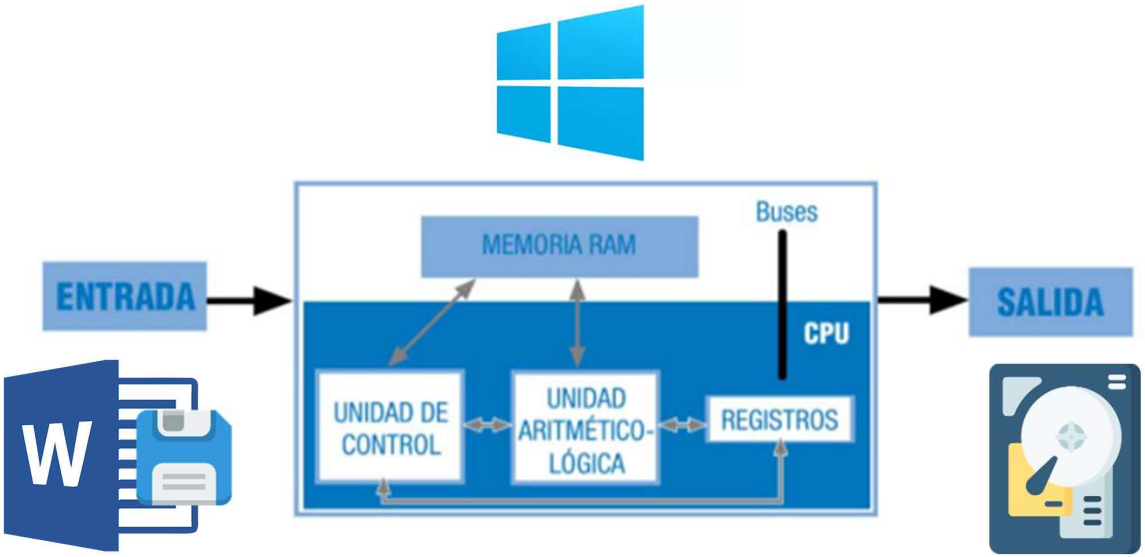
Unidad 1:

Elementos del desarrollo del software

Introducción

Introducción

Programa informático (software): **conjunto de comandos** ejecutados por el equipo (hardware) que permiten obtener, modificar y mostrar información. En definitiva, gestionar la información.



Licencia de software es un **contrato** que se establece entre el desarrollador de un software sometido a propiedad intelectual y a derechos de autor y el usuario final. Según su licencia podemos clasificar el software de 3 maneras:

1. **Software libre:** es aquel en el que el autor **cede** una serie de libertades básicas al usuario, en el marco de una licencia, que establece las siguientes libertades:
   - Libertad de **utilizar** el programa en cualquier contexto.
   - Libertad de **estudiar** como funciona el programa y de **adaptar** su código a necesidades específicas.
   - Libertad de distribuir **copias** a otros usuarios
   - Libertad de **mejorar** el programa y de hacer publicas y distribuir al público las modificaciones.

Software propietario: a diferencia del libre es aquel que se distribuye **sin** tener **acceso** al código fuente.

Software de dominio público: es aquel que carece de licencia y no hay forma de determinarla pues **se desconoce el autor**. La licencia más usada es la GPL (GNU General Public License – Licencia pública general)

Sistema operativo: software encargado de **coordinar al hardware** durante el funcionamiento del sistema informático, actuando de intermediario entre éste y las aplicaciones.

Los programas requieren de una serie de recursos hardware para la ejecución de los mismos, ya sea tiempo de CPU, espacio en memoria RAM, tratamiento de interrupciones, gestión de Entrada/Salida, etc.

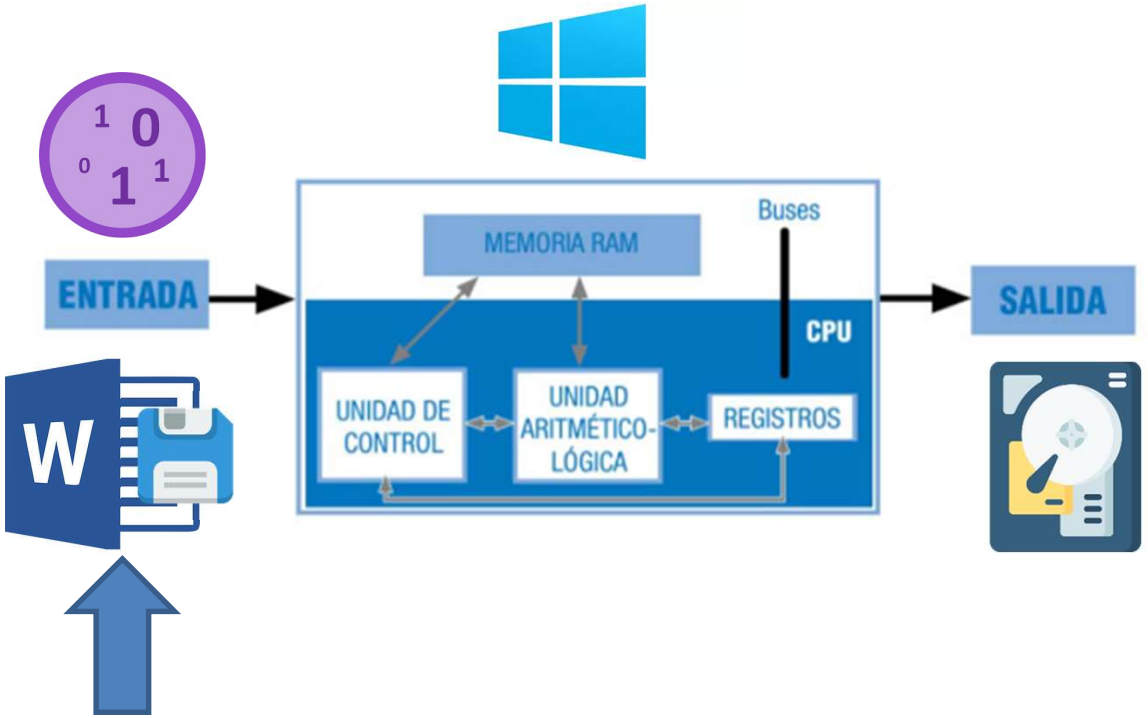
Introducción

```java
package paquete;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        int num = sc.nextInt();

        if(num>0)
        {
            System.out.println("El número es positivo");
        }
        else
        {
            if(num<0)
            {
                System.out.println("El número es negativo");
            }
            else
            {
                System.out.println("El número es cero");
            }
        }
        sc.close();
    }
}
```

Un programa estará formado por un archivo de texto (escrito a través de procesador o editor de texto) llamado archivo fuente o **código fuente**.

Los sistemas informáticos, a través de su CPU, únicamente son capaces de procesar código binario, es decir, una serie de 0s y 1s. A través del lenguaje de programación podremos **escribir de forma legible** lo que se necesite ejecutar.

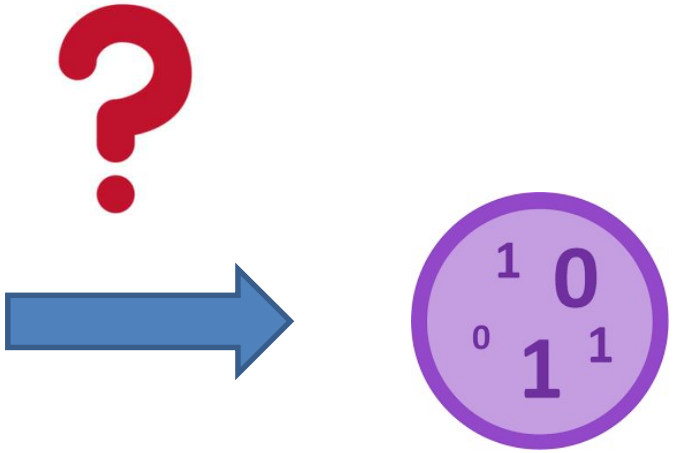
Introducción




```java
package paquete;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        int num = sc.nextInt();

        if(num>0)
        {
            System.out.println("El número es positivo");
        }
        else
        {
            if(num<0)
            {
                System.out.println("El número es negativo");
            }
            else
            {
                System.out.println("El número es cero");
            }
        }
        sc.close();
    }
}
```

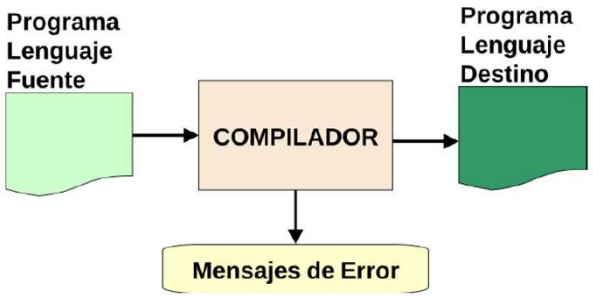
Introducción

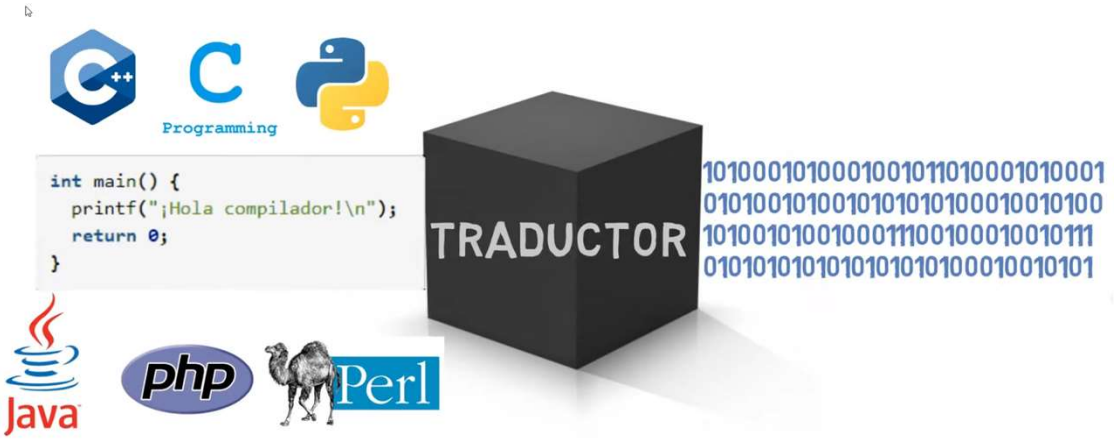

```java
package paquete;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        int num = sc.nextInt();

        if(num>0)
        {
            System.out.println("El número es positivo");
        }
        else
        {
            if(num<0)
            {
                System.out.println("El número es negativo");
            }
            else
            {
                System.out.println("El número es cero");
            }
        }
        sc.close();
    }
}
```



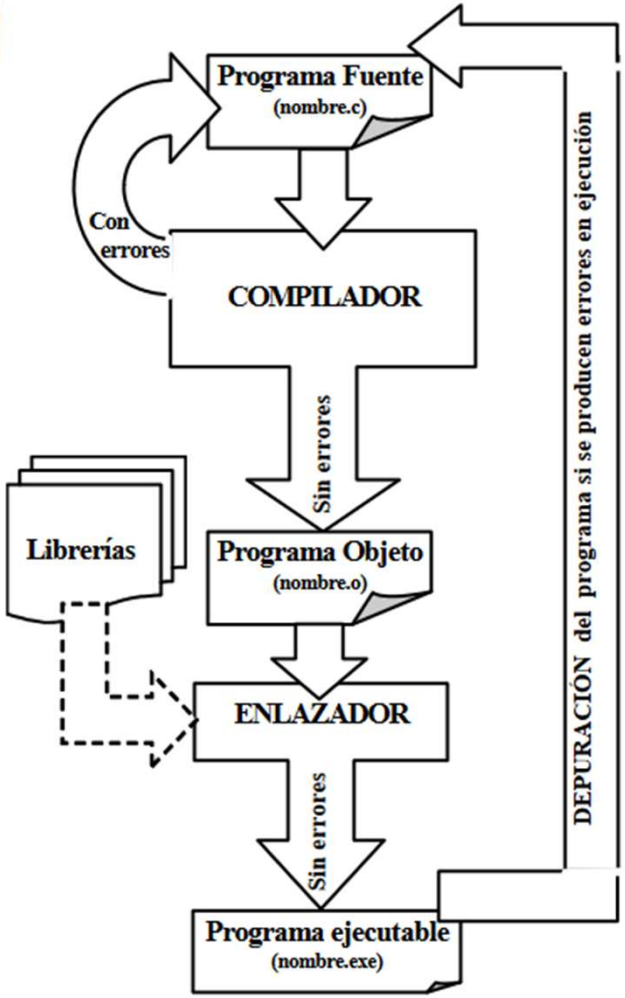
Introducción

Compilador: es el encargado de traducir el programa escrito al lenguaje de máquina, es decir, realizar la **compilación**.
Cada lenguaje de programación tiene su propio compilador (excepto los lenguajes interpretados).



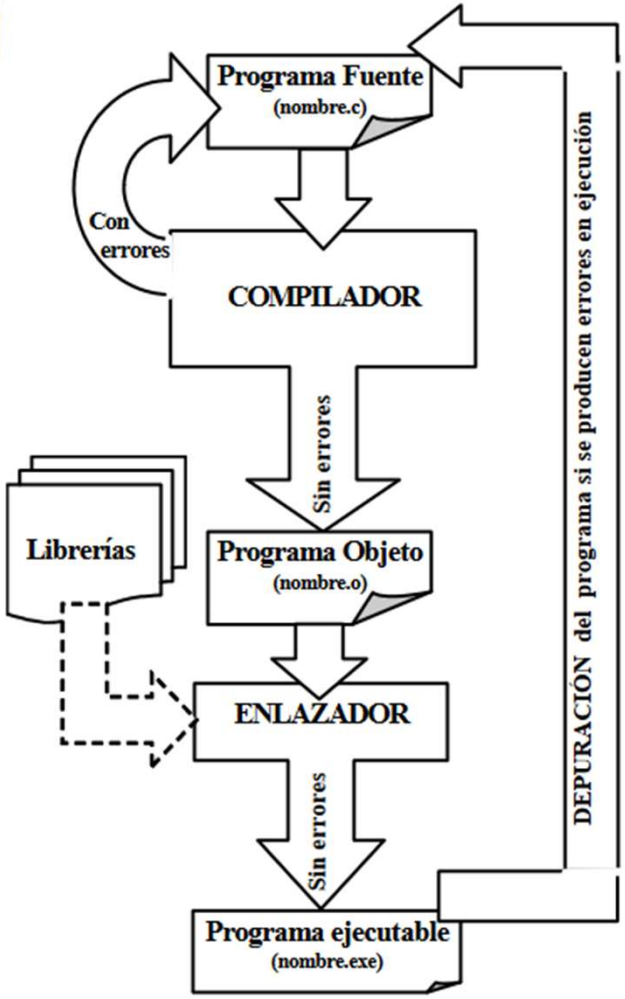


Introducción



Paso 1: El programador escribe el **código fuente** del programa.

Introducción

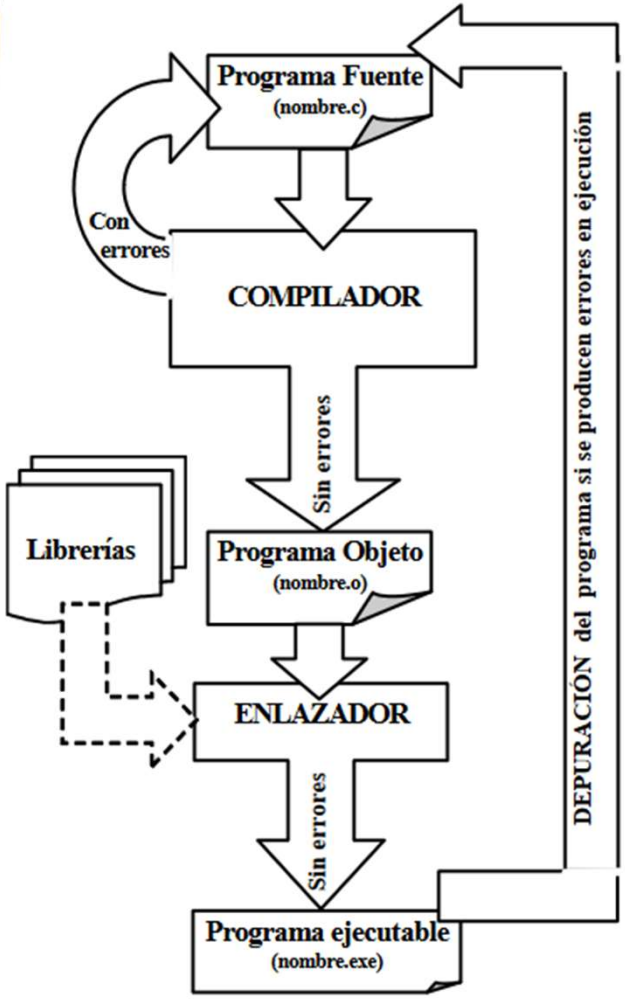


Paso 2: El compilador **traduce** el código fuente (lenguaje de alto nivel) en código objeto (lenguaje máquina) y lo guarda en un archivo objeto.

*Algunos compiladores crean un archivo en ensamblador.

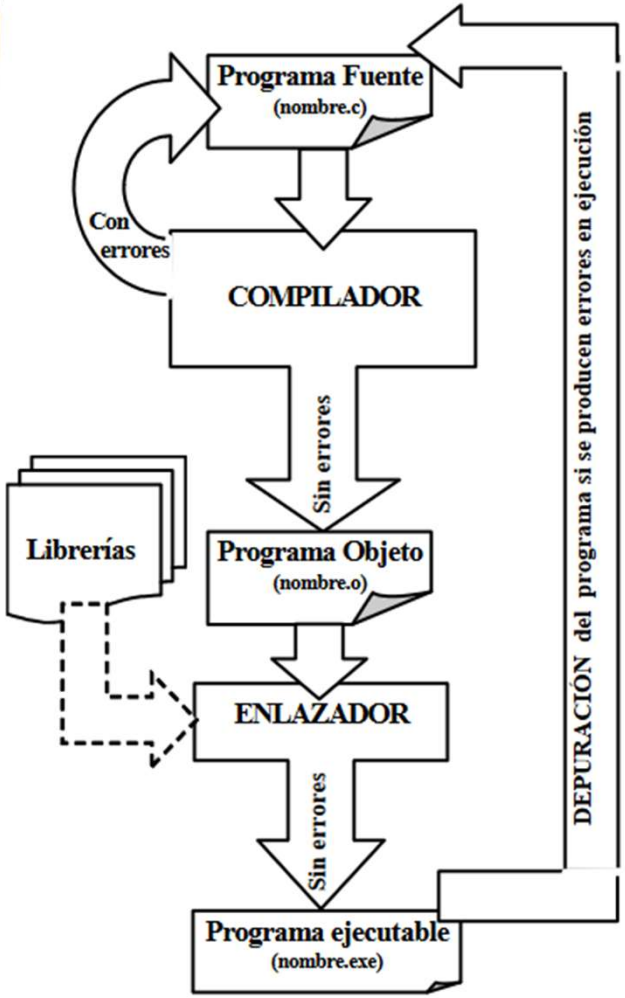
Lenguaje de Alto Nivel C	Lenguaje Ensamblador
SWITCH TYPE) { case 'a': type=type+10; break; case 'b': type= type+20; break; default: break;}	MOV1 R1 = Type LD4 R2 = [R1] ;; cmp.eq P1, P2 = 'a', R2 cmp.eq P3, P4 = 'b', R2 ;; (P1) Add R2 = 10, R2 (P3) Add R2 = 20, R2 ;; st4 (R1) = R2 default::

Introducción



Paso 3: El compilador llama a un editor de vínculos (enlazador o ensamblador) que permite **insertar los elementos adicionales** (funciones y bibliotecas) a los que hace referencia el programa dentro del archivo final.

Introducción



Paso 4: Finalmente tendremos nuestro archivo **ejecutable**, desde el cual arrancaremos la aplicación.

Actividad 1

Con el objetivo de desarrollar un programa informático que se ejecutará en un sistema de información...

a) Hay que escribir las instrucciones en código binario para que las entienda el hardware
b) Únicamente es necesario escribir el programa en algún lenguaje de programación y ejecutar directamente
c) Hay que escribir el programa en algún lenguaje de programación y contar con herramientas software que lo traduzcan a código binario
d) A través del enlazador transformaremos el código fuente en código binario para posteriormente generar el ejecutable

Actividad 1

Con el objetivo de desarrollar un programa informático que se ejecutará en un sistema de información...

a) Hay que escribir las instrucciones en código binario para que las entienda el hardware

b) Únicamente es necesario escribir el programa en algún lenguaje de programación y ejecutar directamente

c) Hay que escribir el programa en algún lenguaje de programación y contar con herramientas software que lo traduzcan a código binario

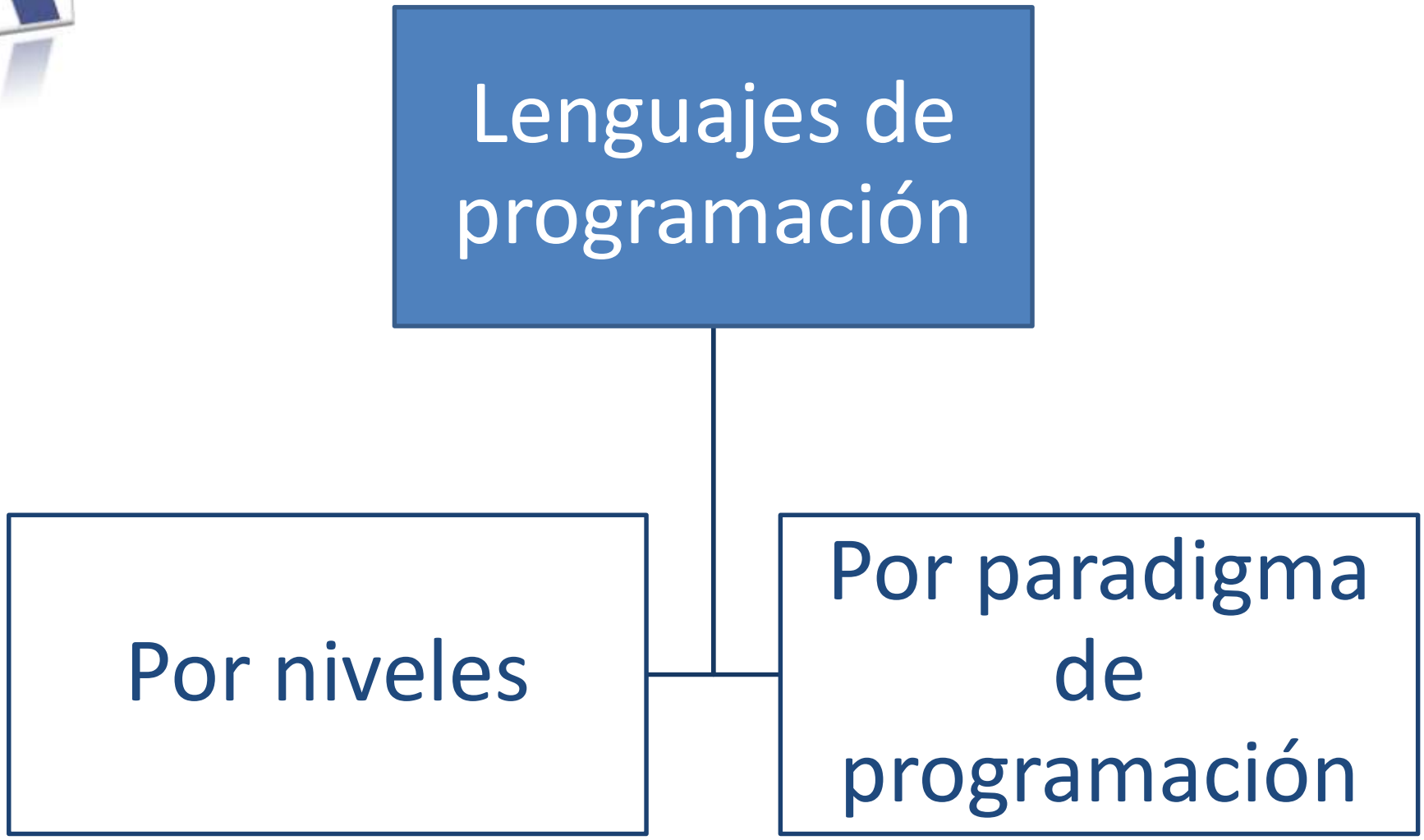
d) A través del enlazador transformaremos el código fuente en código binario para posteriormente generar el ejecutable

Ejemplo de compilación automatizada con Jenkins en Java

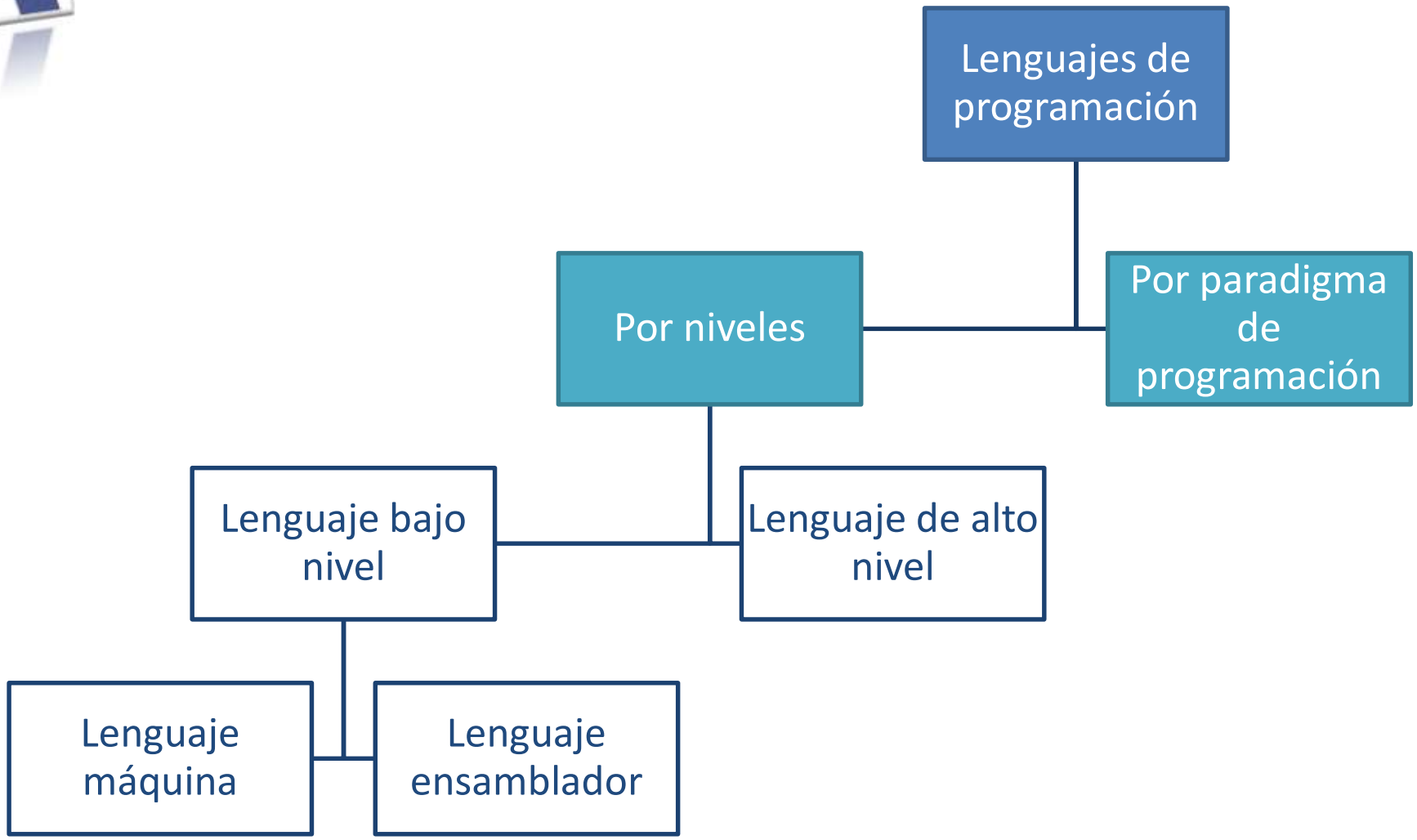
Automatización en Jenkins para proyecto Java

Tipos de lenguajes de programación

Tipos de lenguajes de programación



Lenguajes de programación por niveles



Lenguajes de programación por niveles

1. **Lenguaje de máquina**: es el lenguaje de programación que **entiende** directamente la **máquina**. Este lenguaje de programación utiliza el alfabeto **binario**, es decir, el 0 y el 1. Con estos dos únicos dígitos, se forman las cadenas binarias (combinaciones de ceros y unos) formando las instrucciones entendibles por el microprocesador.

```
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 11110101 00101011 00101011
11001010 11001010 11110101 00101011
11001010 11110101 00101011 00101011
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 11110101 00101011 00101011
```

Lenguajes de programación por niveles

1. **Lenguaje ensamblador:** Son más fáciles de utilizar que el lenguaje máquina, pero son **específicos de cada procesador**, con lo que si se ejecuta en otro procesador se deberá reescribir el programa.

A través del lenguaje ensamblador conseguimos crear **programas muy rápidos**, pero son **muy difíciles de aprender**.

```
; HOLA.ASM
; Programa clásico de ejemplo. Despliega una leyenda en pantalla.
STACK    SEGMENT STACK            ; Segmento de pila
           DW   64 DUP (?)        ; Define espacio en la pila
STACK    ENDS

DATA     SEGMENT                  ; Segmento de datos
SALUDO     DB   "Hola mundo!!",13,10,"$" ; Cadena
DATA     ENDS

CODE     SEGMENT                  ; Segmento de Código
       ASSUME CS:CODE, DS:DATA, SS:STACK

INICIO:                           ; Punto de entrada al programa
       MOV  AX,DATA               ; Pone dirección en AX
       MOV  DS,AX                 ; Pone la dirección en los registros
       MOV  DX,OFFSET SALUDO      ; Obtiene dirección del mensaje
       MOV  AH,09H                ; Función: Visualizar cadena
       INT   21H                  ; Servicio: Funciones alto nivel DOS
       MOV  AH,4CH                ; Función: Terminar
       INT   21H
CODE     ENDS
       END  INICIO                ; Marca fin y define INICIO
```

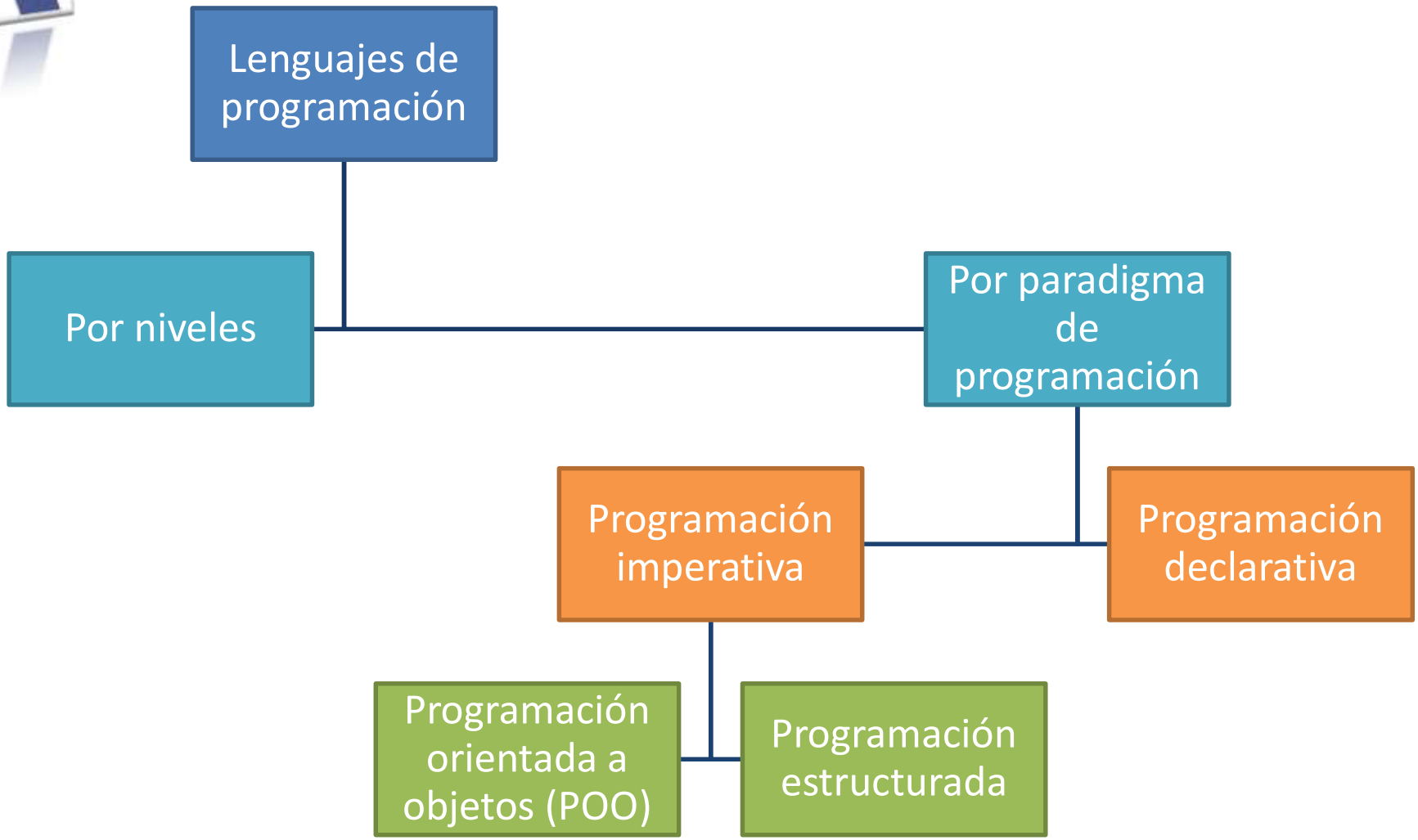
2. **De alto nivel**: Son por regla general, **independientes de la máquina**, y se puede usar en cualquier otra máquina con pocas modificaciones.

Este tipo de lenguajes **necesitan** de un **compilador** o **intérprete**, que traduzca este lenguaje de programación de alto nivel a uno de bajo nivel que la computadora pueda entender.

De forma general, este tipo de lenguajes son **fáciles de aprender** y suelen usar[illegible]**al**. Ej:// Java, Python, C++, …

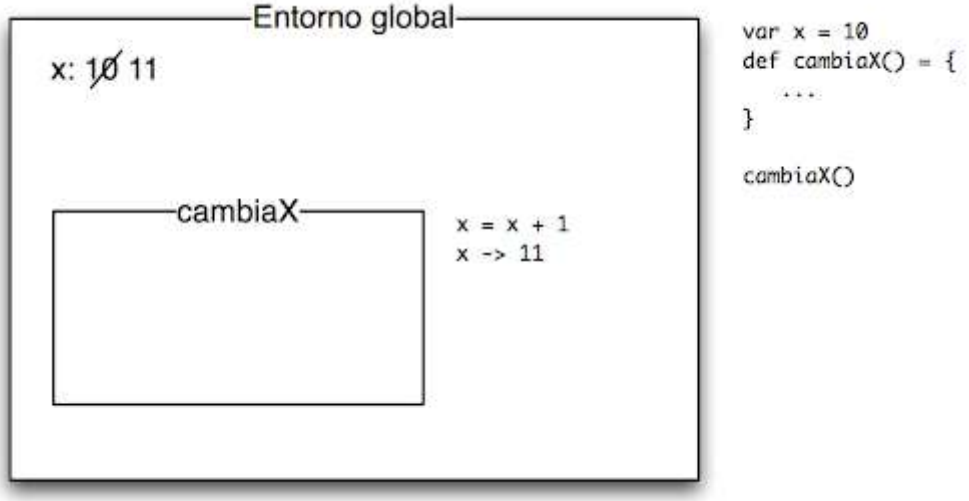
```
public class EjemploBreak {
    public static void main(String args[]){

        for (int i = 0; i < 10; i++) {
            if(i == 6) {
                break;
            }
            System.out.println("i: " + i);
        }
    }
}
```

Lenguajes de programación por paradigma de programación



Lenguajes de programación por paradigma de programación

1. **<u>Programación imperativa</u>**: Con este paradigma se resuelven los problemas **especificando** una secuencia de **acciones** a realizar a través de uno o varios procedimientos denominados funciones o subrutinas. Las sentencias **modifican el estado del programa** (el estado de un programa viene definido por la instrucción que se está ejecutando y el valor de los datos que está tratando en un momento dado).



Lenguajes de programación por paradigma de programación

2. **<u>Programación Declarativa</u>**: la solución es alcanzada a través de mecanismos internos de control, **no se especifica** exactamente **cómo llegar** a ella pero sí la solución al problema.

```
1 Select * from Apps
```

Query Favorites Query History

AppID	AppName	CreatorName	AppCategory	AppPrice
1	AppDividend	Krunal	Software	50
2	Escrow	LVVM	Fashion	60
3	KGB	MJ	Music	70
4	Moscow	Mayor	Area	80
5	MoneyControl	Mukesh	Investment	90
6	Investing	Bill	Stocks	100

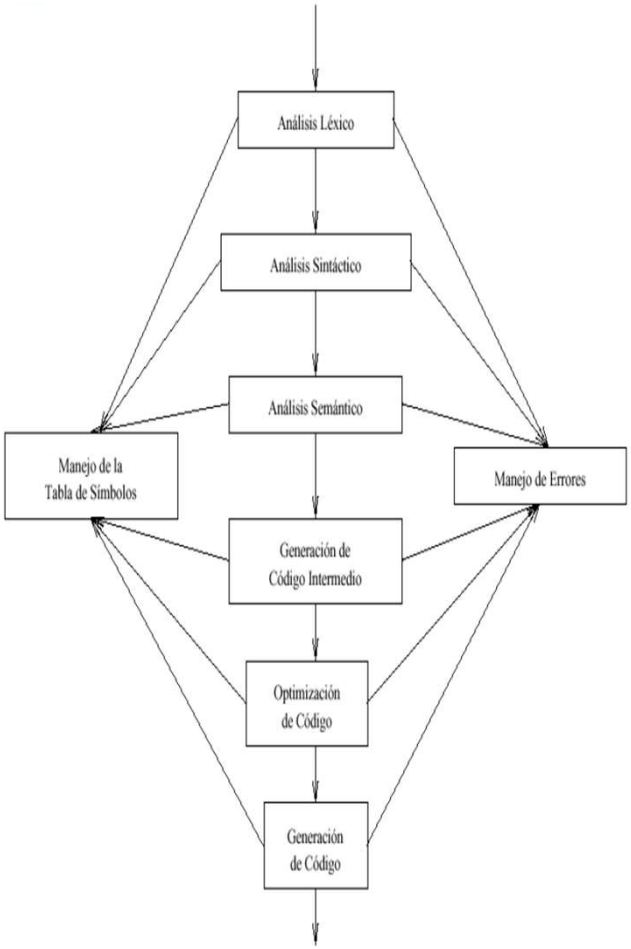
Ranking lenguajes de programación

Language Ranking: IEEE Spectrum

Rank	Language	Type	Score
1	Python		100.0
2	Java		95.4
3	C		94.7
4	C++		92.4
5	JavaScript		88.1
6	C#		82.4
7	R		81.7
8	Go		77.7
9	HTML		75.4
10	Swift		70.4

Compiladores

Compiladores - Fases



La traducción de un compilador consta de **2 etapas**: la etapa de análisis y la etapa de síntesis.

Durante la compilación **pueden generarse mensajes de error**, comunicándose al programador mediante un mensaje del error producido y el lugar en que ocurrió. En algunos casos, ciertos errores no afectan gravemente al proceso de ejecución, e incluso permiten la ejecución del programa. Este tipo de errores se denominan advertencias o **warnings**.

Compiladores – Fase de análisis

En esta fase se dan **tres fases** fundamentales realizadas cada una por un analizador determinado:

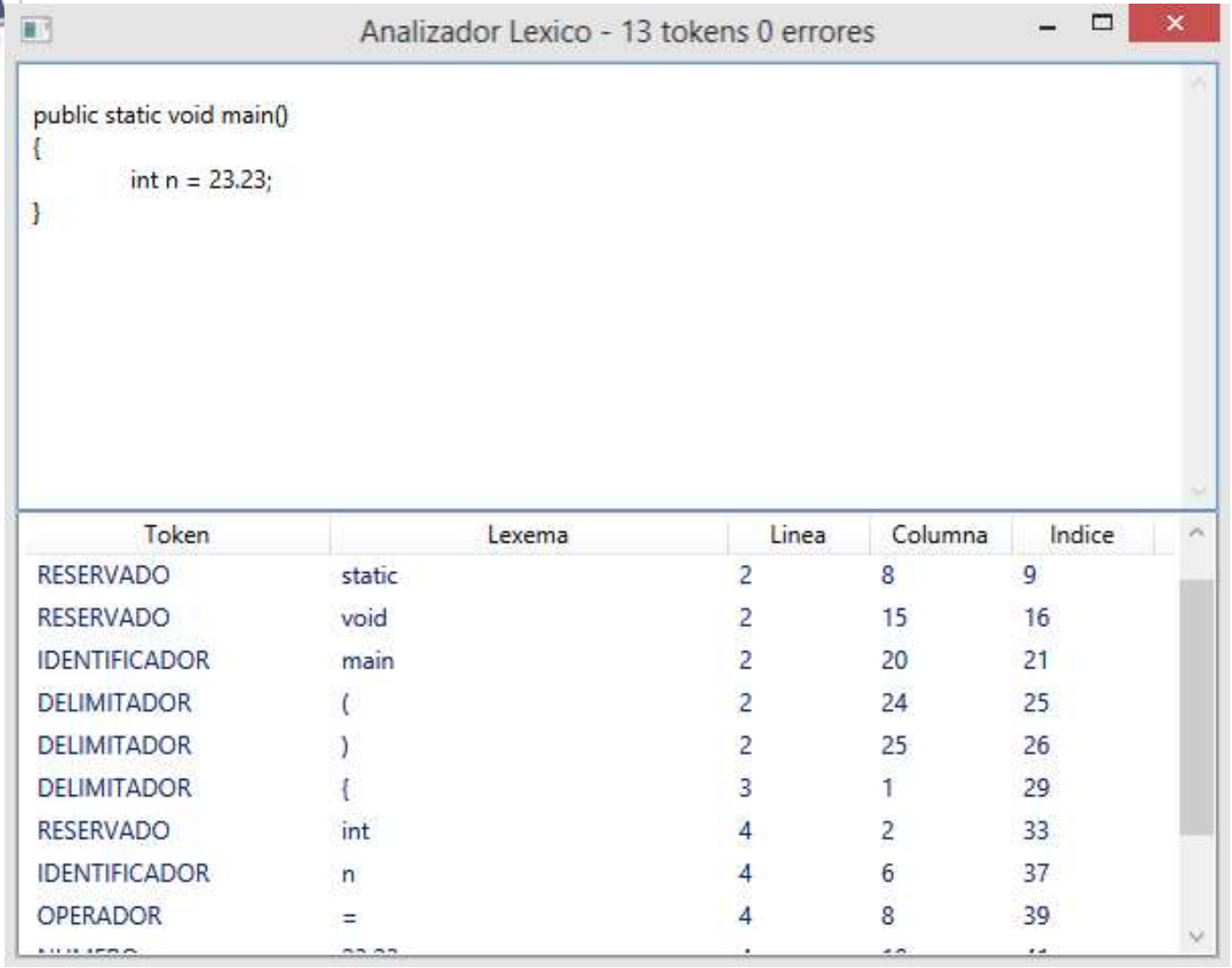
1. Análisis léxico
2. Análisis sintáctico
3. Análisis semántico

Compiladores – Fase de análisis

Analizador léxico: Examina el programa fuente para **localizar** las unidades básicas de información pertenecientes al lenguaje (palabras reservadas, identificadores, constantes, etc.), denominadas **tokens**.

Se identifica el tipo de token y almacena en unas tablas denominadas **tablas de símbolos**, la cual está relacionada con todas las fases del proceso de compilación y está presente en todas ellas. En algunas fases se ocupan de **completar** esta tabla y en otras de **utilizar** la información.

Compiladores – Fase de análisis



Token	Lexema	Linea	Columna	Indice
RESERVADO	static	2	8	9
RESERVADO	void	2	15	16
IDENTIFICADOR	main	2	20	21
DELIMITADOR	(	2	24	25
DELIMITADOR	)	2	25	26
DELIMITADOR	{	3	1	29
RESERVADO	int	4	2	33
IDENTIFICADOR	n	4	6	37
OPERADOR	=	4	8	39

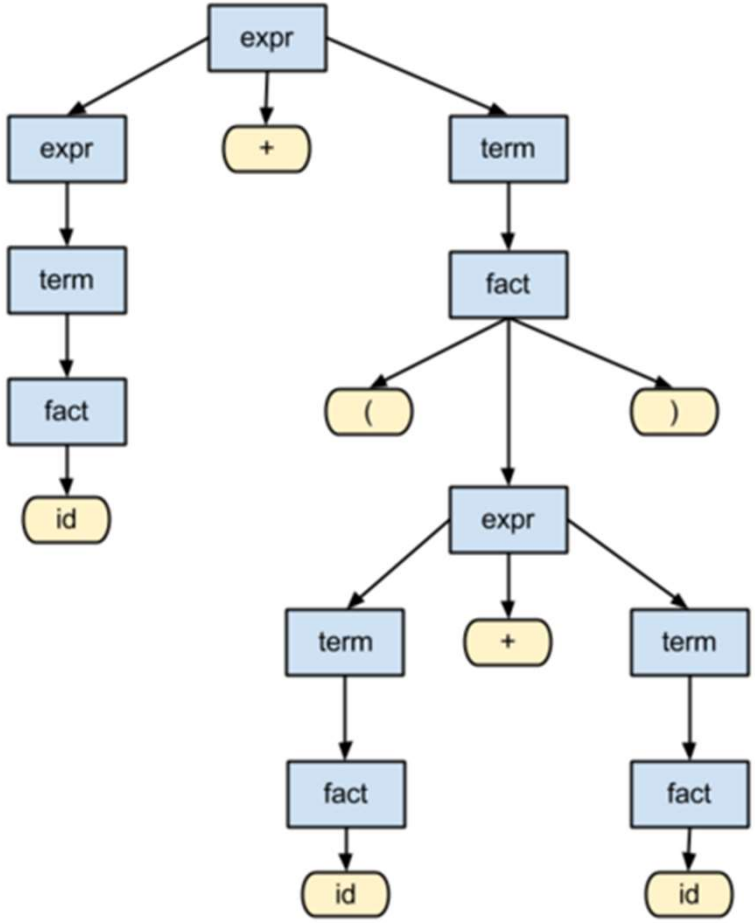
Analizador sintáctico: La sintaxis de un lenguaje de programación especifica **cómo deben escribirse los programas**, mediante un conjunto de reglas de sintaxis o gramática del lenguaje. Un programa es sintácticamente correcto cuando sus estructuras están escritas en un orden correcto.

De esta forma el analizador sintáctico recibe la cadena de tokens obtenida por el analizador léxico y busca en ella los posibles errores sintácticos que aparezcan.

✓ Ej. correcto : int numero = 4*(3+1)

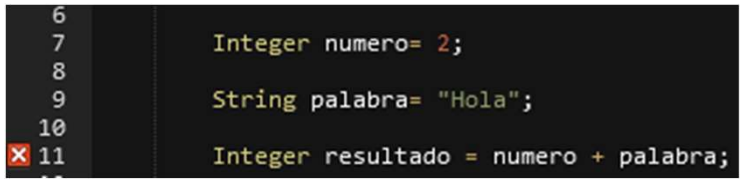
x Ej. incorrecto: int numero = 4*3+1)

Compiladores – Fase de análisis



Analizador semántico: La fase de análisis semántico **revisa** el programa fuente para tratar de encontrar errores semánticos y reúne la **información sobre los tipos** para la fase posterior de generación de código.

En el análisis semántico se realizan, entre otras funciones, la verificación de los tipos de las variables declaradas, comprobando si cada operador tiene **operandos permitidos** por la especificación del lenguaje.



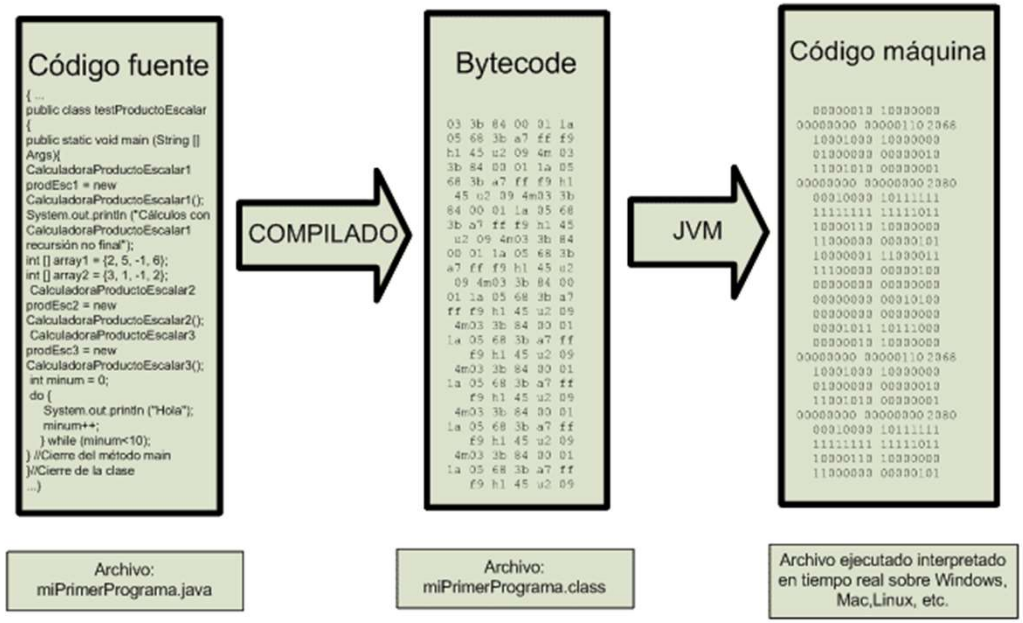
Compiladores – Fase de síntesis

En esta etapa se dan **dos fases**:

1. Generación del código intermedio
2. Optimización de código

Compiladores – Fase de síntesis

Generación de código intermedio: Si no se han producido errores en alguna de las etapas anteriores, se realiza la traducción a un código interno propio del compilador, denominado código intermedio, a fin de permitir la **transportabilidad** del lenguaje a otros ordenadores.

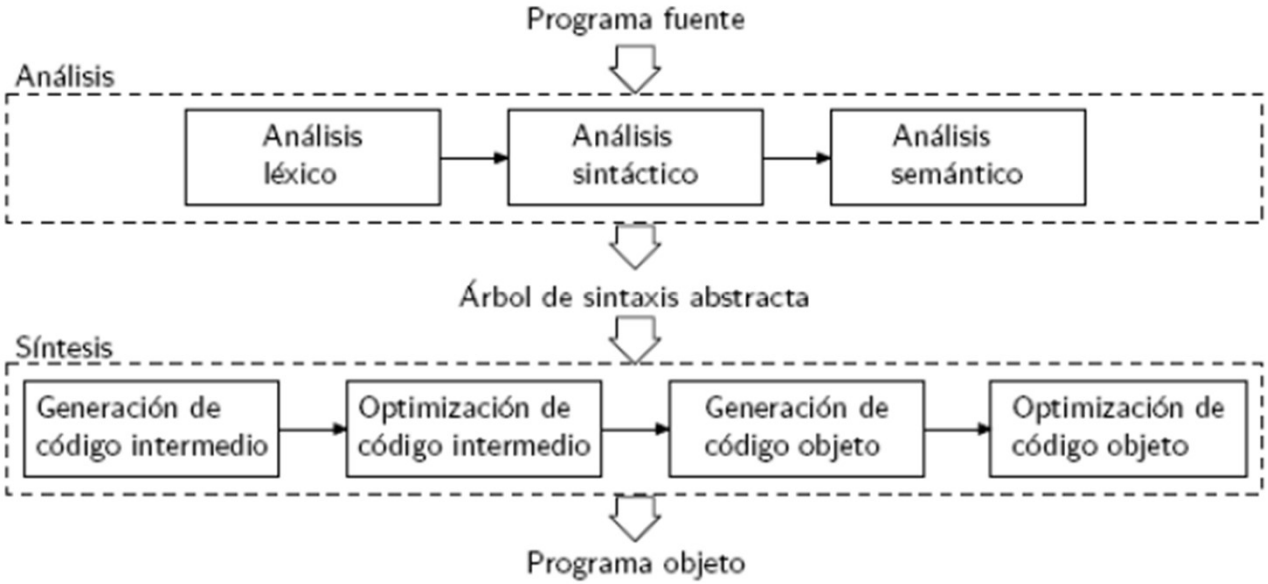


Compiladores – Fase de síntesis

Optimización de código: Recibe el código intermedio y lo **optimiza** atendiendo a determinados factores, tales como la velocidad de ejecución o el tamaño del programa objeto.

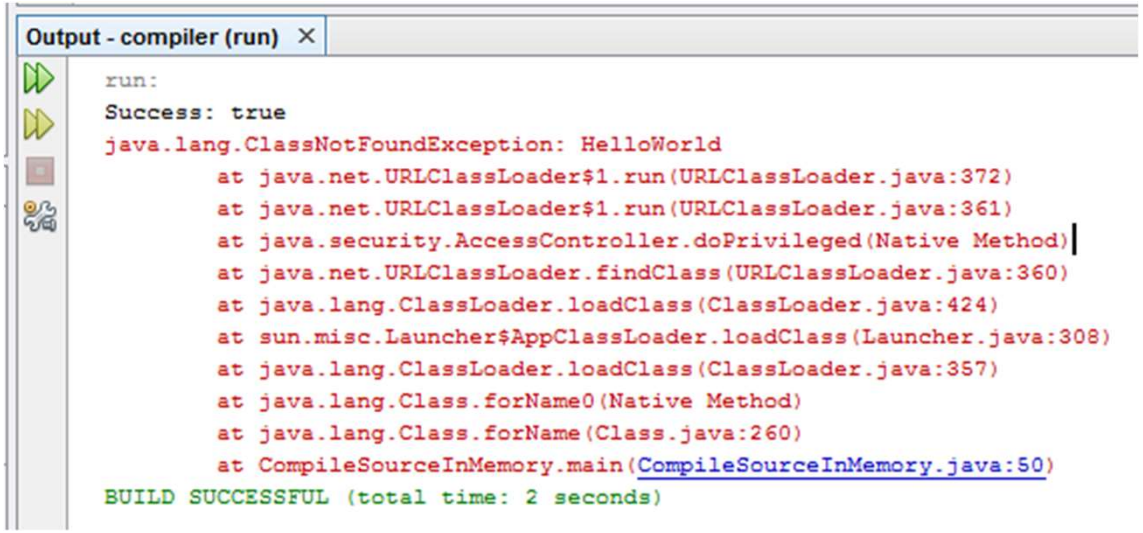
Tras la optimización del código intermedio se lleva a cabo la generación del código final del programa (**Programa objeto**).

Compiladores – Resumen



Depuradores

Depuradores: facilitan la **detección** y, en muchos casos, la **recuperación** de los errores producidos en las diferentes fases de la compilación. El compilador, cuando detecta un error, trata de buscar su localización exacta y su posible causa para presentar al programador un mensaje de diagnóstico, que será incluido en el listado de compilación.



Actividad 2

Compilar un programa Java con errores léxicos, sintácticos y semánticos.

A continuación, solucionar el problema y ver el programa objeto generado.

Fases del desarrollo de una aplicación

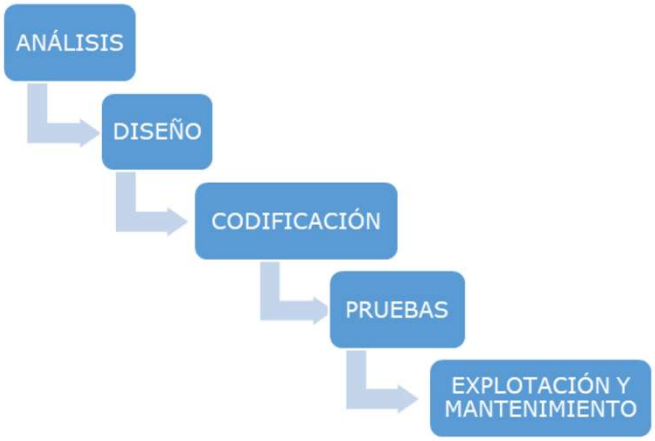
Fases del desarrollo de una aplicación

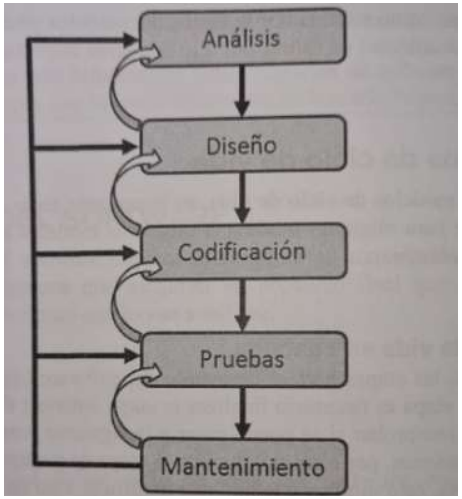
Ciclo de vida del software: marco de referencia que contiene los **procesos**, las **actividades** y las **tareas** involucradas en el desarrollo, la operación y el mantenimiento de un producto software, abarcando la vida del sistema desde su definición hasta su retirada.

Fases del desarrollo de una aplicación

Ciclo de vida en cascada: En este modelo las etapas para el desarrollo del software tienen un **orden**, no se puede pasar a la siguiente sino se ha terminado la anterior. Después de cada etapa se realiza una revisión para ver si se puede pasar a la siguiente.

Una variante es el modelo en cascada con retroalimentación (**permite retroceder** una o varias etapas si se requiere en algún momento).





Ventajas:

- **Fácil** de comprender, planificar y seguir.
- La **calidad** del producto resultante es alta.

Inconvenientes

- La necesidad de tener **todos los requisitos** definidos desde el principio.
- Es **difícil volver atrás** si se cometen errores en una etapa.
- El producto **no** está **disponible** para su uso hasta que se termina.

Se recomienda cuando:

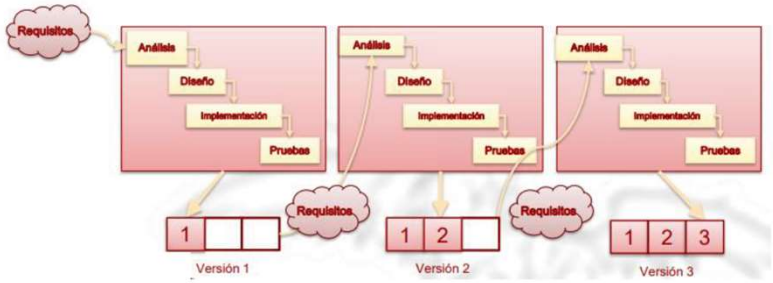
- El **proyecto** es **similar** a algunos que ya se haya realizado con éxito.
- Los **requisitos** son **estables**.
- Los clientes **no** necesitan **versiones intermedias**.

Fases del desarrollo de una aplicación

Ciclo de vida evolutivos: permiten desarrollar **versiones** cada vez más completas hasta llegar a un producto final deseado.

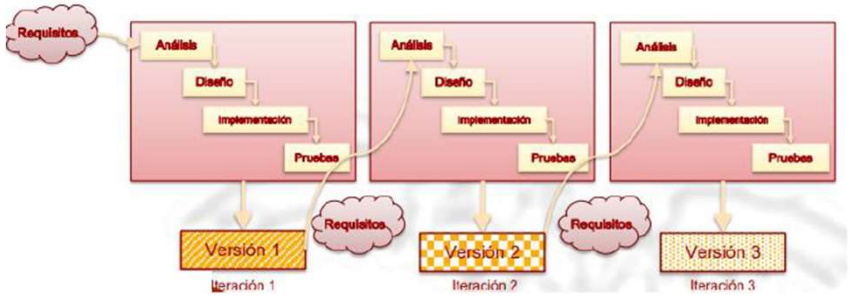
Los más conocidos:

1. **Incremental**: al final de cada ciclo se entrega una **versión parcial** del software con ciertas funcionalidades nuevas respecto a la anterior.



Fases del desarrollo de una aplicación

2. **<u>Iterativo</u>**: los ciclos se repiten hasta obtener un producto satisfactorio y los usuarios deben evaluar el producto en cada iteración y proponer mejoras.



Iterativo

Fases del desarrollo de una aplicación

Ventajas:

- **No** se necesita **conocer** todos los **requisitos** desde el principio
- Permite la **entrega temprana** al cliente de partes operativas
- Las entregas facilitan la **retroalimentación** de los próximos entregables.

Inconvenientes:

- Es **difícil estimar** el esfuerzo y el coste final.
- Se tiene el riesgo de **no acabar** nunca.

Se recomienda para:

- Los **requisitos** o el diseño **no** están **completamente definidos**.
- Se están probando o introduciendo **nuevas tecnologías**.

Fases del desarrollo de una aplicación

Análisis: En esta fase se analizan y especifican los **requisitos** o capacidades que el sistema debe tener porque el cliente así lo ha pedido.

Es necesario obtener unos buenos requisitos y para ello es esencial una buena **comunicación** entre el cliente y los desarrolladores. Se utilizan multitud de **técnicas**, entre las que destacan:

- **Entrevistas**: Lógicamente es la técnica mas tradicional.
- **Brainstorming**: Es una técnica utilizada en el trabajo de equipo para generar nuevas ideas de manera rápida y escueta.
- **Prototipos**
- **Casos de uso**

Fases del desarrollo de una aplicación

Se especifican dos tipos de requisitos:

Requisitos funcionales: Describen con detalles la **función** que realiza el sistema, como reacciona ante determinadas entradas, como se comporta en situaciones particulares, etc.

Requisitos no funcionales: Tratan sobre las **características del sistema**, como puede ser la fiabilidad, mantenibilidad, sistema operativo, plataforma hardware, restricciones, limitaciones, etc.

Actividad 3

Caso práctico, una aplicación para las TPV de un restaurante.

- *¿Qué tipo de modelo de desarrollo de software vamos a hacer?*
- *Hacer de cliente y en lenguaje coloquial decir que queremos que nos fabriquen.*
- *¿Son las necesidades del cliente claras?*
- *¿Cuántas veces creéis que nos reuniríamos con el cliente? ¿Y por qué?*
- *Obtener requisitos funcionales, no funcionales e información que debe incluir.*

Fases del desarrollo de una aplicación

Diseño: Se alcanza con mayor precisión una **solución** óptima de la aplicación. Se deben tener en cuenta los **recursos físicos** del sistema (tipo de equipos, servidores, periféricos, comunicaciones, etc.) y los **recursos lógicos** (sistemas operativos, programas de utilidad, lenguaje/s de programación, bases de datos, etc.) donde se ejecutará la aplicación.

Etapas de la fase de diseño:

- **Diseño externo**: Especifica la **interfaz** para los **usuarios**, formatos de información de entrada y salida (pantalla y listados de información, gráficos, etc.).
- **Diseño de datos**: Establece las **estructuras de datos** de acuerdo con su soporte físico y lógico.
- **Diseño modular**: Representación de forma descendente de la **división** de la aplicación **en módulos** normalmente a través de interfaces.
- **Diseño procedimental**: Establece las **especificaciones para cada módulo**, escribiendo el algoritmo (pseudocódigo) necesario que permita posteriormente una rápida codificación.

Actividad 4

Caso práctico, una aplicación para las TPV de un restaurante:

- *Obtener diseño externo, diseño de datos, diseño modular y diseño procedimental.*
- *¿Y ahora?¿Qué tenemos que decidir antes de pasar a la codificación?*

Fases del desarrollo de una aplicación

Codificación: Se procede a traducir y **crear el código fuente** en el lenguaje de programación que se decidió en la etapa de diseño, además del entorno u entornos de desarrollo y las tecnologías a utilizar.

Actividad 5

Caso práctico, una aplicación para las TPV de un restaurante:

- *Indicar que lenguaje de programación vamos a usar.*
- *Herramientas que se usarán para la obtención del código fuente, objeto y ejecutable. Explicando el por qué.*

Pruebas: **comprobar la calidad** y **estabilidad** del programa.

Existen dos métodos generales en el diseño de pruebas de programas:

- **Caja blanca**: Permiten examinar la estructura interna del programa.
- **Caja negra**: Únicamente se diseñan considerando las entradas y salidas del sistema, sin preocuparse de la estructura interna.

Fases del desarrollo de una aplicación

Existen 3 fases en el proceso de pruebas:

1. **Pruebas de unidad**: Están destinadas a la **prueba** de cada una de las unidades del sistema, centrándose en la verificación de la menor unidad en el diseño del software (**el módulo**).

2. **Pruebas de integración**: Probar la correcta **interconexión** de **los módulos**, ya que esto no queda garantizado con las pruebas de unidad. Algunos de los problemas que pueden surgir son la modificación no deseada de variables globales (causando posibles efectos laterales) y el acarreo de imprecisiones por parte de varias funciones involucradas en el mismo cálculo.

   La **integración** de los módulos ha de realizarse de manera **incremental**, es decir, el programa se irá construyendo añadiendo los módulos poco a poco, de manera que los posibles errores sean fácilmente localizables y corregibles en cada uno de los ellos.

Fases del desarrollo de una aplicación

Según la estrategia de integración elegida, se pueden distinguir entre una integración entre módulos ascendente o bien una integración descendente.

- **Ascendente**: **comienza** con la construcción y prueba de los **módulos de más bajo nivel** y va realizando la integración de los mismos en módulos mayores.

- **Descendente**: se construye **primero** el **programa principal**, y a éste se le van incorporando los módulos conforme se vayan implementando.

3. **Pruebas de sistema**: En esta fase, el sistema construido debe quedar en **perfecto estado de funcionamiento**. Para ello se realizan diversas tareas como:

- Adaptar y ultimar detalles acerca del hardware sobre el que finalmente se instalará el software.
- Forzar errores en el sistema para comprobar su capacidad de recuperación.
- Verificar los mecanismos de protección y seguridad incorporados.
- Realizar pruebas que evalúen el rendimiento del sistema.
- Realizar pruebas que pongan al sistema en situaciones anormales, con la finalidad de evaluar su estabilidad.

Fases del desarrollo de una aplicación

Actividad 6

Caso práctico, una aplicación para las TPV de un restaurante, obtener:

- *Pruebas unitarias*
- *Pruebas de integración*
- *Pruebas de sistema.*

Fases del desarrollo de una aplicación

Explotación: consiste en realizar la **implantación de la aplicación** en el sistema o sistemas físicos donde van a funcionar habitualmente (entorno de producción) y su puesta en marcha para comprobar el buen funcionamiento.

Para ello se realizará:

- Instalación del/los programa/s.
- Eliminación del sistema anterior.
- Conversión de la información del antiguo sistema al nuevo (si la hubiera).
- Pruebas de aceptación al nuevo sistema.

Al final de la explotación se le facilita al cliente toda la **documentación** necesaria para la **explotación** del sistema (manuales de ayuda, de uso, guía de la aplicación, información sobre el soporte, etc.).

Fases del desarrollo de una aplicación

Mantenimiento: es la fase que completa el ciclo de vida y en ella se deben **solventar** los posibles **errores** o **deficiencias** de la aplicación. Es posible que se necesiten aclarar nuevas especificaciones que el cliente no informó de forma correcta.

Existen fundamentalmente 3 tipos de mantenimiento:

- **Mantenimiento correctivo**: **Corregir errores no detectados** en pruebas anteriores y que aparezcan en el uso normal de la aplicación.
- **Mantenimiento adaptativo**: Consiste en modificar el programa a causa de **cambio de entorno** gráfico y lógico en el que estén implementados.
- **Mantenimiento perfectivo**: Consiste en una **mejora** sustancial **de la aplicación** al recibir por parte de los usuarios propuestas sobre nuevas posibilidades y modificaciones de las existentes.

Documentación durante las fases del desarrollo

Documentación durante las fases del desarrollo

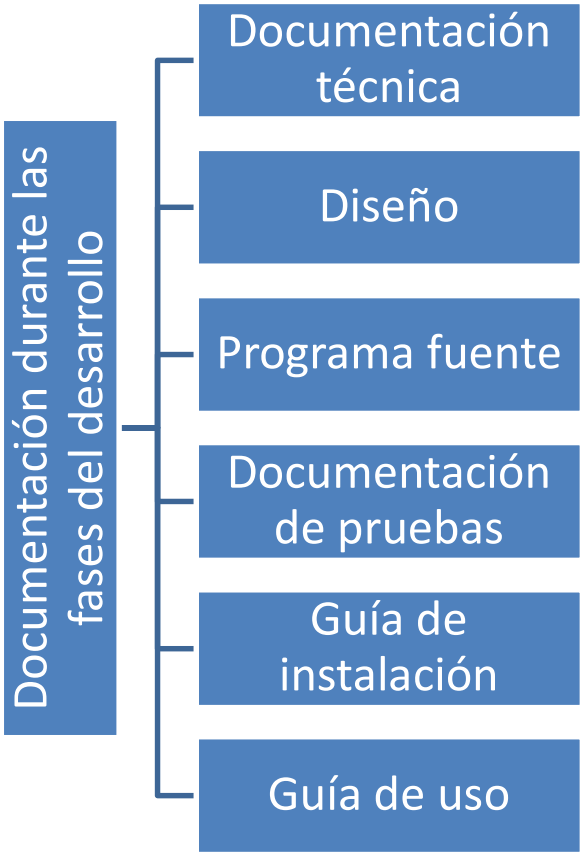
Durante todas las fases del desarrollo del software es muy importante ir **documentando todo**, utilizando tanto documentación externa como interna.

Es **MUY IMPORTANTE** comentar el código a la hora de programar.

Es de vital importancia dejarles un buen **manual de instalación** y **uso** de la aplicación.

Además de disponer de todo el código documentado para que sea más entendible para **posteriores cambios** o **mejoras**.

Documentación durante las fases del desarrollo



Documentación durante las fases del desarrollo

Documentación técnica: La guía técnica o manual técnico es el documento donde queda reflejado el **diseño** del proyecto o aplicación, la **codificación** de los programas y las **pruebas** realizadas para su correcto funcionamiento.

Está destinado al personal técnico en informática (**analistas** y **programadores**). El principal objetivo de este documento es el de **facilitar el desarrollo**, la **corrección** y el futuro **mantenimiento** de los programas de forma rápida y precisa.

Documentación durante las fases del desarrollo

Diseño: Este documento está **destinado** a los **programadores** que van a desarrollar los programas de la aplicación, con objeto de que la codificación sea efectuada con la mayor calidad y rapidez posible.

Este documento debe estar realizado de tal forma que permita la división del trabajo de programación, para que varios programadores puedan trabajar de forma independiente, aunque coordinados por uno de los programadores o por un analista.

Documentación durante las fases del desarrollo

Programa fuente: En el código fuente escrito en los programas se debe cuidar que éste sea **claro**, **legible** y bien **documentado**.

En el estilo de escritura se deben tener en cuenta las siguientes normas:

- Utilizar suficientes espacios y líneas en blanco para **separar bien las expresiones y los bloques de código** cuya funcionalidad sea diferente a la del código adyacente en el mismo fichero fuente.
- Utilizar siempre tabuladores para **delimitar** el ámbito de las **estructuras de control**, respetando el sangrado (espacios en blanco introducidos delante de una línea) de las estructuras anidadas.

Documentación durante las fases del desarrollo

- Utilizar **paréntesis**, ya que, en expresiones complicadas, resulta más rápido desglosar mentalmente la expresión si ésta tiene los paréntesis apropiados.
- Utilizar **separadores visuales**, como guiones entre comentarios, para separar los grandes bloques del programa.
- Utilizar **palabras representativas** para los nombres de identificadores (variables y constantes), evitando nombres muy cortos (poca legibilidad) o muy largos (incómodos a la hora de programar, a la vez que alargan mucho la línea de código).
- Intentar **simplificar el código**, evitando expresiones complicadas en la medida de lo posible.
- **Evitar** grandes **anidamientos** si no son necesarios.
- **Diferenciar** claramente las **zonas** de inclusiones e importaciones, declaraciones de identificadores, procedimientos y funciones.

Documentación durante las fases del desarrollo

En relación a los comentarios, se debe tener en cuenta lo siguiente:

- **Clarifican** la **estructura** de cada módulo. Se incluirán comentarios que clarifiquen la estructura de cada módulo, así como la función realizada por cada bloque de instrucciones y las variables y tipos.
- Son recomendables al principio de cada procedimiento o función, **aclarando** la **funcionalidad** del mismo, además es recomendable indicar en éstos también el significado de los argumentos tomados y el valor devuelto. En cualquier lugar del código en que se vaya a realizar una tarea complicada de comprender es recomendable la aclaración mediante un comentario explicativo.

Actividad 7:

¿Por qué es recomendable realizar este tipo de comentarios?

Actividad 7:

¿Por qué es recomendable realizar este tipo de comentarios?

- El propio programador puede requerir la reutilización de un código escrito tiempo atrás, habiendo ya olvidado en la fecha presente el código que se escribió.
- La programación no es tarea de una única persona, y al tener que compartir el código con otras personas debe aclarar de la mejor manera posible la forma en que ha escrito dicho código, para facilitar la tarea al resto del grupo.

Documentación durante las fases del desarrollo

Documentación de pruebas: Se pueden **especificar** el tipo de prueba, módulos y programas para los que se efectúan dichas pruebas, datos de entrada, datos previstos de salida y los datos reales de salida obtenidos en las pruebas.

Documentación durante las fases del desarrollo

Principios para la realización de pruebas:

- Las pruebas deben ser llevadas a cabo por **personas diferentes** a las que desarrollaron el programa.
- Los casos de pruebas deben ser escritos tanto para condiciones de **entradas inválidas** o inesperadas como para condiciones **válidas** y esperadas.
- La posibilidad de encontrar errores adicionales en una sección del programa es **proporcional** al número de errores ya encontrados en la misma sección.

Guía de instalación: es el documento que contiene la información necesaria para **poner en marcha el sistema** diseñado y para establecer las normas de explotación.

Estas normas harán precisiones sobre las siguientes tareas:

- **Pruebas de implantación** de los programas en el sistema físico donde van a funcionar en adelante, con la colaboración entre usuarios y desarrolladores.
- Forma en que se van a **capturar los datos existentes** en el sistema anterior al que se va a implantar (trasvase de información).
- Pruebas del nuevo sistema con toda la información capturada (tener funcionando en paralelo los dos sistemas y cuando las pruebas sean satisfactorias sustituir el anterior sistema por el nuevo) .

Documentación durante las fases del desarrollo

Guía de uso: es el documento que contiene la información precisa y necesaria para que **los usuarios** utilicen correctamente los programas de la aplicación. Se presenta de forma que el usuario la comprenda con toda claridad, prescindiendo de toda la parte técnica de desarrollo y centrándose en los aspectos de la entrada y salida de la información que maneja la aplicación.

La guía de uso debe estar **redactada** con un estilo claro y en el caso de que se haga necesario el uso de terminología informática no conocida por los usuarios, debe ir acompañada por un glosario de términos informáticos.

FIN
UNIDAD