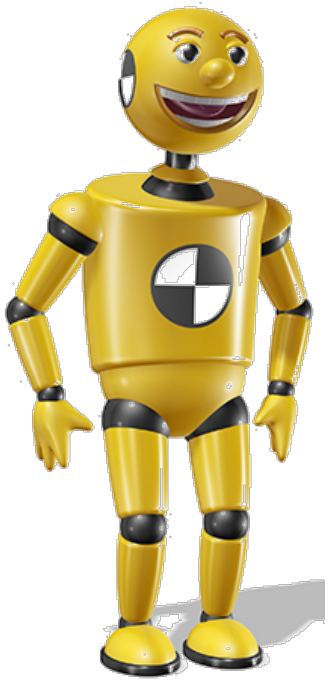
- Los lenguajes de programación de "Alto nivel"
  - Soportan abstracciones (conceptos), que entiende el programador. ¡Pero no la máquina!
- Los lenguajes de bajo nivel
  - Son los que utilizan las instrucciones básicas de un procesador concreto.
  - Lenguaje de máquina

Universidad Zaragoza
1542

Niveles de lenguaje de programación

Juego de instrucciones de Dummy

MI	A	R	MD	A	R	
1	1	0	1	1	0	Avanza 1 m
1	0	1	1	0	1	Retrocede 1 m
1	1	0	0	0	0	Gira Derecha 90 º
0	0	0	1	1	0	Gira Izquierda

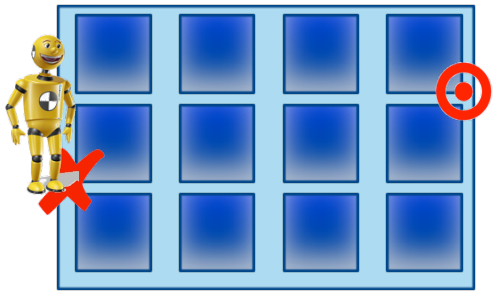


Universidad Zaragoza
1542

Niveles de lenguaje de programación

Problema

Dummy se encuentra en ✗*, mirando al norte, y tengo que llevarlo a* ◎



Programa alto nivel

Avanza
Gira
Avanza ***3 veces***
Gira
Avanza

Programa máquina

MI	***A***	***R***	***MD***	***A***	***R***
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*0*	*0*	*0*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*0*	*0*	*0*
1	*1*	*0*	*1*	*1*	*0*

Universidad Zaragoza
1542

Niveles de lenguaje de programación

Programa alto nivel

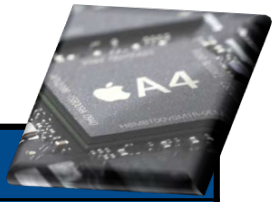
Avanza
Gira
Avanza ***3 veces***
Gira
Avanza

Programa máquina

MI	*A*	*R*	*MD*	*A*	*R*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*0*	*0*	*0*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*0*	*0*	*0*
1	*1*	*0*	*1*	*1*	*0*

Programa máquina

MI	*A*	*R*	*MD*	*A*	*R*
1	*1*	*0*	*1*	*1*	*0*
0	*0*	*0*	*1*	*1*	*0*
0	*0*	*0*	*1*	*1*	*0*
0	*0*	*0*	*1*	*1*	*0*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*1*	*1*	*0*
0	*0*	*0*	*1*	*1*	*0*
0	*0*	*0*	*1*	*1*	*0*
0	*0*	*0*	*1*	*1*	*0*
1	*1*	*0*	*1*	*1*	*0*



MI	A	R	MD	A	R	
1	1	0	1	1	0	Avanza 1 m
1	0	1	1	0	1	Retrocede 1 m
1	1	0	0	0	0	Gira Derecha 90 º
0	0	0	1	1	0	Gira Izquierda

MI	A	R	MD	A	R	
1	1	0	1	1	0	Avanza 1 m
1	0	1	1	0	1	Retrocede 1 m
0	0	0	1	1	0	Gira Izquierda

Universidad Zaragoza
1542

Niveles de lenguajes de programación

- Alto Nivel
  - Mas fáciles de programar
  - Independencia de la maquina
  - Necesidad de traductores compiladores
  - Menor coste de desarrollo
  - Mas fácilmente mantenibles
  - Detección de errores
  - Transportables
  - Son un compromiso entre eficiencia y manejabilidad
- Bajo Nivel
  - Programas específicos para un tipo de máquina
  - Pueden permitir hacer cosas de forma más eficiente.

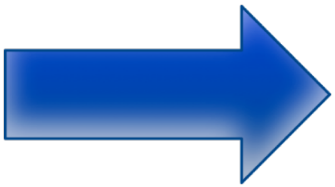
Universidad Zaragoza
1542

Niveles de lenguajes de programación

Código fuente

Programa alto nivel

Avanza
Gira
Avanza ***3 veces***
Gira
Avanza



Compilación

Código objeto (ejecutable)

Programa máquina

MI	*A*	*R*	*MD*	*A*	*R*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*0*	*0*	*0*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*1*	*1*	*0*
1	*1*	*0*	*0*	*0*	*0*
1	*1*	*0*	*1*	*1*	*0*

Universidad Zaragoza
1542

Lenguajes Interpretados/ Compilados

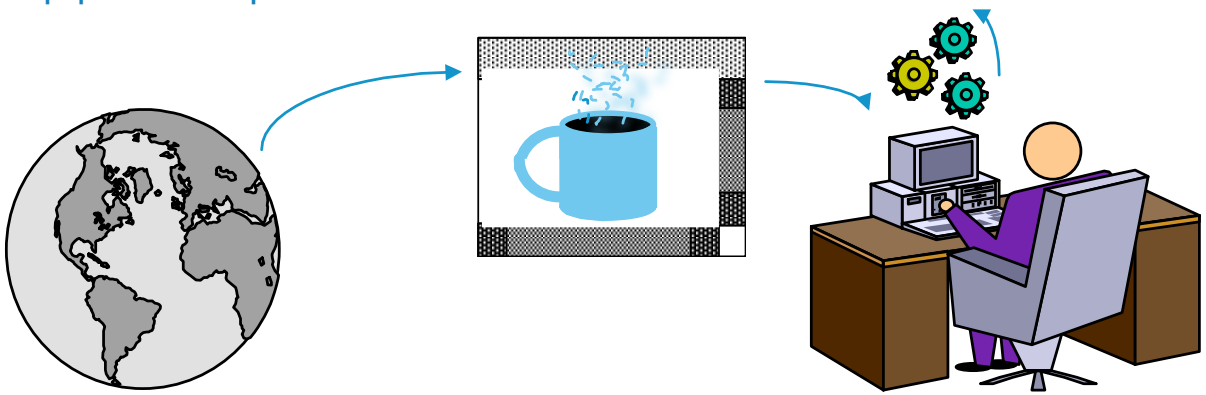
- La ejecución **interpretada** sigue estos pasos
  - Obtención de la siguiente instrucción a ejecutar del código fuente
  - Análisis de la instrucción y determinación de las acciones a ejecutar
  - Ejecución de las correspondientes acciones

- Ejecución del programa **compilado**
  - Se realiza previamente la **compilación** del programa fuente en la versión equivalente del programa en lenguaje máquina. Esta versión se conoce como programa objeto.
  - El **programa objeto** se une, eventualmente, con los subprogramas de biblioteca descritos en el programa fuente para obtener el **programa ejecutable**. Esta operación se denomina edición de uniones (link en inglés).

Universidad Zaragoza
1542

¡¡¡Java, código objeto que se ejecuta en cualquier máquina!!!

- Un **applet** es un programa escrito en Java que anima una porción de la página Web
  - El usuario puede interaccionar con un applet, gracias a que se trata de un programa.
  - Un applet se ejecuta completamente en el cliente:
    - Una vez transmitido la velocidad de la interacción no depende de la red
  - Si es necesario el applet se puede comunicar con el servidor



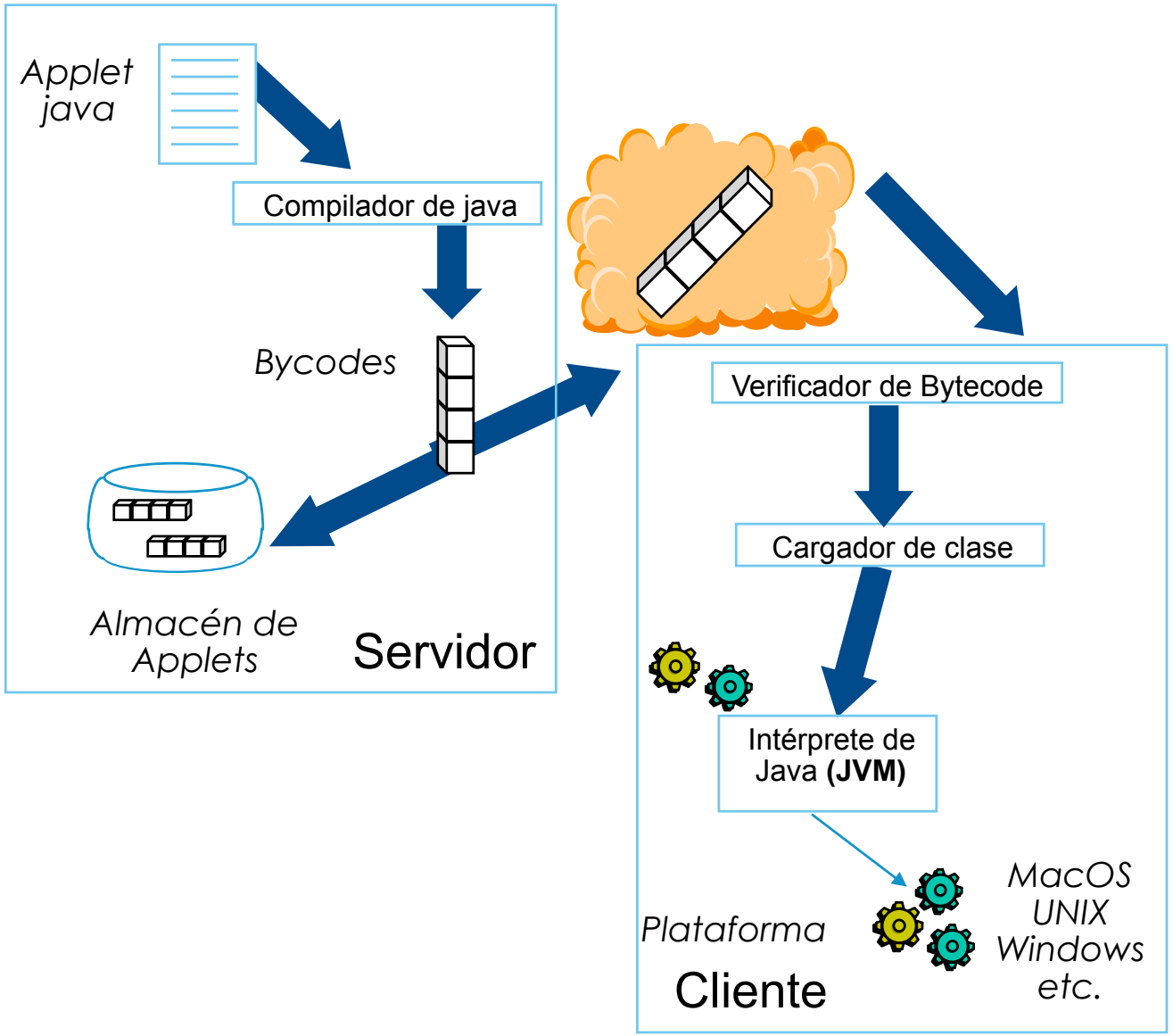
Universidad Zaragoza
1542

¡¡¡Java, código objeto que se ejecuta en cualquier máquina!!!

- Una aplicación java debe poderse ejecutar en una amplia gama de plataformas con diferentes SO y procesadores.
  - Las aplicaciones Java se almacenan en un código intermedio independiente de la plataforma (el byte-code)
  - El procesador no es físico: Es una máquina virtual. Un programa que simula un procesador (idéntico en todas las máquinas).

Universidad Zaragoza

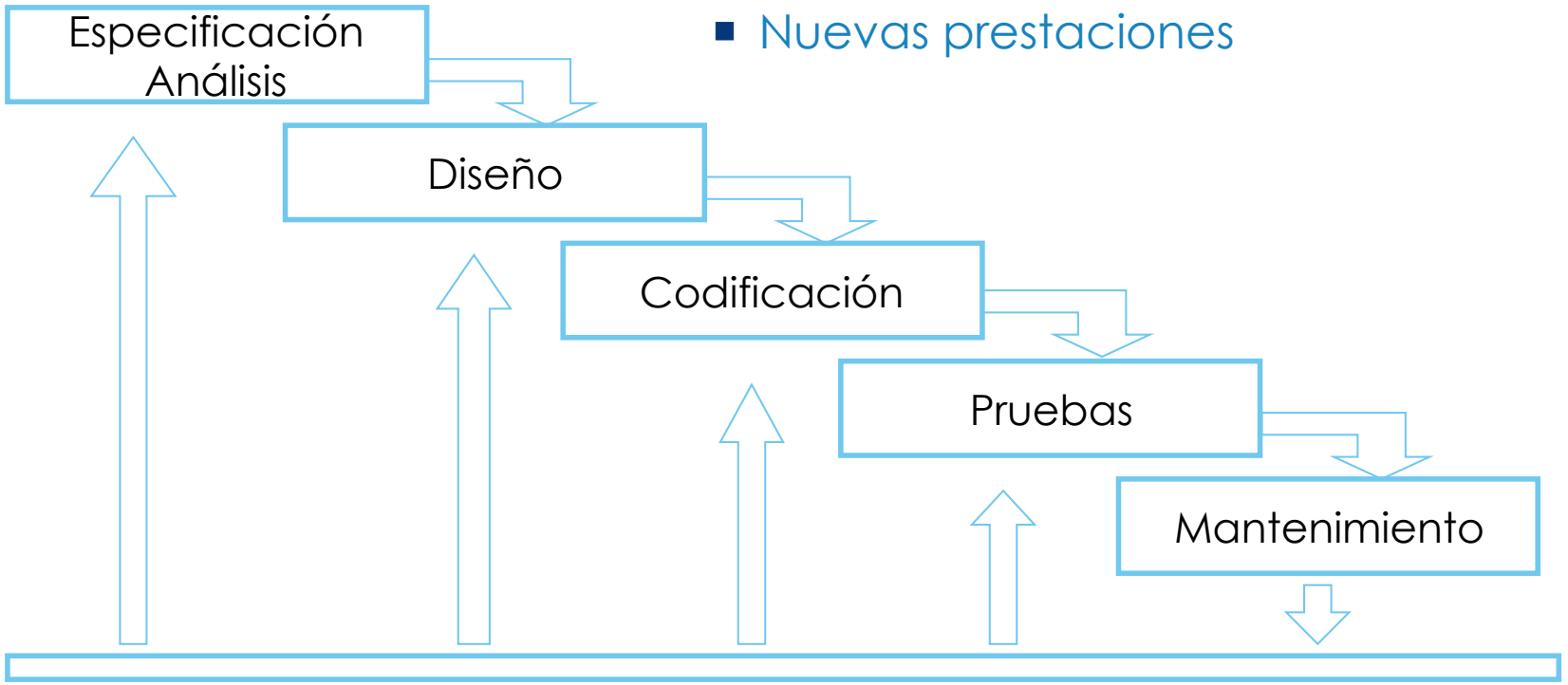
JVM



Universidad Zaragoza
1542

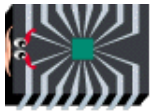
Ciclo de vida de un programa

- Un programa sufrirá cambios:
  - Errores detectados
  - Aparición de nuevos requisitos:
    - Externos: periféricos, sistema operativo
    - Nuevas prestaciones



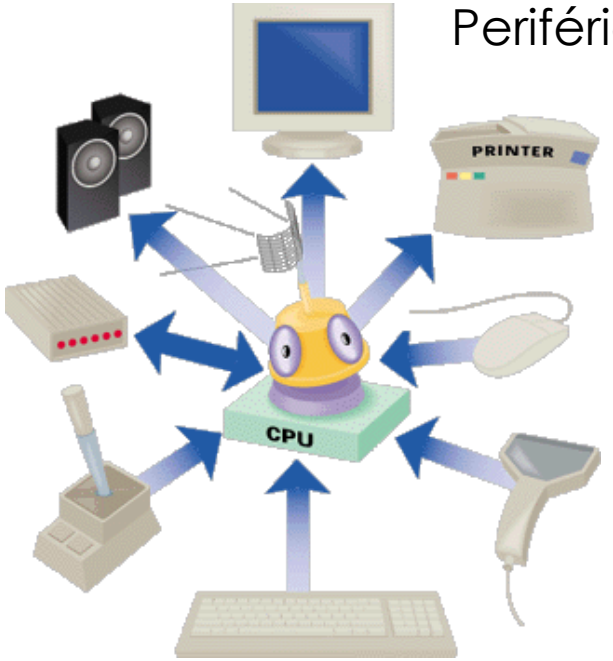
Universidad Zaragoza
1542

El computador (Hardware)



CPU (Central Process Unit)

Periféricos



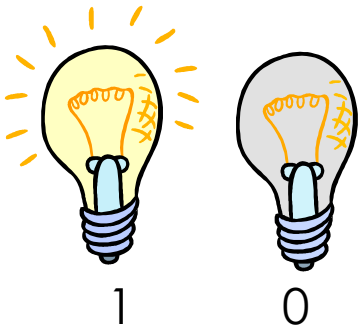
Memoria

Universidad Zaragoza
1542

Memoria de un computador

Memoria

- Se realiza mediante de transistores que se utilizan a modo de "interruptores" que pueden estar encendidos o apagados



*Cada transistor contiene un **bit** de información.*
Teoría del lenguaje: La unidad mínima de información es el bit.

Universidad Zaragoza
1542

Memoria de un computador

- 8 bits = 1 byte Podemos representar 256 valores del 0 al 255 en **binario**

Base 10	**Base 2**	
2562 = 2 * 10^3	**1101= 1 * 2^3=**	**8**
5 * 10^2	**1 * 2^2=**	**4**
6 * 10^1	**0 * 2^1=**	**0**
2 * 10^0	**1 * 2^0=**	**1**
		13

Universidad Zaragoza
1542

Memoria

- **Podemos representar en memoria información numérica y no numérica:**

1101= 13 (Número 13)

Utilizar una **codificación** para los caracteres (ASCII)

El 65 representa la a

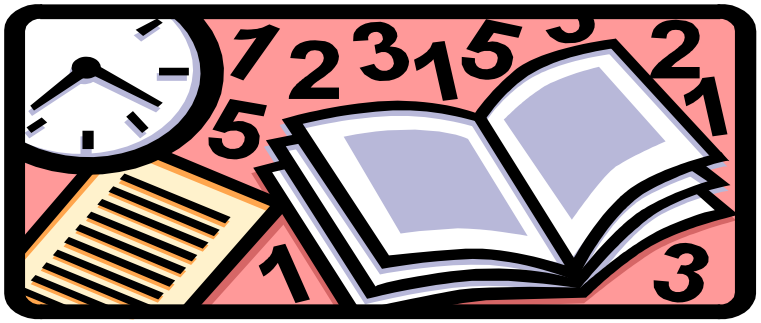
El 66 representa la b

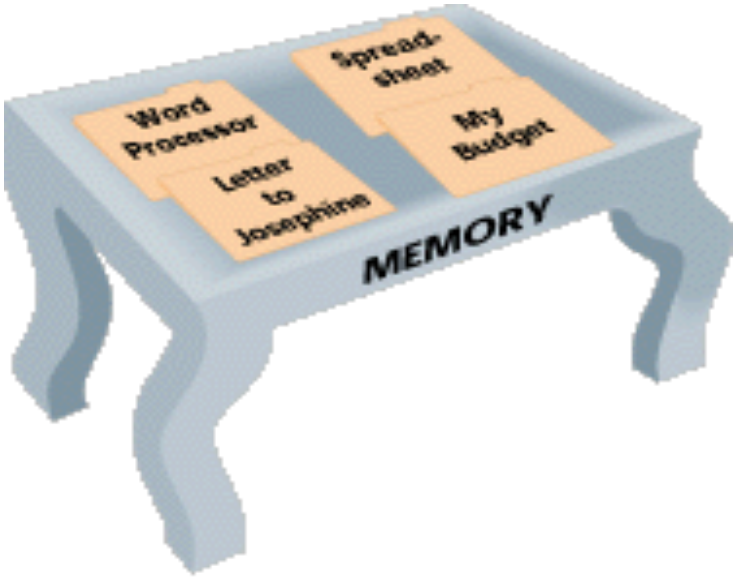
....

Universidad Zaragoza
1542

Memoria

- Agrupando caracteres representamos palabras, documentos, programas ...



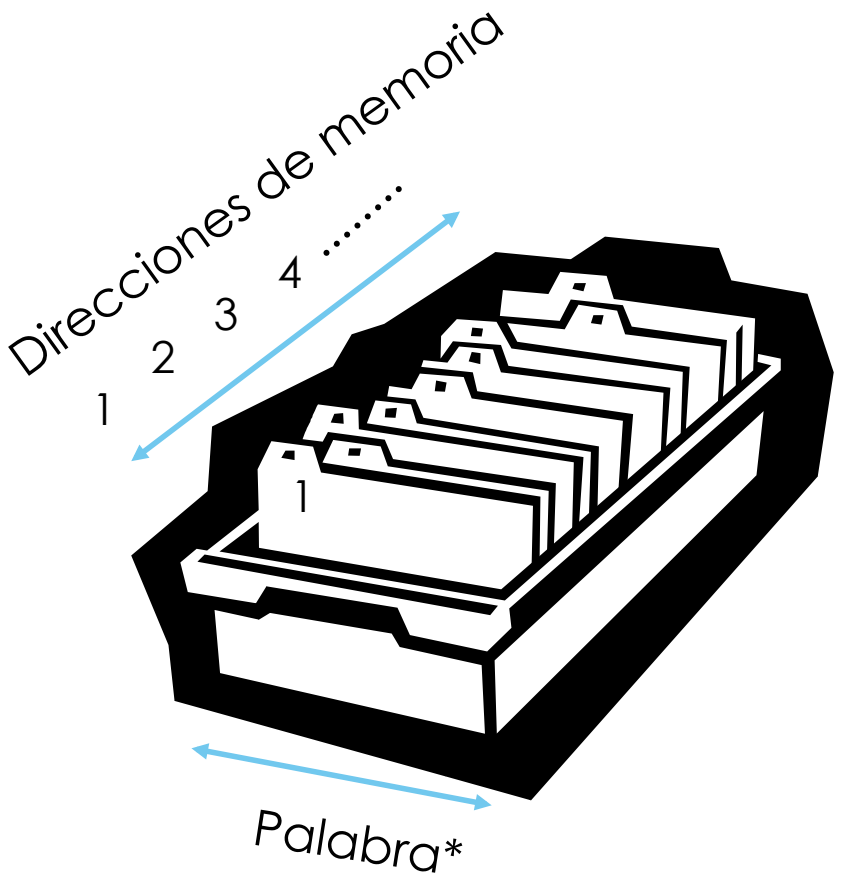


Universidad Zaragoza

Dirección de memoria

Como almacenar y encontrar la información

- En direcciones de memoria

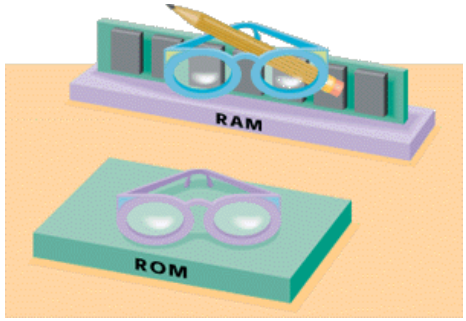


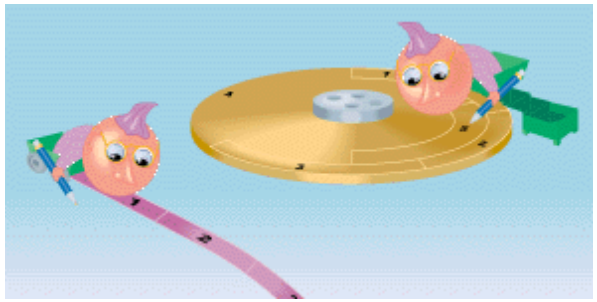
*__Palabra__ del computador = Número de bytes del mínimo bloque direccionable

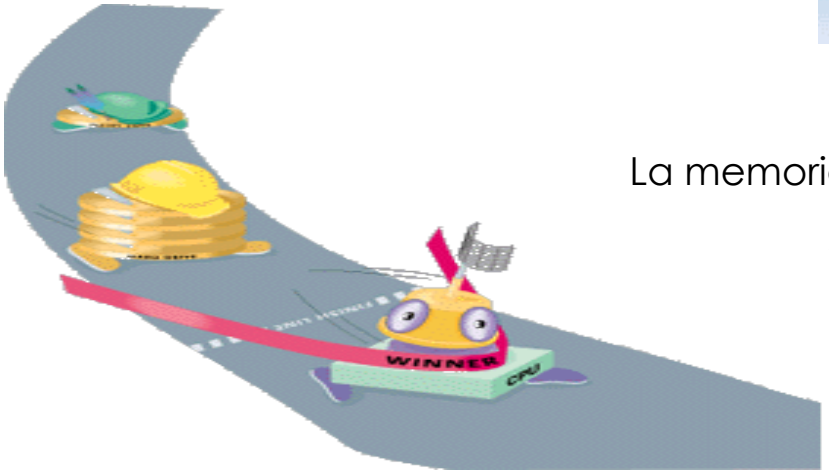
Universidad Zaragoza
1542

Memoria Principal/Memoria Secundaria

- Tipos de memoria
  - Memoria Principal
    - RAM (Read Access Memory)
    - ROM (Read Only Memory)
  - Memoria Secundaria
    - Acceso secuencial
    - Acceso directo







La memoria secundaria solo se debe utilizar para almacenar datos:

Vel. mem. primaria >>>> Vel. mem.secundaria

Universidad Zaragoza
1542

Medida de memoria

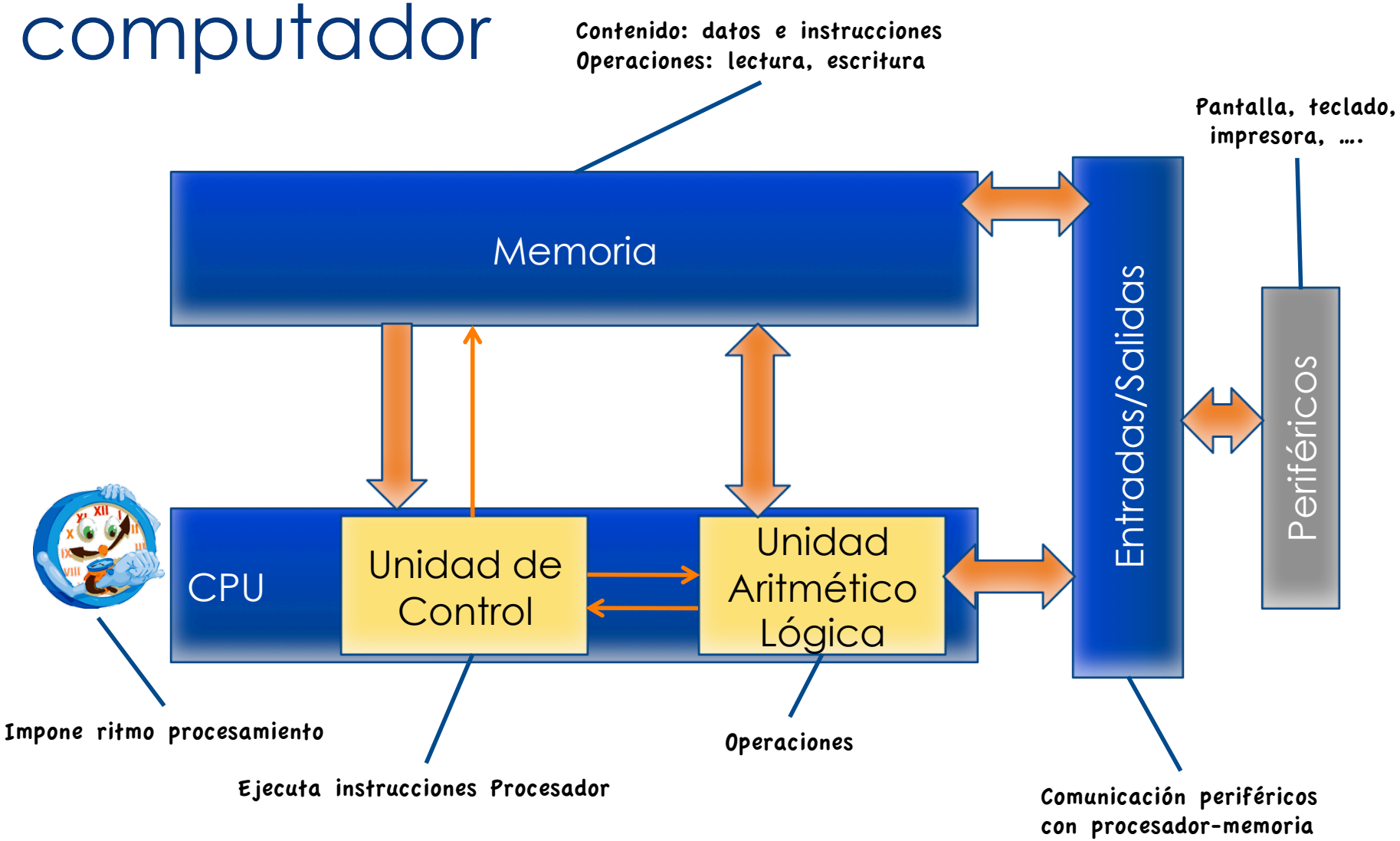
- 1 Kilobyte (KB o K) = 1,024 bytes (1024 es 2^10).
  - Un disco con 360KB de datos, puede representar unos 360.000 caracteres.
- 1 Megabyte (MB or M) es un kilobyte al cuadrado (1,024 veces 1,024, aproximadamente un millón de bytes).
  - Un disco con una capacidad de 1.44MB puede contener 1.440.000 caracteres.
- 1 Gigabyte y Terabyte El termino gigabyte (GB or G) significa un kilobyte al cubo (1,024 * 1,024 * 1,024), o aproximadamente mil millones de bytes. El termino terabyte (TB o T) significa un kilobyte a la cuarta (1,024^4), más de un billón de bytes.
- ¿Cuanta memoria se precisa?:
  - Hace unos pocos años los computadores se vendían de serie con 1MB de RAM. Hoy se ofrecen con 500 MB, 1 GB... o más. Los discos duros están en cientos de gigabytes.
  - Una página de texto típico contiene entre 2,500 y 3,000 caracteres. 1MB equivale aproximadamente a 400 paginas de texto.

Zx81 (1981)
1 Kb RAM

Mac Mini(2010)
2 Gb RAM

Universidad Zaragoza
1542

Funcionamiento del computador



Universidad Zaragoza
1542

Propiedades de los algoritmos

- Condiciones Necesarias
  - Finitud
  - No ambigüedad
- Propiedades deseables
  - Generalidad
  - Eficiencia
  - Independencia de la máquina

Universidad Zaragoza
1542

Sistema Operativo

- Sistema Operativo
  - Conjunto de programas que tienen por misión facilitar la utilización del computador
    - Acceso a los usuarios autorizados
    - Edición de ficheros de texto
    - Puesta a punto y ejecución de programas
    - Seguridad y protección
    - Facturación y contabilidad de uso de recursos
    - Etc.
  - Gestión óptima de la máquina
    - Gestión de memoria
    - Control de dispositivos periféricos
    - Acceso a ficheros
    - Asignación de recursos y ordenación de tareas
    - Etc.

Windows

Mac OX

Unix

Lynux

Universidad Zaragoza
1542

Entorno de programación

- Entorno de trabajo:
  - Programa cuya misión es facilitar el desarrollo de programas utilizando un determinado lenguaje
  - Integra herramientas:
    - Edición e programas fuente (C, C++, Java, Pascal, Ada, ...)
    - Compilación de programas fuente
    - Depuración de programas
    - Ejecución y prueba de programas
    - Construcción de programas ejecutables y bibliotecas

Universidad Zaragoza
1542