Introducción al desarrollo del software

Entornos de desarrollo

Clasificación de los lenguajes según su ejecución

Un programa escrito en lenguaje de alto nivel tiene que traducirse a un código que entienda la máquina, los programas traductores son de dos tipos: compiladores o intérpretes.

- **Lenguajes compilados**: Un compilador es un programa que traduce un programa escrito en un determinado lenguaje (código fuente) a un lenguaje máquina (código objeto). Si el código fuente tiene errores el compilador devolverá errores y no se generará el fichero objeto.
- **Lenguajes interpretados**: Cada vez que se ejecuta una instrucción, se debe interpretar y traducir a lenguaje máquina sin producirse un fichero objeto. Ejemplos de este tipo de lenguajes son: PHP, JavaScript, Python, Perl, Logo, Ruby, ASP, etc.

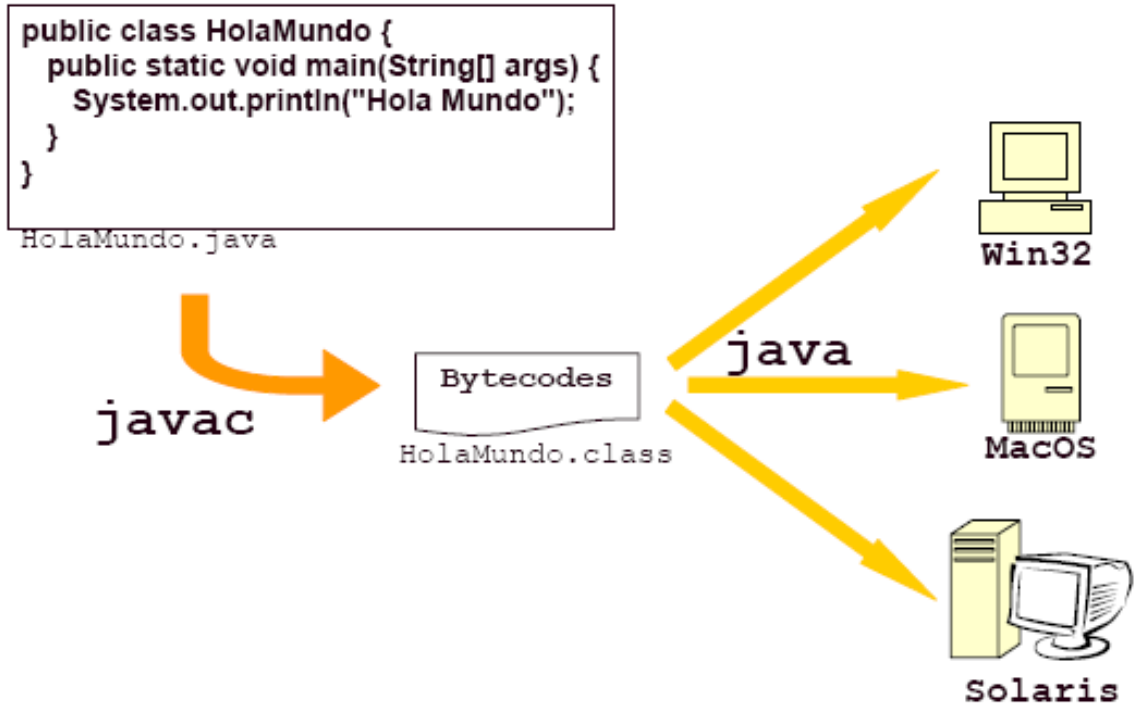
El programa destino en lenguaje máquina que produce un compilador es normalmente más rápido que un programa escrito en un lenguaje interpretado, pero cada vez que se realiza una modificación del código fuente hay que volver a repetir todo el proceso.

Java

- Lenguaje compilado creado para el uso en electrodomésticos en 1991.
- Entre sus diversos uso, permite crear:
  - ✓ Aplicaciones independientes, **Stand-alone Application**.
  - ✓ Aplicación ejecutada en el navegador, **Applet**.
  - ✓ Aplicación ejecutada en el servidor, **Servlet**.
- Para poder desarrollar aplicaciones Java, necesitaremos de un JDK, **Java Development Kit**.
- Para poder ejecutar el código Java, nos hará falta un JRE, **Java Runtime Environment**.

El código java es interpretado por la **JVM** (Java Virtual Machine). Esta genera un código intermedio llamado **bytecode** (híbrido entre lenguaje de alto nivel y ensamblador).

Java

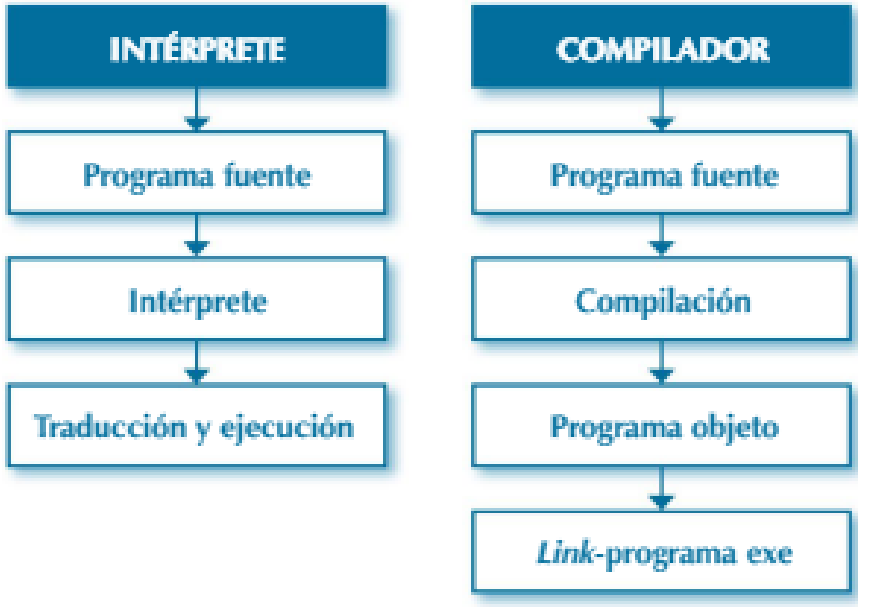

public class HolaMundo {
public static void main(String[] args) {
System.out.println("Hola Mundo");
}
}
HolaMundo.java
javac
Bytecodes
HolaMundo.class
java
Win32
MacOS
Solaris

Codificación

Los **traductores** son programas cuya finalidad es traducir lenguajes de alto nivel a lenguajes de bajo nivel.

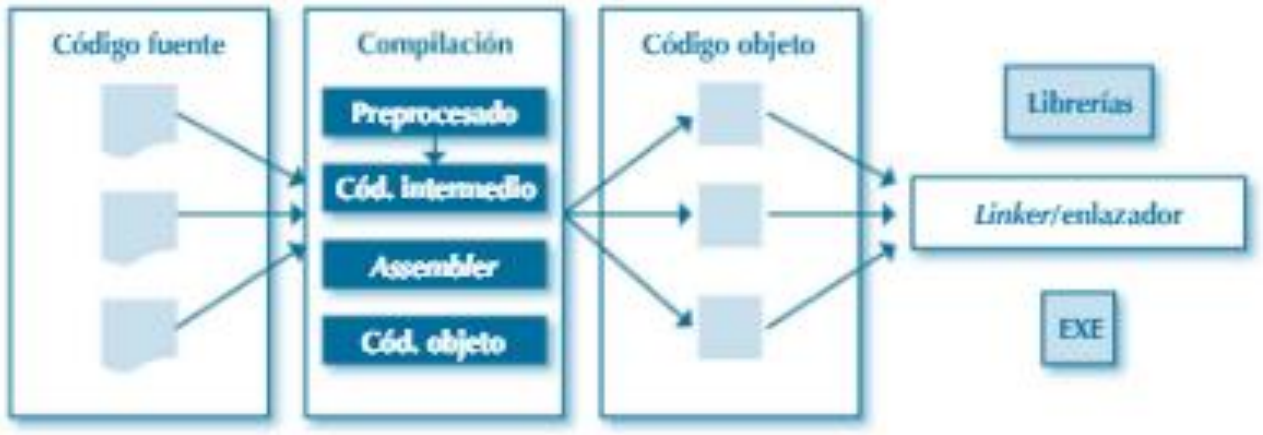
Un **intérprete** traduce el código fuente línea a línea, tiene que estar en memoria para poder realizar este proceso.

Un **compilador** traduce el código fuente a código máquina. El compilador está en la máquina de desarrollo. El código generado solo funcionará en una máquina con un hardware y software determinado. Si se cambian de hardware o software, habría que recompilarlo

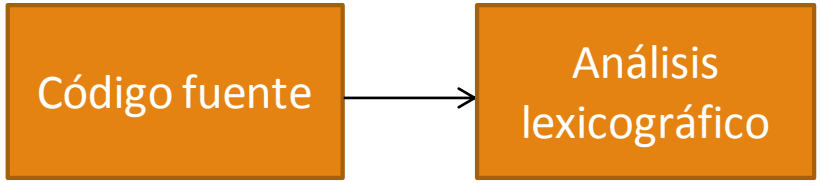


Codificación: Fases de un compilador

1) **Edición**: aquí el programa fuente debe ser tecleado a la computadora a través de un **EDITOR**, de tal manera que se convierta en un archivo de programa almacenado → **PROGRAMA FUENTE (CÓDIGO FUENTE)**
2) **Compilación**: el **COMPILADOR** se encarga de traducir la edición del programa fuente a lenguaje máquina. De ser necesario la compilación se repite hasta no producir errores, dando como resultado el programa objeto → **PROGRAMA OBJETO (CÓDIGO OBJETO)**
3) **Enlace**: el sistema operativo es instruido a tomar el programa objeto y ligarlo con las librerías del programa compilador, dando como resultado un programa ejecutable → **PROGRAMA EJECUTABLE (CÓDIGO EJECUTABLE)**
4) **Ejecución**: Una vez que contamos con el ejecutable, este puede "correrse" en el sistema operativo obteniendo por salida los resultados del programa.



Codificación: Compilación

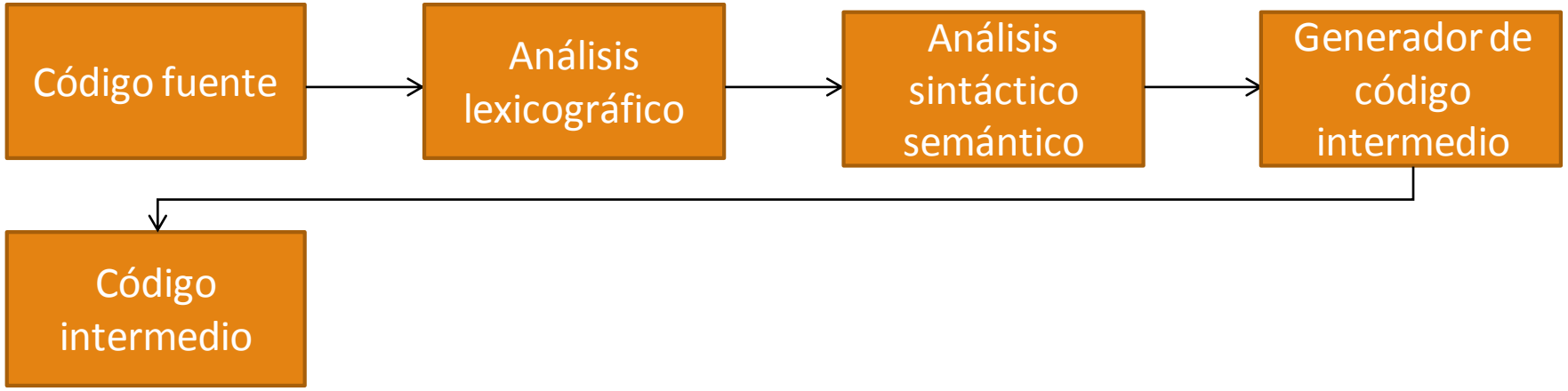


Se busca dentro del código fuente palabras reservadas, operaciones, caracteres de puntuación y se agrupan en cadenas denominadas **lexemas**.

Codificación: Compilación

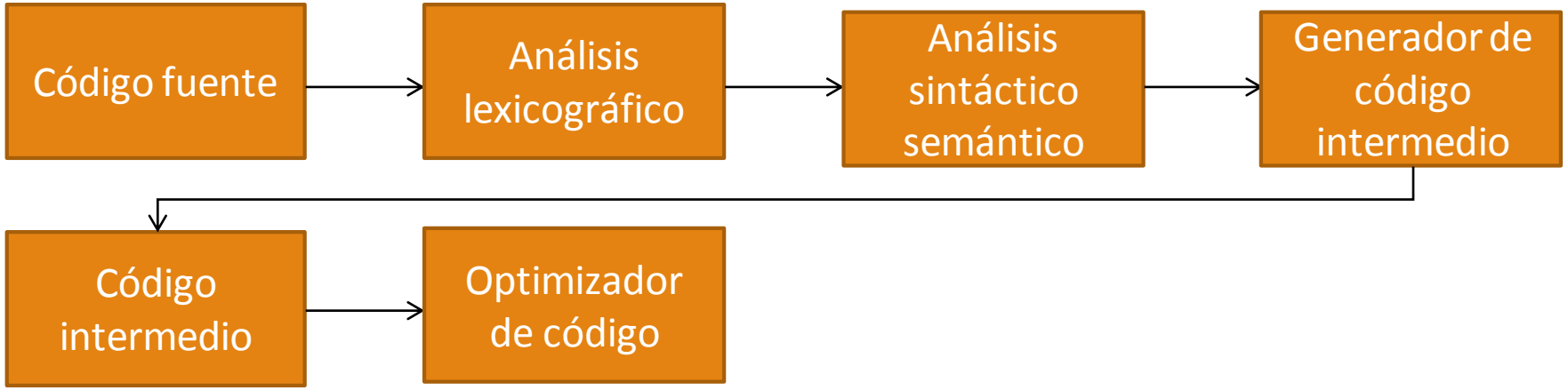
Agrupa los resultados obtenidos en la fase anterior como frases gramaticales. Tras el análisis sintáctico, verifica la coherencia de las frases gramaticales, es decir: si los tipos de datos son correctos, si los *arrays* tienen el tamaño y tipo adecuados, y así consecutivamente con todas las reglas semánticas de nuestro lenguaje.

Codificación: Compilación



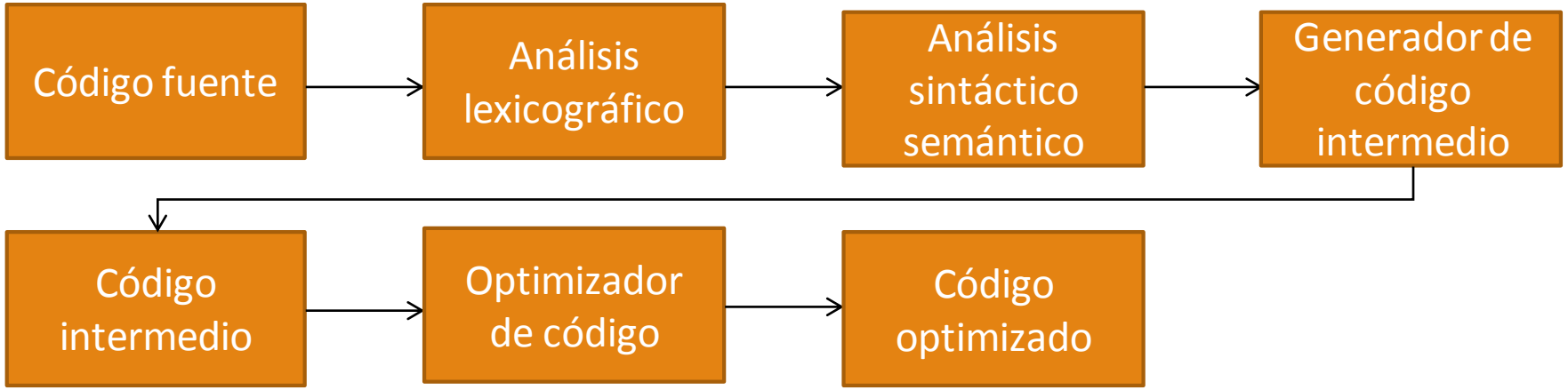
Tras el análisis anterior, se genera una representación intermedia en pseudoensamblador, facilita las tareas de traducción restante.

Codificación: Compilación



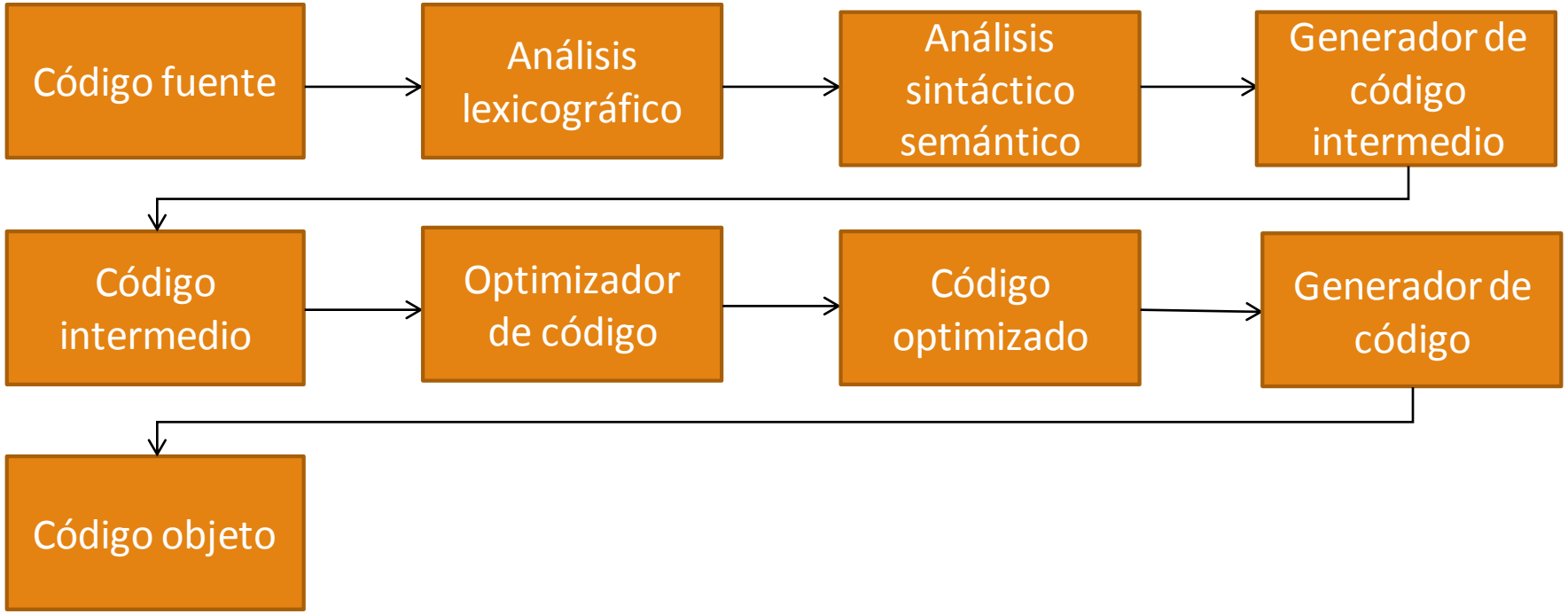
Optimiza el código anterior para que sea más eficiente en la máquina. Ejemplo: Crear variables de intercambio de valores en decisiones.

Codificación: Compilación



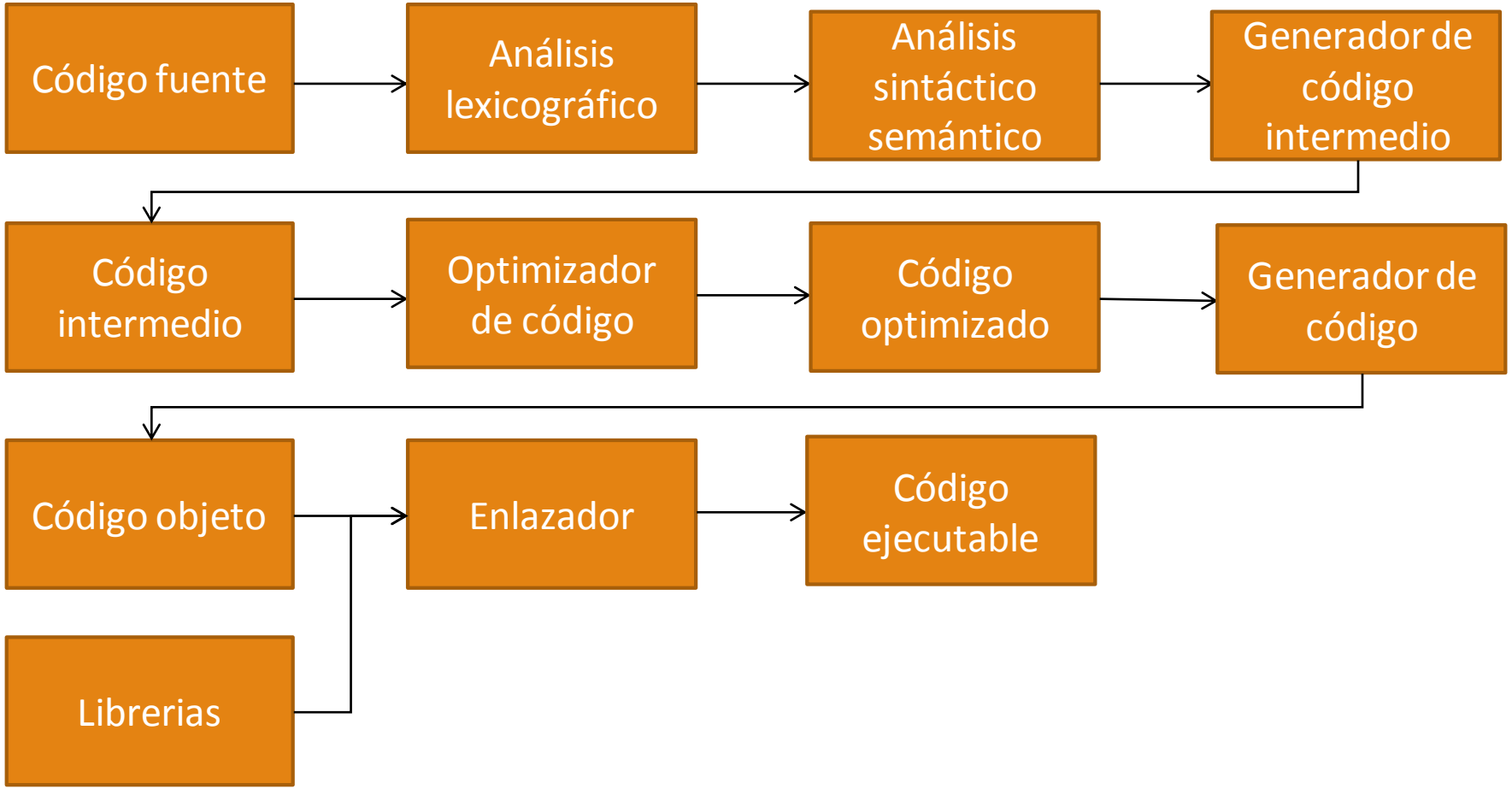
Tras esta optimización de los recursos, genera un código optimizado.

Codificación: Compilación



Genera el código objeto con posiciones de memoria sin establecer, en otras palabras, no sabemos en que zona de la memoria se ejecutará nuestro programa.

Codificación: Compilación



Codificación: Depuración

Aunque un programa ejecutable en primera instancia no marque errores, es fundamental aplicarle una serie de **pruebas** (otra fase, en la que: Es el proceso de ejecución del programa con una amplia variedad de datos que determinan si el programa tiene errores) con la finalidad de cerciorar su funcionamiento y evitar **errores en su operación** (**ERRORES DE EJECUCIÓN**).

En cuanto a las pruebas a llevar a cabo, está la de ingresar valores extremos de entrada con la intención llevar al limite al programa y validar su comportamiento, el realizar las operaciones manualmente y compararlas con los resultados del programa, otra forma es ingresar datos erróneos que determinen su fiabilidad, es decir, su capacidad para recuperarse o, en su caso, el nivel de control frente al uso inadecuado del programa.

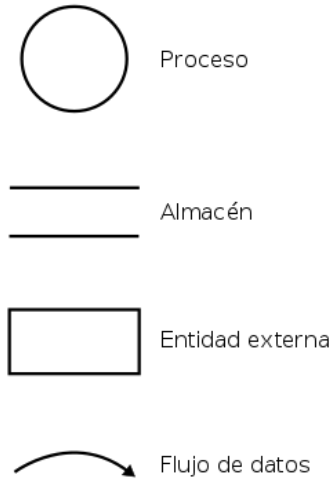
Los posibles errores que no detecta la compilación del sistema son los de **ejecución y lógicos**. En cuanto a los primeros se dice que la computadora los puede "entender", pero no ejecutar, tales como división entre cero, la raíz cuadrada de un número negativo, exceder un rango de valores no permitidos, realizar una operación con un dato leído como carácter en vez de numérico, etc. Por otra parte, los errores lógicos se dice que son más difíciles de corregir, ya que generalmente son producto del mal análisis y diseño del problema.

Técnicas de desarrollo gráfico

- **Diagrama de Contexto: Nivel 0**
  - ✓ Representan todas las interacciones que realiza un sistema con su entorno.
  - ✓ Se dibuja un sólo proceso que representa al sistema en cuestión y se escribe su nombre en dicha burbuja.
  - ✓ De él solamente parten los flujos de datos que denotan las interrelaciones entre el sistema y sus agentes externos, no admitiéndose otros procesos ni almacenamientos. Esto es debido a que son procesos estructurados y ordenados que poseen una cardinalidad.

- **Diagrama de Nivel Superior: Nivel 1**
  - ✓ Se plasman todos los procesos que describen al proceso principal.

- **Diagrama de Detalle o Expansión: Nivel 2**
  - ✓ En estos se explotaran las excepciones a los caminos principales de la información dado que aumenta progresivamente el nivel de detalle.



Técnicas de desarrollo gráfico

Diagramas de flujo

Nombre	Símbolo	Función
Inicio/Final		Se utiliza para representa el inicio o fin de un proceso o programa
Entrada/Salida		Se utiliza para representar la introducción de datos por medio de periféricos.
Proceso		Se utiliza para representar cualquier tipo de operación que pueda originar cambio de valor, formato o posición de la información almacenada en memoria, operaciones aritméticas, de transformaciones, etc.
Decisión		Se utiliza para indicar operaciones lógicas o de comparación entre datos.
Documento		Se utiliza para representar la salida de datos por impresora, pero en ocasiones es usado para mostrar datos o resultados.
Desplegar/Mostrar		Este es utilizado para representar la salida o para mostrar la información por medio del monitor o la pantalla.
Proceso predefinido		Se utiliza para representar procesos ya definidos tales como llamada a procedimientos o funciones y el inicio del mismo.
Base de datos		Se utiliza para representar la escritura o almacenado de datos en la base de datos.
Almacenamiento de datos		Se utiliza para representar la escritura o almacenado de datos en disco o en línea.
Unir		Se utiliza para acoplar segmentos del diagrama o para recibir la línea de flujo.
Multi-documento		Se utiliza para representar la salida, despliegue o impresión de varios documentos.
Entrada manual		Representa la intervención de usuario para dar una entrada a datos requeridos (No se confunda con el símbolo de Entrada / Salida).
Operación manual		Representa la intervención del usuario para realizar un proceso manual.
Almacenamiento interno		Se utiliza para representar el almacenamiento en memoria de algún proceso o valor.
Cinta magnética		Representa datos grabados en una cinta magnética.
Límite de ciclo		
Preparación		Expresa proceso de llamada a un proceso subalterno.
Tarjeta		Representa la entrada de datos o lectura de datos de una tarjeta perforada o recientemente de memorias de almacenamiento.
Retraso		Representa la atraso para poder iniciar el siguiente proceso o tarea.
Conector (dentro de página)		Sirve para enlazar dos partes cualesquiera de un diagrama a través de un conector en la salida y otro conector en la entrada. Se refiere a la conexión en la misma página del diagrama
Conector (fuera de página)		Sirve para enlazar dos partes cualesquiera de un diagrama a través de un conector en la salida y otro conector en la entrada. Se refiere a la conexión en distinta página del diagrama
Línea de flujo		Indica el sentido de la ejecución de las operaciones

Pseudocódigo

```
Si condición Entonces
    instrucciones1;
Si no Entonces
    instrucciones2;
Fin Si
```

```
Si condición1 Entonces
    instrucciones1;
Si no si condición2 Entonces
    instrucciones2;
Si no si condición3 Entonces
    instrucciones3;
...
Si no Entonces
    instrucciones_n;
Fin Si
```

```
Según variable Hacer
    caso valor1;
        instrucciones1;
    caso valor2;
        instrucciones2;
    caso valor3;
        instrucciones3;
    ...
    De Otro Modo
        instrucciones_n;
Fin Según
```

Técnicas de desarrollo gráfico

1) Programa que lee dos números y muestra la suma en pantalla.
2) Programa que lee dos números y muestra el mayor en pantalla, si son iguales deberá mostrar un mensaje indicándolo.
3) Programa que lee dos números en un proceso repetitivo. Este proceso terminará cuando los números leídos sean iguales.
4) Programa que lee diez números en un proceso repetitivo y muestra su suma.
5) Programa que lee un número por teclado y muestra a qué día de la semana corresponde.
6) Programa que lee una nota (0-10) por teclado y muestre el siguiente literal:
    - 5 => Suficiente
    - 6 => Bien
    - 7, 8 => Notable
    - 9,10 => Sobresaliente
7) Programa que lee dos números por teclado y muestre el menor de ellos.
8) Programa que lee un número por teclado
9) Programa que lee 10 números y visualiza cuántos son positivos, cuántos son negativos y cuántos son nulos.