

LENGUAJES DE MARCAS

UNIDAD 03

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

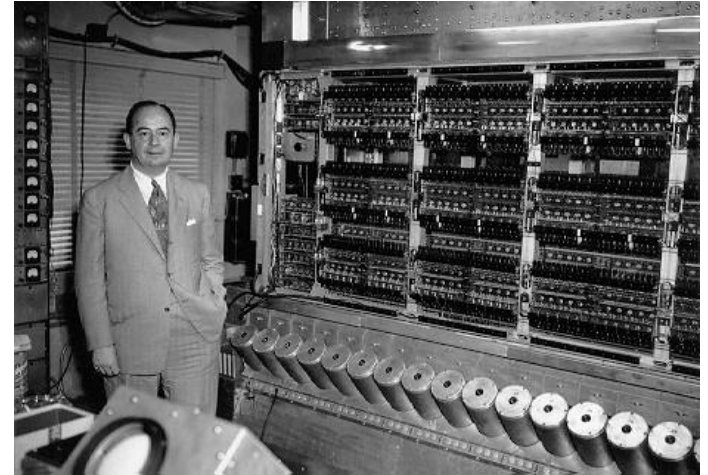
Contenidos

- La arquitectura von Neumann.
- Ejecución de un programa.
- Lenguajes de Programación.
- El proceso de traducción: compiladores e intérpretes.
- Algoritmos y programas.
- Elementos de los lenguajes de programación.
 - Tipado de datos
 - Constantes y variables
 - Operadores
 - Instrucciones de declaración y control de flujo
 - Bibliotecas

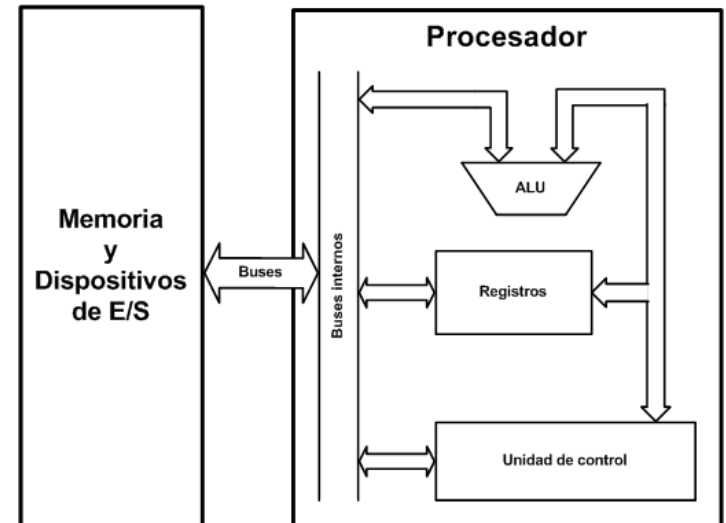
CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

1. La arquitectura von Neumann

- En 1947, el científico húngaro John von Neumann se interesa por el problema de “recablear” la computadora ENIAC cada vez que era necesario modificar su programación.
- Fruto de esta investigación surge una arquitectura que permanece vigente desde entonces y es la utilizada en la mayoría de computadoras actuales.
- La aportación revolucionaria de von Neumann fue precisamente hacer ver la necesidad de **separar el programa de la máquina** misma, surgiendo así el concepto de **programa almacenado**.
- A diferencia de los computadores predecesores, von Neumann propuso que tanto el programa como sus datos fueran almacenados en la memoria del computador. Esto simplificaba la labor de programación al no tener que llevar a cabo el recableado del computador y además permitía diseñar software independiente del hardware.



John von Neumann junto a la computadora EDVAC, la primera computadora con arquitectura von Neumann.

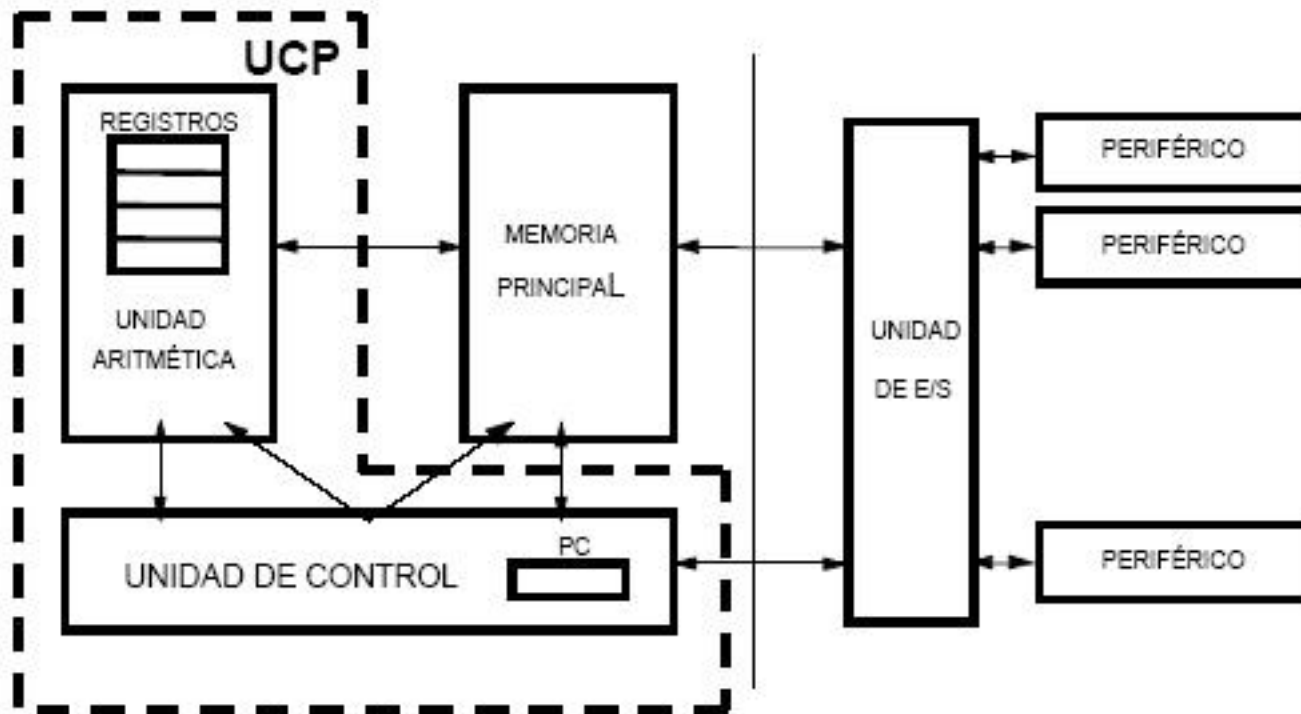


Esquema simplificado de la arquitectura de von Neumann.

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

1. La arquitectura von Neumann

- Esquema general de la arquitectura de von Neumann:

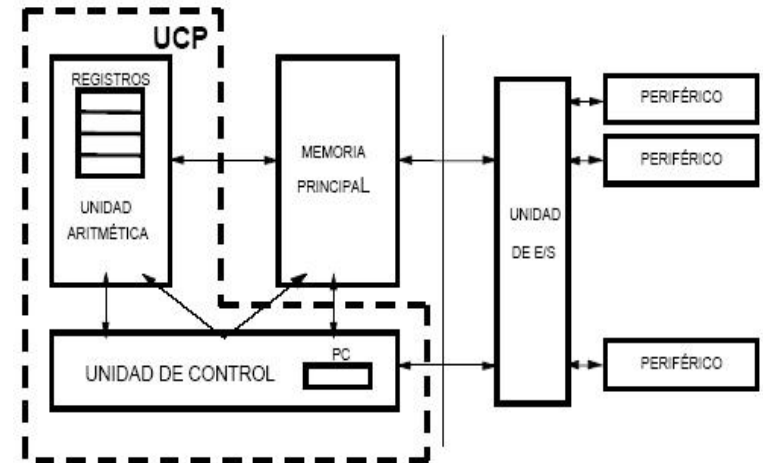


CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

1. La arquitectura von Neumann

Componentes de la arquitectura:

- La **Unidad Central de Proceso (CPU)**, que se encarga de ejecutar las instrucciones, contando para ello con:
 - **Registros:** son pequeñas memorias donde se guardan temporalmente los datos con los que se va a trabajar.
 - El **Contador de Programa** es un registro especial que contiene la dirección de la memoria principal en la que se encuentra la siguiente instrucción a ejecutar.
 - **Unidad Aritmético-Lógica (ALU)**, que se encarga de realizar operaciones aritméticas (suma, resta...), operaciones lógicas (AND, OR, NOT...) y operaciones de desplazamiento.
 - **Unidad de control**, encargada de ejecutar las instrucciones que se leen de la memoria principal.
- **Memoria principal**, Es un conjunto de *celdas*, cada una de las cuales posee una dirección. En cada celda se almacenan las instrucciones y los datos que han de ser procesados por la CPU.
- **Unidad de Entrada/Salida**, que se encarga de gestionar las comunicaciones con los periféricos.
- **Bus de datos**, son las líneas de comunicación que ponen en contacto los anteriores componentes y que transportan instrucciones y datos.

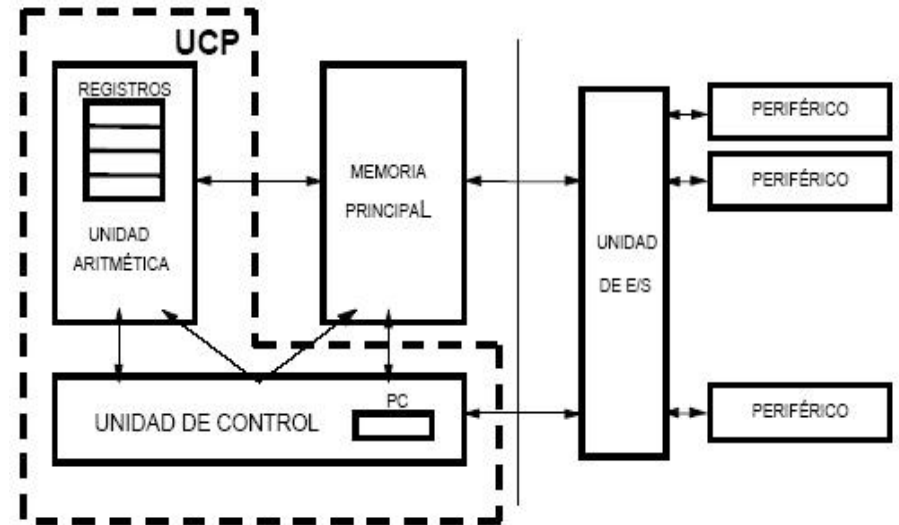


CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

2. Ejecución de un programa

- Un ordenador con arquitectura von Neumann ejecuta un programa mediante los siguientes pasos:

- 1) Las instrucciones del programa son llevadas a la memoria principal (RAM) y el contador de programa almacena la posición de la primera instrucción.
- 2) La CPU obtiene la instrucción apuntada por el contador de programa y la almacena en un registro.
- 3) La Unidad de Control de la CPU reconoce la instrucción y da las órdenes adecuadas al resto de componentes del ordenador para realizar las operaciones correspondientes a esa operación.



- 4) Una vez que se ha ejecutado la instrucción pueden ocurrir dos situaciones:
 - a) El contador de programa apunta a la siguiente dirección y comienza de nuevo el proceso en el punto 1)
 - b) La instrucción ejecutada puede cambiar el valor del contador de programa para que “salte” a otra instrucción, permitiendo de ese modo realizar operaciones repetitivas y/o condicionales.

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

3. Lenguajes de programación

- A diferencia de los seres humanos, el lenguaje que entiende la CPU se compone exclusivamente de unos y ceros (lenguaje binario).
- Por ello, para comunicarnos con el procesador y pedirle que realice una operación, es necesario escribir una orden o **instrucción** que es una secuencia de unos y ceros.



- Por ejemplo, si queremos sumar $3+2$ deberíamos escribir una instrucción parecida a la siguiente:

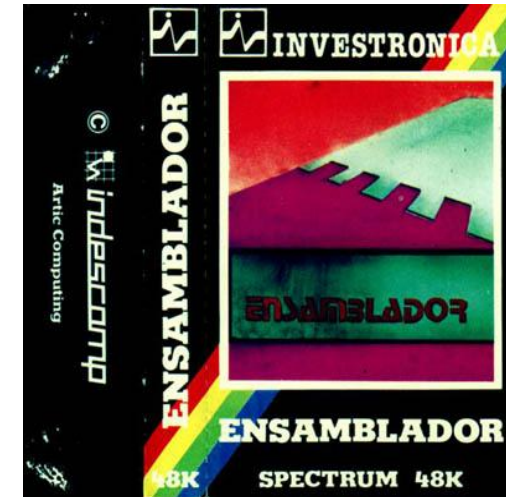
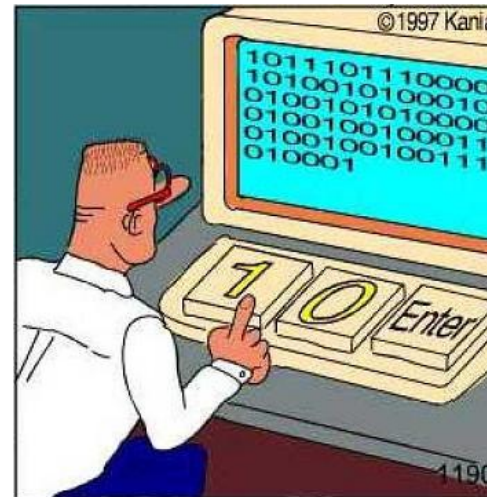
Código de la operación (suma)	Primer sumando en binario	Segundo sumando en binario	Registro en el que se almacenará el resultado
00001	00011	00010	00001

- El conjunto de instrucciones que una CPU es capaz de reconocer se llama **repertorio de instrucciones** del procesador.
- Las instrucciones (también llamadas **palabras**) que manejan los ordenadores actuales son secuencias de 32 o 64 unos y ceros. Por eso hablamos de procesadores de 32 y 64 bits respectivamente.

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

3. Lenguajes de programación

- El lenguaje binario resulta difícil de leer y escribir para los seres humanos. Por eso, los lenguajes de programación han ido evolucionando hacia lenguajes de **alto nivel**, es decir, lenguajes parecidos al humano.
- El lenguaje ensamblador constituye un primer nivel de lenguajes que utiliza palabras cortas y abreviaturas en vez de instrucciones binarias.



```
; Primer programa de prueba. Inicializa el Puerto B y pone todos los
; pines del puerto en un estado lógico "1"
; Fecha: 17.01.07   Autor: Jorge A. Bojórquez   micropic.wordpress.com

list      p=l6f628a      ; Declaración del procesador
include   p16f628a.inc   ;
__config  0x3F38         ; Declaración de la configuración

                                ; Inicio del programa
org       0x00             ; Vector de Inicio
goto      Inicio           ; Ir a la etiqueta 'Inicio'

Inicio    bsf      STATUS,RPO ; Seleccionar el banco de memoria 1
          clrf     PORTB      ; Configurar puerto B como salida
          bcf      STATUS,RPO ; Seleccionar el banco de memoria 0

          movlw    0xFF        ; Cargar al acumulador W el valor 0xFF
          movwf    PORTB       ; Pone todos los pines del Puerto B en "1"

Ciclo     goto      Ciclo

end
```

Ejemplo de un programa en lenguaje ensamblador. En la mayoría de lenguajes, las instrucciones son palabras y abreviaturas en inglés

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

3. Lenguajes de programación

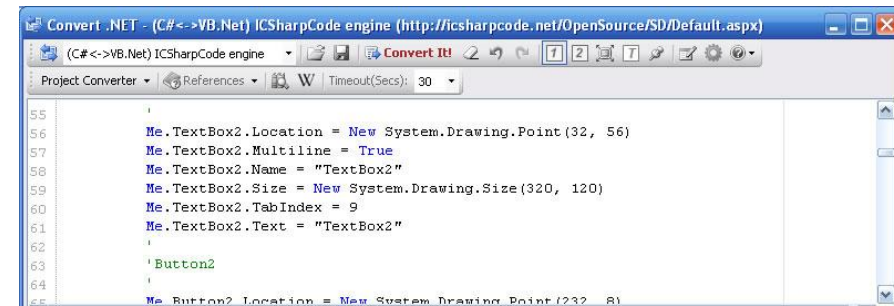
- Aunque el lenguaje ensamblador utiliza palabras y abreviaturas que resultan más cómodas que el manejo de cadenas de unos y ceros, todavía utiliza conceptos como **registros** y **direcciones de memoria** que quedan alejados del lenguaje natural.
- En la actualidad hablamos de **lenguajes de alto nivel** para referirnos a los lenguajes de programación más evolucionados, que se parecen más al lenguaje natural y al modo de percibir la realidad de los seres humanos.
- Algunos de estos lenguajes como Java o la familia de lenguajes .Net suponen una drástica mejora en cuanto a facilidad de aprendizaje, legibilidad y tiempo de desarrollo.



CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

4. El proceso de traducción: compiladores e intérpretes

- Con independencia del lenguaje de programación que se utilice para desarrollar un programa, es necesario que las instrucciones escritas (llamadas **código fuente**) se traduzcan a una secuencia binaria equivalente, puesto que éste es el único lenguaje que la CPU puede comprender y ejecutar.
- Esta tarea de conversión desde un lenguaje de alto nivel a un lenguaje binario la realizan los **compiladores e intérpretes**.



```
55 Me.TextBox2.Location = New System.Drawing.Point(32, 56)
56 Me.TextBox2.Multiline = True
57 Me.TextBox2.Name = "TextBox2"
58 Me.TextBox2.Size = New System.Drawing.Size(320, 120)
59 Me.TextBox2.TabIndex = 9
60 Me.TextBox2.Text = "TextBox2"
61
62
63 'Button2
64
65 Me.Button2.Location = New System.Drawing.Point(232, 8)
```



traducción

00011	01011	01011	00010
00001	00011	00010	00001
...			



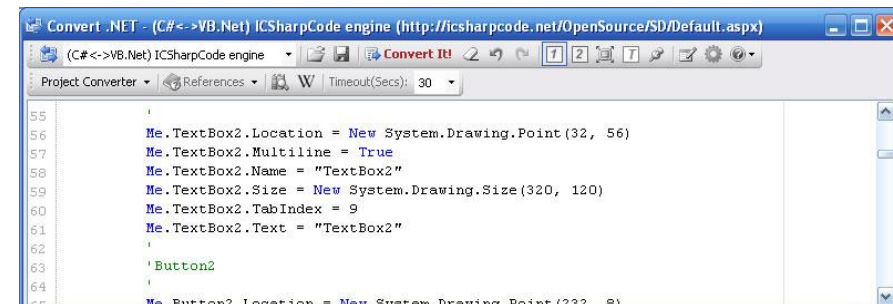
ejecución



CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

4. El proceso de traducción: compiladores e intérpretes

- Un **compilador** realiza las siguientes operaciones:
 - Analiza** el código fuente en busca de errores.
 - Genera una versión **optimizada** y en **formato binario** del código fuente.
 - Guarda esta versión en un fichero llamado **código objeto** o **fichero ejecutable**.
- En los sistemas Windows, un fichero ejecutable tiene extensión **“.exe”**
- En los sistemas Linux, los ficheros ejecutable se pueden identificar por tener asociado un permiso de ejecución.



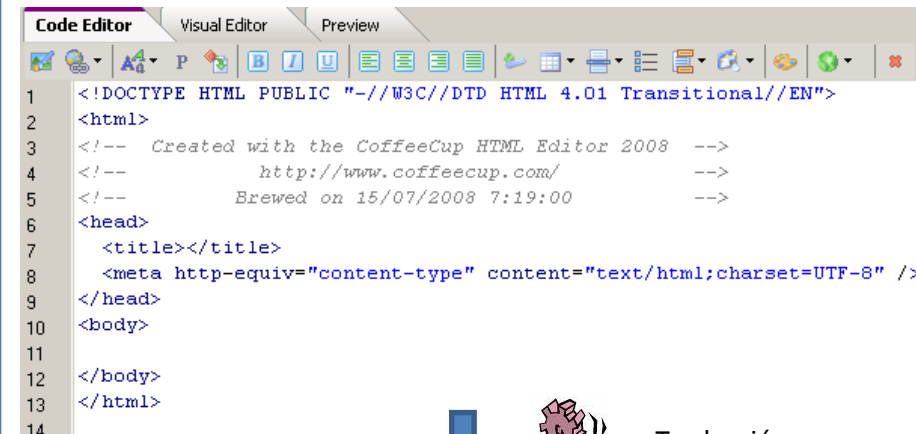
```
55  
56 Me.TextBox2.Location = New System.Drawing.Point(32, 56)  
57 Me.TextBox2.Multiline = True  
58 Me.TextBox2.Name = "TextBox2"  
59 Me.TextBox2.Size = New System.Drawing.Size(320, 120)  
60 Me.TextBox2.TabIndex = 9  
61 Me.TextBox2.Text = "TextBox2"  
62  
63 'Button2  
64  
65 Me.Button2.Location = New System.Drawing.Point(232, 8)
```



CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

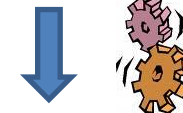
4. El proceso de traducción: compiladores e intérpretes

- Un **intérprete** toma una instrucción del código fuente, la traduce al formato binario y la ejecuta.
- Cuando se utiliza un intérprete **no** se produce un análisis del código fuente para detectar errores y **tampoco** se genera un código optimizado. En caso de existir un error en el código fuente, la ejecución del programa se detiene.
- Como un intérprete traduce y ejecuta de una vez, no se genera un archivo ejecutable.
- Un ejemplo de código interpretado son las páginas web, que están escritas en HTML, un lenguaje interpretado (siendo el navegador la aplicación que hace de intérprete).



```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2 <html>
3 <!-- Created with the CoffeeCup HTML Editor 2008 -->
4 <!-- http://www.coffeecup.com/ -->
5 <!-- Brewed on 15/07/2008 7:19:00 -->
6 <head>
7   <title></title>
8   <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
9 </head>
10 <body>
11
12 </body>
13 </html>
14
```

Intérprete



Traducción y
ejecución directa

No hay ejecutable. Se lee una instrucción, se traduce y se ejecuta.
A continuación se lee la siguiente instrucción y se repite el proceso.

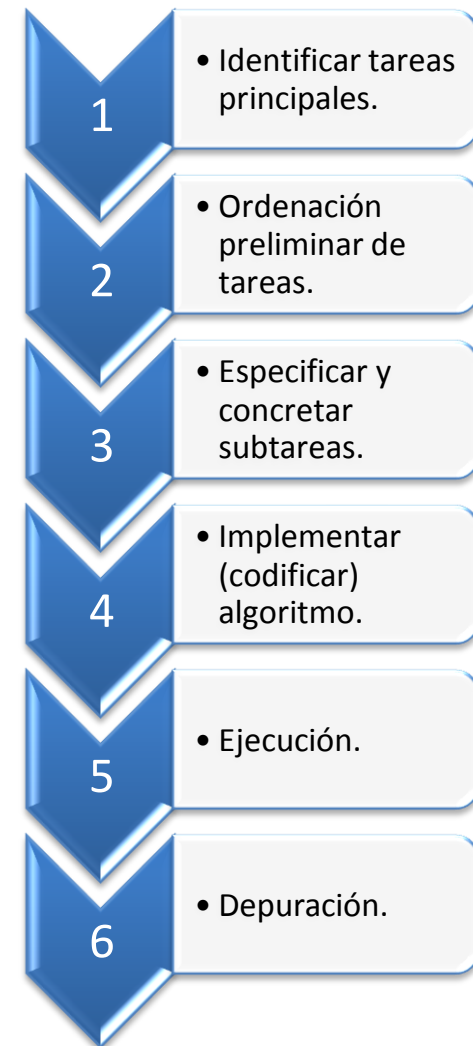
00011	01011	01011	00010
00001	00011	00010	00001
...			



CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

5. Algoritmos y programas

- **Programa:** secuencia de instrucciones que son ejecutadas por la CPU para realizar una tarea determinada.
- **Algoritmo:** descripción ordenada de los pasos necesarios para realizar una tarea. Podemos pensar en un algoritmo como en una especie de receta de cocina donde detallamos qué vamos a hacer, en qué orden y cómo. Generalmente un algoritmo se representa de manera gráfica y/o utilizando una notación especial denominada **pseudocódigo**.
- Antes de escribir un programa necesitamos establecer el algoritmo que ejecutará el programa. Esto permite especificar claramente cuál será la tarea a realizar y el modo de realizarla.
- Una vez que se ha diseñado el algoritmo, es posible traducirlo a un lenguaje de programación (instrucciones) para crear un programa.
- Para diseñar un algoritmo se debe comenzar por **identificar las tareas más importantes** y disponerlas en el **orden apropiado**.
- Sobre este algoritmo básico se van realizando sucesivas operaciones de refinamiento, que suelen consistir en delimitar subtareas y concretar los detalles de cada una de ellas creando así **módulos** (partes de código fuente) más o menos independientes.
- Este enfoque para la elaboración de algoritmos modulares se conoce como **modularización** y la técnica utilizada para refinar el módulo, de lo general a lo específico, se conoce como diseño descendente o **TOP-DOWN**.



CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- Una vez diseñado y depurado el algoritmo, se pasa a su codificación en un lenguaje de programación.
- La codificación es una traducción del algoritmo a un lenguaje de programación concreto. Generalmente el algoritmo es abstracto pero al codificarlo es necesario precisar todos los detalles utilizando el léxico y la sintaxis concreta del lenguaje de programación utilizado.
- Para codificar un algoritmo, los lenguajes de programación cuentan con los siguientes elementos:
 - **Tipado de datos.**
 - **Constantes y variables.**
 - **Operadores.**
 - **Instrucciones de declaración y control de flujo.**
 - **Bibliotecas funciones y/o de clases.**

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- **Tipado de datos:** la mayoría de lenguajes de programación necesitan que el programador especifique el **tipo de dato** de la información con la que opera el programa.
- Una **declaración** es una instrucción en la que se especifica el tipo de dato que se va a aplicar a un dato que se usa en el programa.
- El tipo de dato le indica al computador la naturaleza de la información (números, texto u otras estructuras) y también el rango posible de valores que puede tomar ese dato.

× Tipos de datos básicos en Java:

Tipo de dato	Descripción	Rango de valores
boolean	Valor lógico “verdadero” o “falso”	“true” o “false”
char	Un carácter	Cualquier carácter Unicode
byte	Un número entero de 8 bits	[-128, +128]
short	Un número entero pequeño	[-32768, +32768]
int	Un número entero	[-2,147,483,648, +2,147,483,647]
long	Un número entero grande	$[-2^{63}, +2^{63}]$
float	Un número con decimales	$[-3.4 \cdot 10^{38}, +3.4 \cdot 10^{38}]$
double	Un número con decimales de doble precisión	$[-1.8 \cdot 10^{308}, +1.8 \cdot 10^{308}]$

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- **Constantes:** son almacenes en la memoria principal en los que se guarda información que no cambia (es constante) durante la ejecución del programa.
- **Variables:** son almacenes en la memoria principal donde se puede depositar información de un tipo determinado (textual, numérica, etc.) durante la ejecución del programa. A diferencia de las constantes, la información guardada puede ser sobrescrita o borrada.
- Una constante o variable debe ser declarada antes de ser utilizada. En la **declaración** se especifica el tipo de dato y un nombre para esa constante o variable.
- El valor de las constantes debe establecerse en el momento de su declaración. El valor de las variables puede establecerse en cualquier momento.

```
// la declaración de constantes en Java
// tiene la siguiente sintaxis:
// final <tipo_de_dato> <nombre> = <valor>;

// La siguiente sentencia declara la
// constante PI como un número con decimales
// e inicializa su valor a 3.14

final double PI = 3.14;

// la declaración de variable tiene
// la siguiente sintaxis:
// <tipo_de_dato> <nombre> [= <valor>]

// la siguiente sentencia declara
// una variable de tipo entero llamada
// número.

int número;

// la siguiente sentencia declara e inicializa
// una variable de tipo carácter llamada letra

char letra = 'h';
```


CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- **Operadores.** Son elementos que actúan sobre valores, constantes o variables y realizan sobre éstos una operación de alguno de los siguientes tipos:
 - **Operadores aritméticos** : Suma (+), Resta(-), Producto(*), Potencia (^), División (/), Módulo (\).
 - **Operador de asignación (=)**: Asigna el valor de la expresión que está a su derecha al elemento que tiene a su izquierda
 - **Operadores relacionales**: Igual que (==), Menor que (<), Mayor que (>), Distinto de (!=), Mayor o igual que (>=), Menor o igual que (<=).
 - **Operadores lógicos**: Y (And), O (Or), No(Not).
 - **Operadores alfanuméricos**: Concatenar (+).
- La sintaxis de los operadores difiere entre distintos lenguajes. Por ejemplo, en lenguaje Java el operador concatenación es "+", mientras que en Visual Basic .Net es "&".

```
// declara e inicializa la variable a
int a = 7;

// declara e inicializa la variable b
int b = 3;

// declara la variable c.
int c;

// guarda en la variable c la suma del
// contenido de las variable a y b.
// Ahora c = 10
c = a + b;

// expresión lógica que evalúa si c es
// igual a 10
if (c == 10)
{
    // bloque de instrucciones
    // a ejecutar si c = 10
}

// continua la ejecución del programa...
```

```
// declara e inicializa 2 variables de tipo String
// (cadenas de caracteres)
String palabra1 = "hola";
String palabra2 = "mundo";

// declara e inicializa una variable de tipo String
// cuyo valor es la concatenación del valor de las
// variables palabra1 y palabra2.
// saludo ahora contiene "hola mundo"
String saludo = palabra1 + palabra2;
```

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- **Instrucciones:** son las órdenes básicas con las que se construye el código fuente de los programas. Pueden ser de los siguientes tipos:
 - **Declaración**
 - **Control de flujo:**
 - Selección (if, if-else , if-elseif, switch).
 - Iteración (for, while-do, do-until).
- Cada instrucción o grupo de instrucciones que se incluyen en el código fuente y que realiza una operación atómica se denomina **sentencia**. En algunos lenguajes de programación (como Java, C, C++ y C# .Net) es necesario terminar las sentencias con el carácter “;”.
- La mayoría de lenguajes incluyen además **instrucciones de salto incondicional (goto, continue, exit...)**. Estas instrucciones rompen la organización de un programa estructurado por lo que **su uso está desaconsejado**. No obstante, existen por motivos de compatibilidad hacia atrás.



CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- **Instrucciones de declaración:** permiten indicar al compilador que va a utilizarse una variable, una constante o una función.
- El valor de las constantes no cambia durante la ejecución del programa por lo que en la declaración debe especificarse el valor que tendrá la constante.
- Es necesario realizar una **declaración** antes de poder utilizar una variable, constante o función. En la declaración se especifica un nombre y un tipo de dato para el elemento que se declara.
- El valor de las variables sí puede cambiar durante la ejecución del programa por lo que las variables pueden ser declaradas o bien pueden ser declaradas e inicializadas en la misma instrucción.
- En algunos lenguajes de programación, como Java, no existe una instrucción especial para hacer una declaración, basta con indicar el tipo y el nombre del dato. Por el contrario, en otros lenguajes se emplean instrucciones de declaración como “Dim” en Visual Basic .Net .
- Para declarar constantes suele especificarse una palabra reservada (como “final” en Java y “Const” en Visual Basic .Net) que indica al compilador que el valor del elemento declarado no puede cambiar.

```
// la declaración de constantes en Java
// tiene la siguiente sintaxis:
// final <tipo_de_dato> <nombre> = <valor>;

// La siguiente sentencia declara la
// constante PI como un número con decimales
// e inicializa su valor a 3.14

final double PI = 3.14;

// la declaración de variable tiene
// la siguiente sintaxis:
// <tipo_de_dato> <nombre> [= <valor>]

// la siguiente sentencia declara
// una variable de tipo entero llamada
// número.

int número;

// la siguiente sentencia declara e inicializa
// una variable de tipo carácter llamada letra

char letra = 'h';
```

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- **Instrucciones de control de flujo selectivas:** permiten evaluar una condición y, en función del resultado de dicha evaluación se ejecuta un determinado bloque de instrucciones

— Instrucción if:

- Si la condición se evalúa como **“true”**, se ejecutan las instrucciones del bloque if.
- Si la condición se evalúa como **“false”**, la ejecución continúa por la instrucción que sucede al bloque if.
- La secuencia de instrucciones del bloque if debe ir **encerrada entre llaves**. No obstante, si sólo hay una instrucción en el bloque, las llaves pueden omitirse.
- En la secuencia de instrucciones del bloque if pueden aparecer otras estructuras de control anidadas.

× Sintaxis en Java:

```
if ( <expresión_lógica> )  
{  
    <bloque_instrucciones>  
}
```

```
int a = 7;  
int b = 3;  
  
if (a > b)  
{  
    System.out.print("a es mayor que b");  
}  
  
System.out.print("fin del programa");
```

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- Instrucciones de control de flujo selectivas:

- Instrucción if – else:

- Si la condición se evalúa como “*true*”, se ejecutan las instrucciones del bloque *<bloque_instrucciones_true>*.
 - Si la condición se evalúa como “*false*”, se ejecutan las instrucciones del bloque *<bloque_instrucciones_false>*.
 - Las secuencias de instrucciones de los bloques if y else deben ir **encerradas entre llaves**. No obstante, si sólo hay una instrucción en alguno de los bloques, las llaves pueden omitirse en dicho bloque.
 - En las secuencias de instrucciones de los bloques if y else puede aparecer otras estructuras de control anidadas.

× Sintaxis en Java:

```
if ( <expresión_lógica> )
{
    <bloque_instrucciones_true>
}

else
{
    <bloque_instrucciones_false>
}
```

```
int a = 7;
int b = 3;

if (a > b)
{
    System.out.print("a es mayor que b");
}
else
{
    System.out.print("b es mayor o igual que a");
}

System.out.print("fin del programa");
```

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- **Instrucciones de control de flujo selectivas:**

- **Instrucción if - elseif:**

evalúa secuencialmente múltiples condiciones hasta encontrar una cuya evaluación devuelva **“true”** o hasta llegar al final de la sentencia.

- Si el resultado de evaluar la primera condición da como resultado **“true”**, se ejecuta `<bloque_instrucciones_1>` y, a continuación se ejecuta la siguiente instrucción a la estructura if – elseif.
- Si el resultado de evaluar la primera condición es **“false”** se pasa a evaluar la condición de la primera instrucción elseif. El proceso se repite hasta alcanzar una evaluación que devuelva **“true”** o hasta alcanzar el final de la estructura if – elseif.
- Pueden añadirse tantos bloques else-if como sean necesarios.
- Opcionalmente puede añadirse un bloque else que se ejecutará si ninguna de las condiciones evaluadas ha devuelto un valor **“true”**

- **Sintaxis en Java:**

```
if ( <expresión_lógica_1> )
{
    <bloque_instrucciones_1>
}
else if ( <expresión_lógica_2> )
{
    <bloque_instrucciones_2>
}
else
{
    <bloque_instrucciones_n>
}
```

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

– Instrucción if – elseif:

```
// Declara dos constantes, con la edad y el sexo
// de un usuario que quiere acceder a la aplicación.
// También declara una variable que indica si el
// usuario puede acceder a la aplicación.
final int EDAD = 25;
final char SEXO = 'V';
boolean permitirAcceso = false;

// Si el usuario es menor de 18 años se muestra un
// mensaje de error y sale de la aplicación
if (EDAD < 18)
{
    System.out.print(";Tienes que se mayor de edad para entrar!");
}

// Si el sexo del usuario no es 'M' (Mujer), se muestra
// un mensaje de error y sale de la aplicación
else if (SEXO != 'M')
{
    System.out.print(";Esta aplicación es sólo para mujeres!");
}

// Si el usuario es mayor de edad y mujer, entra en la
// aplicación.
else
{
    permitirAcceso = true;
}

if (permitirAcceso == true)
    System.out.print("Entrando en la aplicación...");
else
    System.out.print("Fin de la aplicación");
```

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- **Instrucciones de control de flujo selectivas:**

- **Instrucción switch:**

evalúa una expresión y compara el resultado de la evaluación con una lista de constantes. Si encuentra una coincidencia, ejecuta las instrucciones asociadas a esa constante.

- La evaluación de <expresión> debe arrojar un tipo de dato **byte, short, int, char o String**. Los valores de las constantes en las cláusulas **case** debe coincidir con el tipo de dato que devuelve la expresión evaluada.
- Pueden añadirse tantos bloques case como sean necesarios.
- Opcionalmente puede añadirse un bloque **default** que se ejecutará si ninguna de las condiciones evaluadas ha devuelto un valor “true”.
- Si se omite la instrucción **break** en algún bloque case, continuarán ejecutándose las sentencias del siguiente bloque case, hasta encontrar una instrucción break o alcanzar el final de la estructura switch.

× **Sintaxis en Java:**

```
switch ( <expresión> )
{
    case <valor_1>:
        <bloque_instrucciones_1>
        break;
    case <valor_2>:
        <bloque_instrucciones_2>
        break;
    case <valor_n>:
        <bloque_instrucciones_n>
        break;
    default:
        <bloque_instrucciones_d>
        break;
}
```


CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

— Instrucción switch:

```
int día = 8;
String nombreDía;
switch (día)
{
    case 1:
        nombreDía = "Lunes";
        break;
    case 2:
        nombreDía = "Martes";
        break;
    case 3:
        nombreDía = "Miércoles";
        break;
    case 4:
        nombreDía = "Jueves";
        break;
    case 5:
        nombreDía = "Viernes";
        break;
    case 6:
        nombreDía = "Sábado";
        break;
    case 7:
        nombreDía = "Domingo";
        break;
    default:
        nombreDía = "Día incorrecto";
        break;
}
System.out.println(nombreDía);
```

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- **Instrucciones de control de flujo iterativas:**

- **Instrucción for:**

repite un bloque de instrucciones un número determinado de veces.

- **[inicialización]** es una expresión opcional que se ejecuta sólo la primera vez que se alcanza la instrucción for. Habitualmente se utiliza para inicializar una variable que sirve de contador de iteraciones.
- **[condición]** es una expresión lógica (debe devolver *“true”* o *“false”*) opcional. Si la evaluación de esta expresión devuelve *“true”*, el bloque de instrucciones se ejecuta. En caso contrario, finaliza la sentencia. Habitualmente en esta cláusula se comprueba el valor de la variable que sirve de contador. Si se omite esta cláusula, toma por defecto el valor *“true”* por lo que el bloque de instrucciones se repite indefinidamente.
- **[expresión]** es una expresión opcional que se ejecuta después de haber ejecutado el bloque de instrucciones. Habitualmente esta cláusula se utiliza para incrementar la variable que sirve de contador de iteraciones.

- **Sintaxis en Java:**

```
for( [inicialización] ; [condición] ; [expresión] )  
    {  
        <bloque_instrucciones>  
    }
```

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- Instrucciones de control de flujo iterativas:
 - Instrucción for:

```
System.out.println("voy a contar hasta 5");  
final int NUM_ITERACIONES = 5;  
int contador;  
  
for (contador = 1; contador <= NUM_ITERACIONES; contador = contador + 1)  
{  
    System.out.println(contador);  
}
```

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- **Instrucciones de control de flujo iterativas:**

- **Instrucción while:**

Evalúa una expresión lógica y, en caso de devolver “true”, ejecuta un bloque de instrucciones. La ejecución del bloque de instrucciones se repite mientras la evaluación de la expresión devuelva “true”.

- **Instrucción do - while:**

Ejecuta un bloque de instrucciones y después evalúa una expresión lógica. Si el resultado de la evaluación devuelve “true” se repite la ejecución del bloque de instrucciones.

- El comportamiento de cualquiera de estas instrucciones puede reproducirse con una instrucción for.

```
x  Sintaxis en Java:  
while ( <expresión_lógica> )  
    {  
        <bloque_instrucciones>  
    }
```

```
x  Sintaxis en Java:  
do  
    {  
        <bloque_instrucciones>  
    }  
while ( <expresión_lógica> )
```

CONCEPTOS BÁSICOS SOBRE LENGUAJES DE PROGRAMACIÓN

6. Elementos de los lenguajes de programación

- **Bibliotecas de funciones y bibliotecas de clases:**
 - Una **función** es un conjunto de sentencias ordenadas que realizan una tarea determinada. La construcción de funciones evita tener que codificar una y otra vez secuencias de instrucciones que realizan tareas habituales. Por ejemplo, la mayoría de lenguajes de alto nivel tienen funciones que permiten trabajar con ficheros (crear, borrar, copiar,...), acceder a bases de datos, dibujar interfaces gráficas, etc.
 - Aunque aún no hemos abordado el estudio de las clases, los lenguajes de programación orientados a objetos, como Java, poseen bibliotecas de clases cuya aplicación es parecida a las bibliotecas de funciones.
 - Para programar en un lenguaje de programación específico es necesario conocer los elementos del lenguaje que hemos visto y la biblioteca de funciones/clases que ofrece el lenguaje elegido.
 - La biblioteca de funciones/clases de un lenguaje cualquiera suele estar formada por miles de componentes por lo que es imposible conocerla completamente. Por ello, es una tarea crítica desarrollar la capacidad que permita localizar y manejar la documentación de la biblioteca para poder aplicar las funciones que se necesiten en cada situación.
 - Además de la biblioteca de funciones/clases que ofrece un lenguaje, el programador puede crear su propia biblioteca de funciones para ayudarse en el desarrollo de software.