Introducción a la Programación.

Andrés Arcia
Departamento de Computación
Escuela de Ingeniería de Sistemas
Facultad de Ingeniería
Universidad de Los Andes

Introducción a la Programación

Algunas definiciones importantes:

¿Qué es Programar?

Es la acción de escribir instrucciones correctas para que sean interpretadas por una máquina.

¿Qué es el Software?

Son programas. Para que tengan sentido de software deben ser ejecutados sobre una máquina.

¿En qué medida nos compete programar?

Depende de su interes. De todas formas hoy en día es un “must” para cualquier ingeniero.

¿En que medida dependemos de software?

Depende de su estilo de vida, pero para el ciudadano común la dependencia es bastante: celulares, controles remotos (TV, DVD, radios, mp3 players, etc.), cajeros automaticos, etc. ¿Sabia Ud. Que muy pronto será espiado sin darse cuenta?

Lenguajes de Programación

- ¿Qué es un Lenguaje de Programación?

  Es un conjunto de reglas para comunicar ideas. Generalmente las ideas se le comunican a una máquina.

- De que hay que estar pendiente cuando programamos en un lenguaje:

  Sintaxis / Semantica

  Sistema de tipos

  Errores / Excepciones

Lenguajes de Programación

- Paradigmas de la Programación
  - Programación Imperativa
  - Programación Orientada por Objeto
  - Programación Funcional
  - Programación por Eventos
  - Programación Concurrente
  - etc.
- Lenguajes de programación populares:
  - C, C++, Java, PHP, Perl, XHTML.
- Dominios de aplicación importantes:
  - Programación Sistema
  - Sistemas de Gestión de Información
  - Programación Web

Niveles de los lenguajes

Lenguaje Natural

Lenguaje de Programación

Compilador / Interprete

Lenguaje Máquina

¿Qué aprenderemos en el curso?

Aprenderemos a programar en lenguaje **C**.

- C fue diseñado para dar soporte a UNIX a mediados de los 70. Hoy en dia lo encontramos en los mas grandes OS: Linux y FreeBSD.
- C esta normalizado por ANSI (1988).

Enlaces a revisar

yahoo.com: Tutorial de C, curso de C

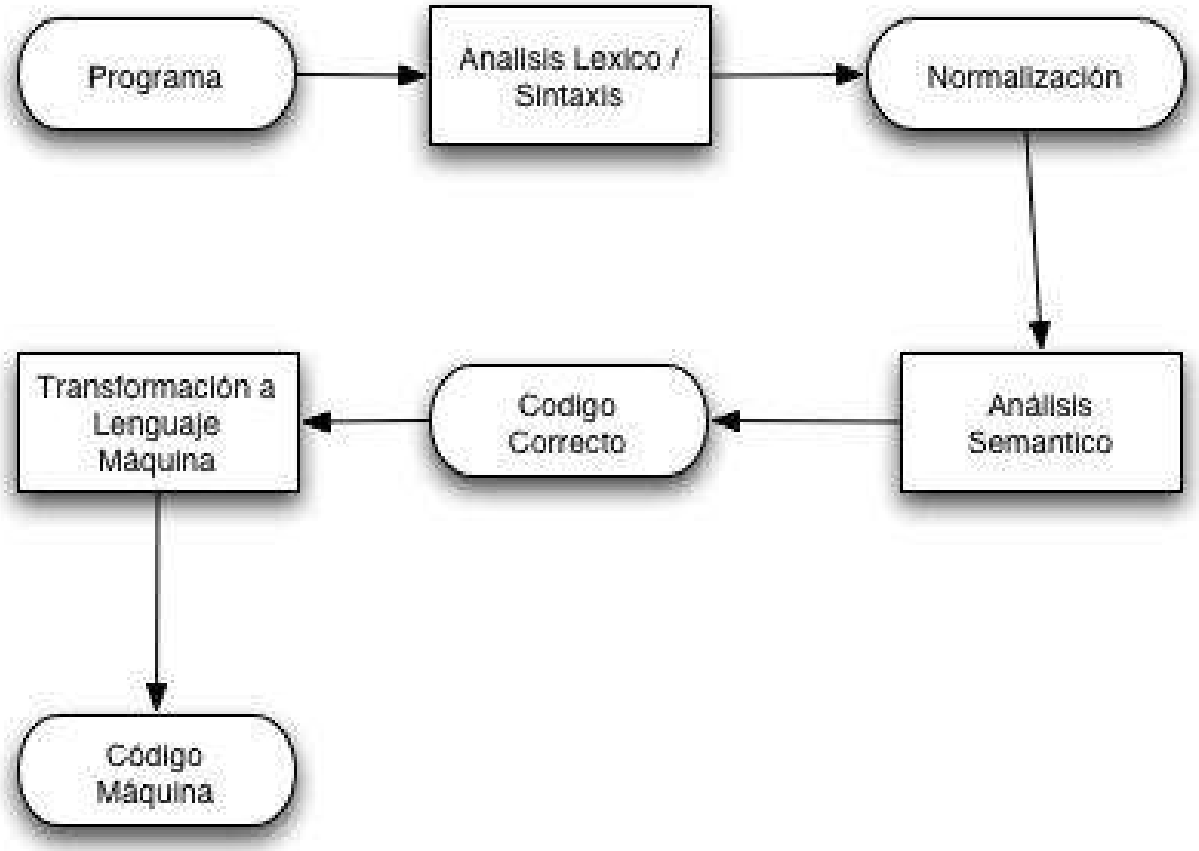
google.com: Tutorial de C, curso gratis de C, etc.

www.elrincondelc.com

www.emagister.com

gcc.gnu.org

Etapas de la Compilación



Lenguaje de Máquina

- Lenguaje mas básico, propio de cada computadora, ya que está relacionado con el diseño del hardware de la misma (dependiente de la máquina). Por lo general consisten en cadenas de números al final reducidos a ceros y unos (sistema numérico binario).

- Operaciones:
  - Cargar
  - Almacenar
  - Sumar
  - Restar

Lenguaje de Máquina

Ejemplo:

Código de operación	Dirección	Significado
00010101	10000001	(a) Cargar contenido de la dir. 129 en Acumulador
00010111	10000010	(b) Sumar contenido de la dir 130 al Acumulador
00010110	10000011	(c) Almacenar contenido del Acumulador en la dir. 131

(c) $10000011 = 2^7+2^1+2^0 = 131$.

Lenguaje Ensamblador

- Consiste en abreviaturas similares al inglés, llamadas instrucciones mnemotécnicas, que permiten representar las operaciones elementales de la computadora (dependiente de la máquina).

  Ejemplo:

Código de operación	Dirección	Instrucción en lenguaje ensamblador
00010101	10000001	LOAD A
00010111	10000010	ADD B
00010110	10000011	STORE C

Lenguaje Ensamblador

- *Lenguaje de bajo nivel o ensamblador*:

  La computadora no entiende directamente lenguaje ensamblador por lo que un programa escrito en este lenguaje tiene que ser traducido a lenguaje de máquina por un programa llamado un *ensamblador* para que pueda ser ejecutado por la computadora.

  Los lenguajes ensambladores todavía requieren que el programador tenga un buen conocimiento de la arquitectura de la computadora.

  Como los lenguajes ensambladores son dependientes de la máquina, todo programa escrito en un lenguaje ensamblador particular tendrá que ser reescrito si se va a ejecutar en otro tipo de computadora.

Lenguaje de Alto Nivel

- Permite a los programadores escribir instrucciones en un lenguaje mas familiar para ellos y que contiene notaciones matemáticas comúnmente utilizadas (independiente de la máquina).

  Ejemplo:

Código de operación	Dirección	Instrucción en lenguaje ensamblador	Instrucción en lenguaje de alto nivel
00010101	10000001	LOAD A	
00010111	10000010	ADD B	
00010110	10000011	STORE C	C = A + B

  Con este tipo de lenguajes, la programación es mas fácil para los usuarios ya que éste no necesita tener conocimiento de la arquitectura de la computadora.

Lenguaje de Alto Nivel

Lenguaje de alto nivel:

Como ocurre con los lenguajes ensambladores, la computadora no entiende directamente lenguaje de alto nivel, por lo que un programa escrito en este lenguaje tiene que ser traducido a lenguaje de máquina por un programa llamado un *compilador* para que pueda ser ejecutado por la computadora.

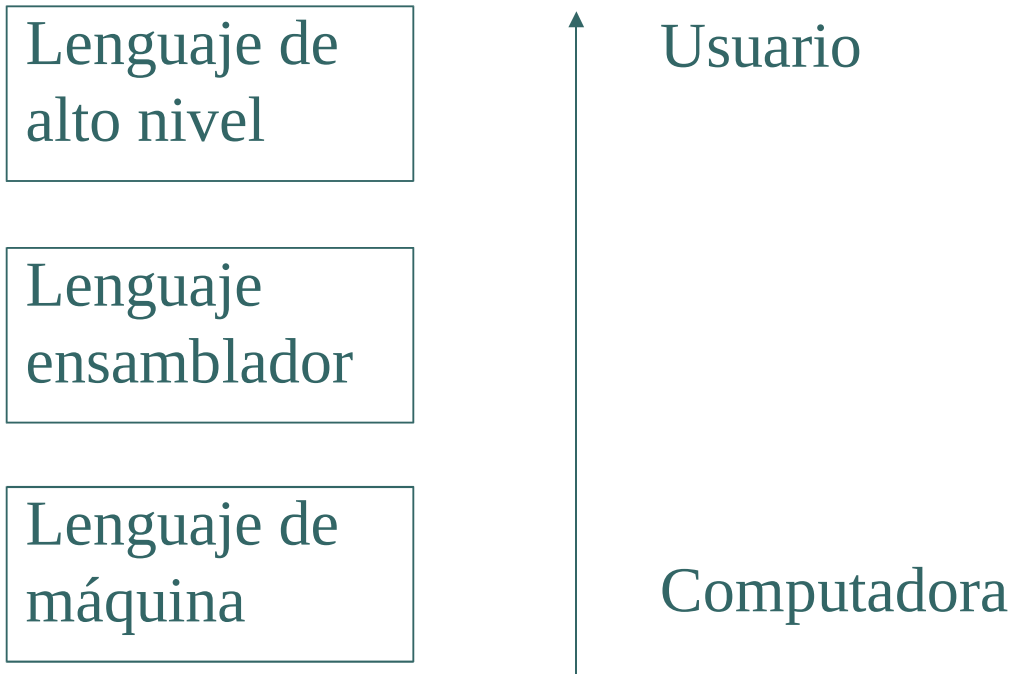
Los lenguajes de alto nivel permiten portabilidad, mejor expresión de las ideas, facilidad de programar ciertas clases de problemas, menos posibilidad de cometer errores, una visión más amplia del problema, etc.

Lenguaje de Alto Nivel

Ejemplos de lenguajes de alto nivel:

- Java
- C
- C++
- COBOL
- FORTRAN
- PROLOG
- LISP
- PL/I
- SMALLTALK
- ADA
- BASIC
- Visual Basic
- Pascal

Lenguajes de Programación



Lenguajes de Programación

Importante

Cada CPU tiene su propio lenguaje de máquina interno. La programación a este nivel se realiza generalmente en el lenguaje ensamblador específico de la computadora. Cada instrucción en lenguaje ensamblador corresponde a una instrucción en lenguaje de máquina.

Si existe una estandarización para un lenguaje de alto nivel, cualquier programa escrito usando este estándar debe poder ejecutarse en cualquier computadora después de compilarlo. Esto se le conoce como *portabilidad* de programas.

Lenguajes de Programación

Elementos de un lenguaje de programación

Un sublenguaje para definir los datos
- Qué datos tenemos
- Cómo les llamamos
- Cómo son (tipo y/o estructura)
- Qué se puede hacer con ellos

Un sublenguaje para definir los algoritmos
- Qué le hacemos a los datos
- En qué orden (cuándo se lo hacemos)
- Cuántas veces

Metodología de Desarrollo de Programas

El desarrollo de programas sigue hoy en dia distintas metodologías: De arriba hacia abajo, espiral, modular, etc.

En este curso Usted aprendera que lo más importante es comprender el problema cabalmente. Luego Usted podrá encasillarlo en cualquiera de las técnicas existentes.

Para la comprensión de un problema se requiere que Usted este alerta con todos los sentidos.

El proceso del pensamiento y abstracción del problema NO TIENE METODOLOGIA ESPECIFICA.

“NO” CREA EN CUENTOS DE CAMINO. Por ejemplo entrada, proceso y salida. Es posible que funcione pero dese cuenta que lo está castrando!!!

Cuando Usted logre conseguir la explicación más sencilla y la analogia correcta a un problema entonces ya lo habrá resuelto. Ejercicio: Explique la suma de números enteros, la resta, la multiplicación y la división.

Metodología de Desarrollo de Programas

Diseño del algoritmo: Descripción de una secuencia finita y ordenada de pasos – sin ambigüedades – que conducen a la solución de un problema dado.

Herramientas de diseño

- Diagramas de flujo (para la programación estructurada)
- Círculos y canales de mensaje (programación orientada a objetos)
- Pseudocódigo
- Trazas personales
- Grafismos
- Formulas matemáticas
- Todo aquello que le ayude a representar el problema.

Metodología de Desarrollo de Programas

Codificación: Traducción del algoritmo a un programa escrito en un lenguaje de programación adecuado (código fuente).

Corrida en frío del programa: Prueba manual de la correctitud del programa.

Depuración del programa: Identificación y eliminación de errores.

- Errores de sintaxis: Violan las reglas del lenguaje de programación. Un buen compilador localizará e identificará la mayoría de estos automáticamente.
- Errores lógicos: Equivocaciones que causan que el programa se ejecute de forma inesperada o incorrecta.

Ejecución del programa: Ejecución del código ejecutable (código en lenguaje de máquina) del programa bajo el control del CPU, una instrucción a la vez.

Metodología de Desarrollo de Programas

Puesta en operación: Instalación del hardware y software, capacitación, etc..

Mantenimiento del programa: Comienza tan pronto como el producto es lanzado. Permite corregir defectos menores, añadir una mayor funcionalidad, ya sea en respuesta a las demandas del mercado o a las peticiones del usuario.

Metodología de Desarrollo de Programas

Análisis E-P-S

- *Especificaciones de entrada:* Información necesaria para la solución del problema.
  - ¿ Qué datos son de entrada ?
  - ¿ Cuántos datos se introducirán ?
  - ¿ Cuáles datos de entrada son válidos ?

- *Proceso:* Operaciones o cálculos necesarios para encontrar la solución del problema.
  - ¿ Qué tipo de ecuaciones ?
  - ¿ Cuántas ecuaciones ?
  - ¿ Qué transformaciones sobre la data?

Metodología de Desarrollo de Programas

Análisis E-P-S

- *Especificaciones de salida:* Resultados finales de los cálculos.
  - ¿ Cuáles son los datos de salida
  - ¿ Cuántos datos de salida se producirán
  - ¿ Qué precisión tendrán los resultados
  - ¿ Se debe imprimir un encabezado

Metodología de Desarrollo de Programas

Diseño del algoritmo

Un algoritmo debe ser *preciso* e indicar el orden de realización de cada paso.

Un algoritmo debe ser *finito*. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos.

Ejemplo 1: Supóngase que tiene un examen de PDI. Según una decisión aleatoria, algunos estudiantes estudian antes del examen y otros se van a ver un espectáculo. Realizar el análisis E-P-S y diseñar un algoritmo para representar esta situación.

Análisis E-P-S

Entrada: una moneda para hacer la decisión aleatoria.

Proceso: lanzar la moneda y luego tomar la decisión.

Salida: resultado en el examen.

Diseño del algoritmo

Algoritmo estudiarONoEstudiar

0. Inicio
1. Lanzar una moneda.
2. Si el resultado es “cara” ir a 5.
3. Si el resultado es sello estudiar para el examen.
4. Ir a 6.
5. Ver un espectáculo.
6. Presentar el examen al día siguiente.
7. Fin

Ejemplo 2: Realizar el análisis E-P-S y diseñar un algoritmo para calcular el área de superficie de un paralelepípedo de dimensiones l (largo), a (ancho) y h (altura)

Análisis E-P-S

Entradas: tres números reales l (largo), a (ancho) y h (altura)

Proceso:

Calcular el área del paralelepípedo

$$AS = 2 \times (l \times a + l \times h + a \times h)$$

Salidas: número real que representa el área (AS)

Diseño del algoritmo

Algoritmo Area

0. Inicio
1. Escribir("Largo del paralelepipedo = ")
2. Leer el valor de l
3. Escribir("Ancho del paralelepipedo = ")
4. Leer el valor de a
5. Escribir("Altura del paralelepipedo = ")
6. Leer el valor de h
7. AS = 2 * (l * a + l * h + a * h)
8. Escribir("Area del paralelepipedo =", AS)
9. Fin

Metodología de Desarrollo de Programas

Codificación

Traducir el algoritmo producido en el paso anterior en un programa escrito en un lenguaje de programación de alto nivel (programa fuente o código fuente). En nuestro caso C++.

Los diferentes pasos de un algoritmo se expresan en los programas como instrucciones (término usado para los lenguajes de máquina y bajo nivel), sentencias o proposiciones (términos usados para los lenguajes de alto nivel).

Metodología de Desarrollo de Programas

Codificación

Programa: Secuencia de sentencias, cada una de las cuales especifica ciertas operaciones que debe ejecutar la computadora.

Tipos básicos de sentencias:

- Sentencias de entrada/salida
- Sentencias aritmético-lógicas
- Sentencias de decisión o selectivas
- Sentencias repetitivas o lazos

Metodología de Desarrollo de Programas

Codificación: *Tipos básicos de sentencias*

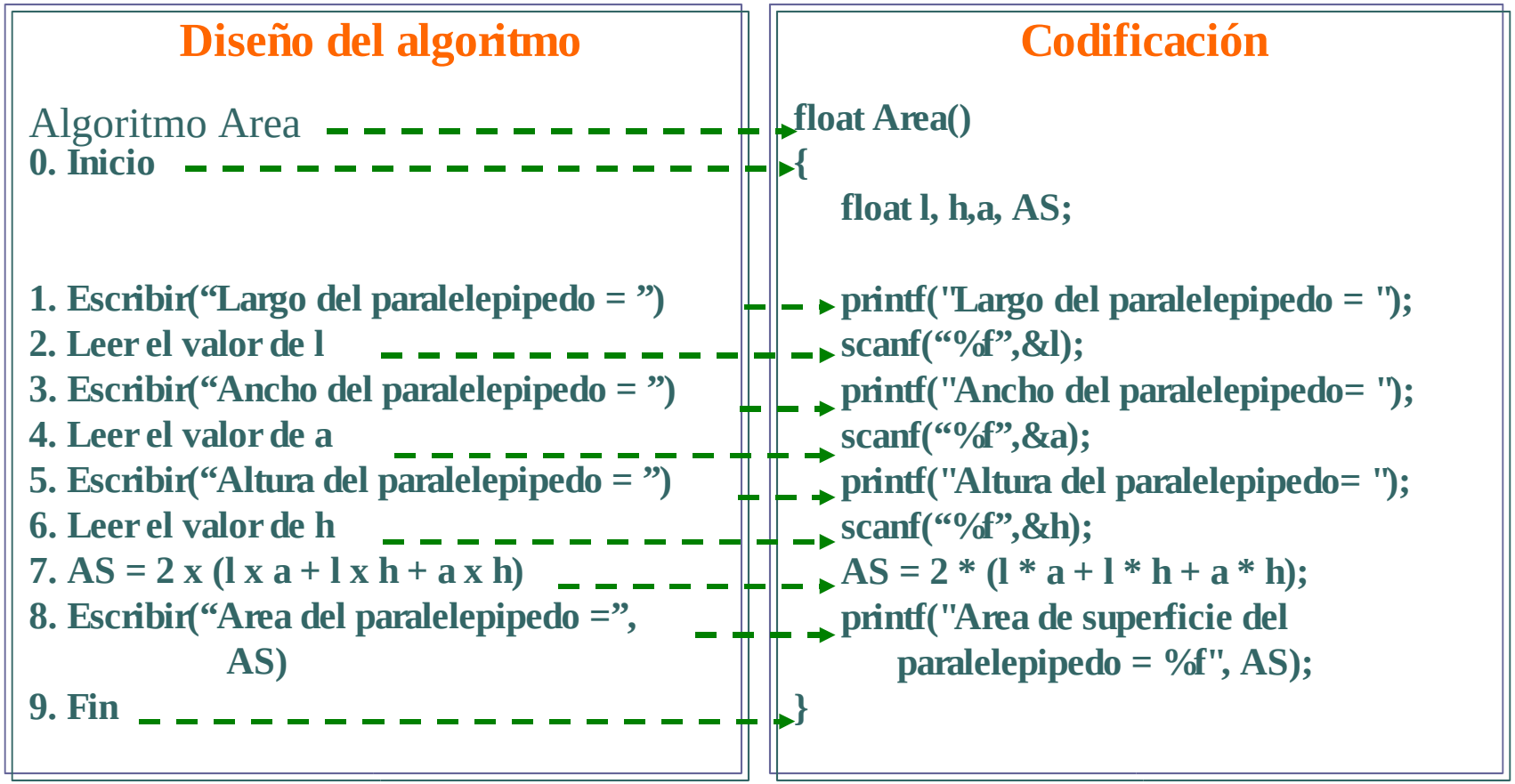
Sentencias de entrada/salida: Sentencias de transferencia de información y datos entre dispositivos de E/S (teclado, impresora, discos, etc.) y la memoria principal.

Sentencias aritmético-lógicas: Sentencias que ejecutan operaciones aritméticas (suma, resta, multiplicación, etc.) o lógicas (y lógico, o lógico, negación).

Sentencias de decisión o selectivas: Sentencias que permiten la selección de tareas alternativas en función de los resultados de diferentes expresiones condicionales.

Sentencias repetitivas o lazos: Sentencias que permiten la repetición de secuencias de sentencias un número determinado o indeterminado de veces.

Ejemplo 2: Realizar el análisis E-P-S y diseñar un algoritmo para calcular el área de superficie de un paralelepípedo de dimensiones l (largo), a (ancho) y h (altura).



Metodología de Desarrollo de Programas

Corrida en frío del programa

El programador realiza una corrida en frío sobre el programa fuente escogiendo un conjunto de datos de entrada, ejecutando manualmente cada sentencia del programa fuente y verificando que los resultados obtenidos son los esperados de acuerdo al conjunto de datos de entrada.

Como una técnica de depuración, el programador debe realizar este proceso utilizando conjuntos de datos que permitan ejecutar todos los “caminos” posibles del programa.

Ejemplo 2: Realizar el análisis E-P-S y diseñar un algoritmo para calcular el área de superficie de un paralelepípedo de dimensiones l (largo), a (ancho) y h (altura)

Codificación

```
float Area()
{
    float l, h,a, AS;

    printf("Largo del paralelepipedo = ");
    scanf("%f",&l);
    printf("Ancho del paralelepipedo= ");
    scanf("%f",&a);
    printf("Altura del paralelepipedo= ");
    scanf("%f",&h);
    AS = 2 * (l * a + l * h + a * h);
    printf("Area de superficie del
        paralelepipedo = %f", AS);
}
```

Corrida en frío

l = 3.0
a = 2.5
h = 7.3
AS = 2 x (3.0 x 2.5 + 3.0 x 7.3 +
2.5 x 7.3) = 95.3

Metodología de Desarrollo de Programas

Ejecución del programa

El programa fuente es introducido a la computadora utilizando un programa llamado *editor*.

Una vez editado, el programa fuente es traducido por el *compilador* a un programa escrito en lenguaje de máquina (código objeto), siempre y cuando el programa no tenga *errores de sintaxis* (errores de gramática).

Ejemplo

```
if (a < b          // Falta un paréntesis que cierra
```

Metodología de Desarrollo de Programas

Ejecución del programa

Si el programa fuente tiene errores de sintaxis no se genera código objeto. Los errores deben ser corregidos usando el editor y luego se el programa fuente se debe volver a compilar.

Cuando el programa está sintácticamente correcto, el código objeto es encadenado con las funciones de librería (otros programas) requeridas usando un programa llamado *encadenador*.

Metodología de Desarrollo de Programas

Ejecución del programa

El código objeto compilado y encadenado es cargado en la memoria principal para su ejecución por un programa llamado *cargador*.

El código objeto compilado, enlazado y cargado (código ejecutable) es *ejecutado* con los datos de entrada.

Metodología de Desarrollo de Programas

Comprobación del programa

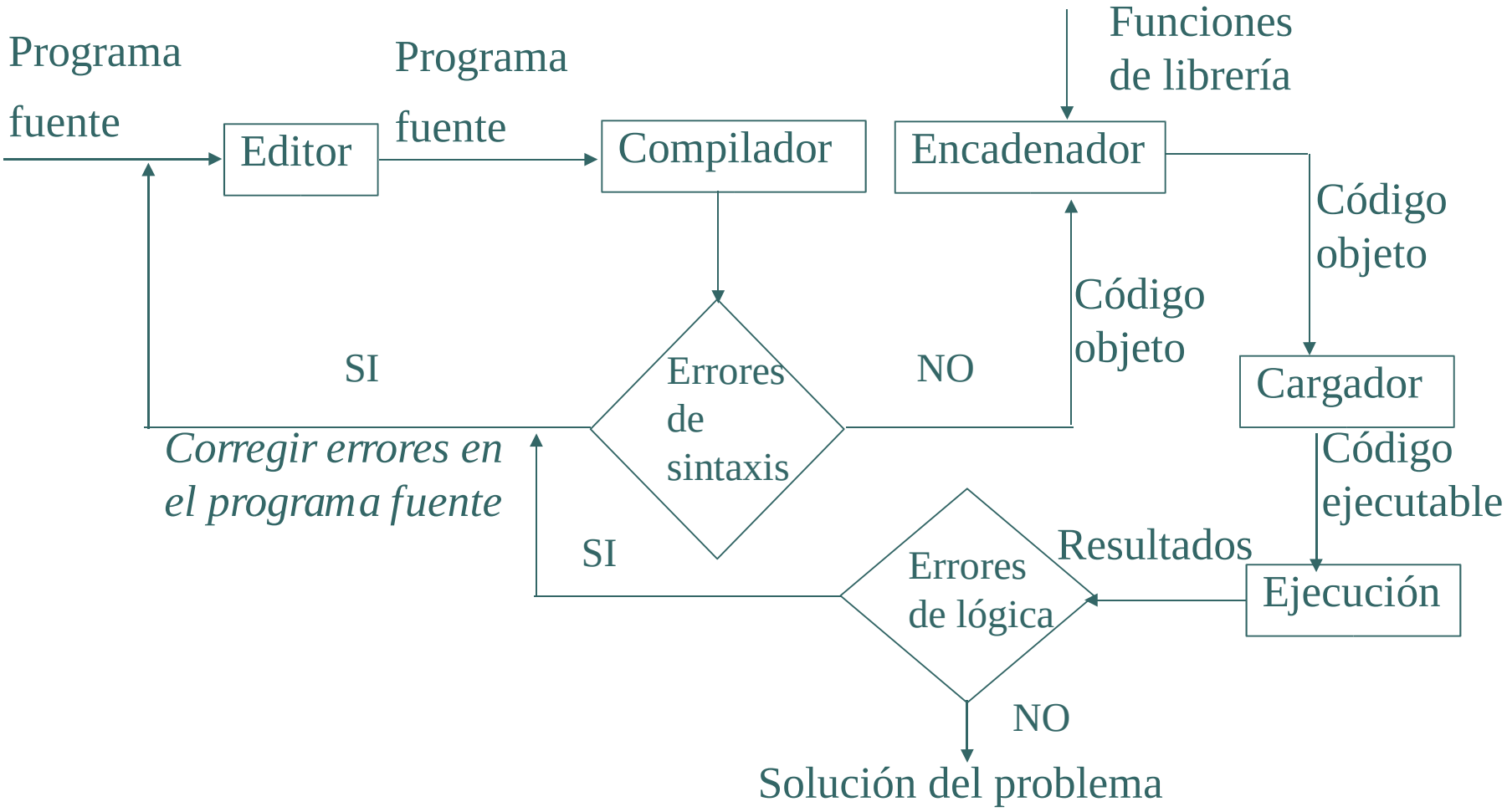
Comprobar que el programa realiza las tareas para las cuales ha sido diseñado y produce el resultado correcto y esperado.

Si el programa tiene *errores de lógica* (errores en el método de solución por lo que las salidas obtenidas no corresponden con las salidas esperadas), éstos deben ser corregidos en el programa fuente usando el editor, el cual se debe volver a compilar.

Ejemplo

```
b = 0;
c = 5/b;          // División entre cero
```

Metodología de Desarrollo de Programas



Ejercicios

Suponga que dispone de un Robot que sigue instrucciones muy simples: mover <dirección> <# de pasos>, tomar objeto, dejar objeto.

Proponga como trasladar un objeto desde A hasta B dando instrucciones al Robot.

Qué suposiciones esta haciendo acerca del robot: ¿Tiene visión? ¿Escucha?

Ejercicios

Busque en diferentes manuales de vehiculos cómo cambiar un caucho.

Busque una receta de cocina.

Busque las instrucciones de como armar algún juguete para niños.

Preguntese acerca del nivel de detalle en cada caso. Hace falta mas o menos detalle. Critique.

Ejercicios

Realizar el análisis E-P-S y diseñar un algoritmo para resolver los siguientes problemas:

Cambiar el caucho a un carro.

Comprar una entrada para un concierto.

Saber si un número n es primo o no. Correr el algoritmo en frío para los valores de n = 131, n = 28, n = 7 y n = 1024.

Calcular la circunferencia y superficie de un círculo.

Ejercicios

Un corredor de maratón recorre una distancia *d* en *h* horas y m minutos. Calcular el tiempo medio en minutos por kilómetros. Correr el programa en frío para los valores d = 42,195 Km., h = 2 y m = 25.

Encontrar la media de una lista indeterminada de números positivos terminada con un número negativo. Correr el programa en frío para la lista 2, 4, 5, 6, 23, 54, 12, 65, -1.

Resolución de Ejercicios

Cambiar el caucho a un carro.

Algoritmo pinchazo
0. Inicio
1. Si el gato del carro está dañado
    Llamar a un amigo
    Ir a 2
  sino
    Sacar el gato
    Poner el gato
    Levantar el coche con el gato
    Aflojar y sacar los tornillos del caucho
    Quitar el caucho
    Poner el caucho de repuesto
    Poner los tornillos y apretarlos
    Bajar el gato
    Guardar el gato
  fin_si
2. Fin