RESUMEN DE CONTENIDOS N° 5

Introducción a la Programación

Revisión de Conceptos

Se estudió en la Unidad 1 el proceso de resolución de problemas computacionales, donde se distinguen las etapas siguientes.

Definición del problema.

Análisis del problema.

Elección del método

Codificación.

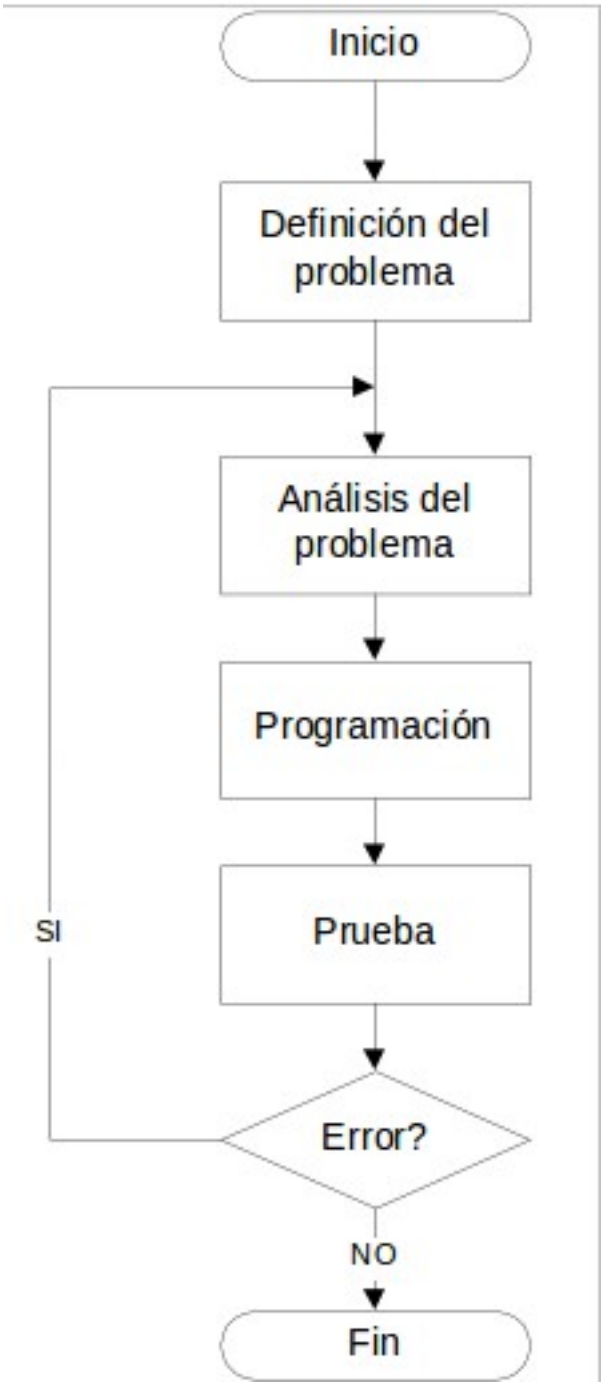
Prueba.

Depuración.

Documentación.

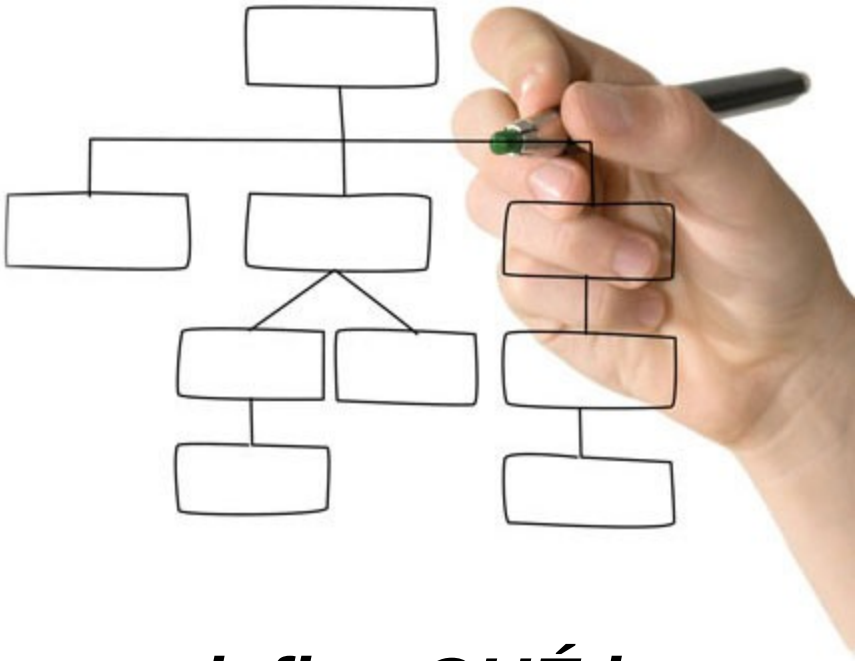
Las etapas correspondientes a la Codificación, Prueba y Depuración constituyen el proceso de **Programación**, que se desarrollará a partir de aquí en la asignatura.

FUNDAMENTOS DE PROGRAMACIÓN



Estrategia - Método Top Down

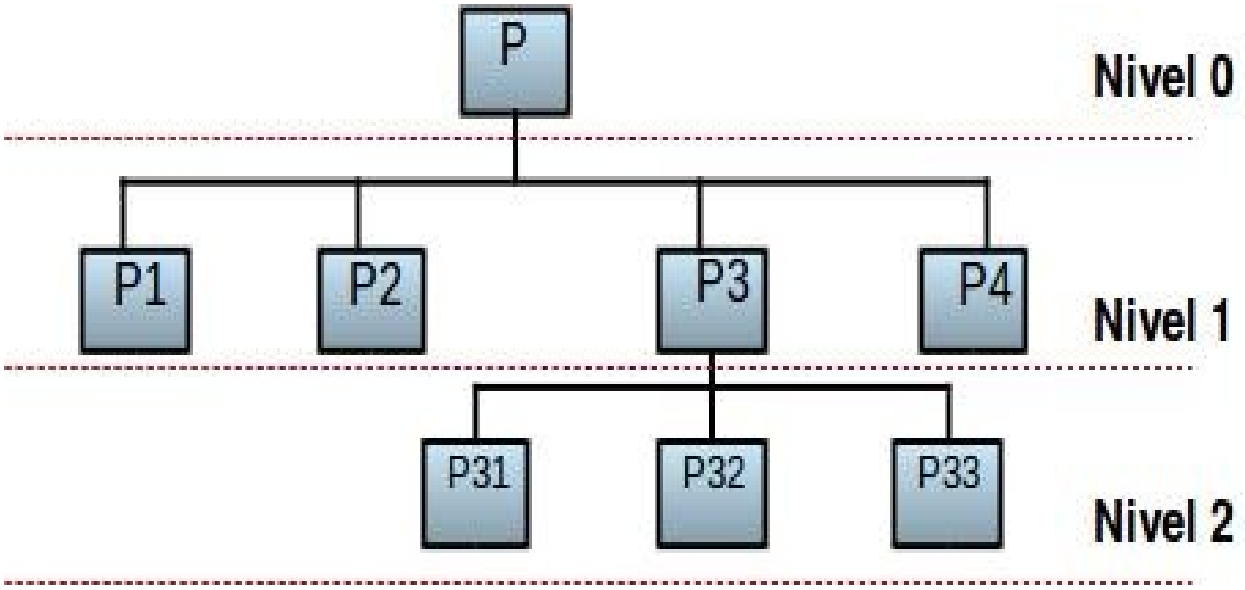
Diseñar una estrategia, consiste en dividir o descomponer el problema original en una sucesión de problemas más simples, de tamaño suficientemente pequeño como para que cada uno de ellos pueda ser comprendido en su totalidad. Esta técnica es conocida como Top-Down o de refinamientos sucesivos y, como se verá más adelante, se adapta perfectamente a la codificación de programas mediante un lenguaje modular y estructurado



La estrategia nos define QUÉ hacer

Estrategia - Método Top Down

El diseño de la estrategia consiste en encontrar un método que nos permita llegar a resolver el problema planteado. Como primer paso de esta etapa, **debemos preparar un plan o esquema general de las tareas que deben realizarse** para llegar a la solución. Este esquema se denomina estrategia y debe ser una lista de **QUÉ** hacer.



ESTRATEGIAS

En la unidad 1, definimos ESTRATEGIA y ubicamos a la **Definición de una estrategia** como una etapa dentro de la **Resolución de problemas**, que debe realizarse previa a la **Definición de algoritmos**. Comenzaremos a aplicar estos conceptos y los vistos en esa unidad referidos a diseño descendente, en la resolución de problemas que se nos plantean.

Representa un proceso iterativo. Debajo de él pueden graficarse los módulos que forman el ciclo.

Representa una alternativa cuya ejecución dependerá de una condición.

ESTRATEGIAS

Ejemplo Práctico:

Una compañía paga a cada uno de sus vendedores una bonificación anual basada en el sueldo del vendedor y en su total de ventas durante el año. El valor de la bonificación es del 3 % del total de las ventas si las ventas son inferiores a 4 veces el sueldo del vendedor, o del 5 % si son mayores o iguales a ese monto.

Realizar un algoritmo que ingrese por cada vendedor el nombre y apellido, el sueldo y el total de ventas. El fin de datos se produce al ingresar como nombre y apellido 'ZZZZ'.

Para cada empleado informar el nombre y apellido y la bonificación correspondiente, y además informar el total que debe pagar la compañía en concepto de bonificaciones, con leyenda indicativa.

Algoritmo

En esta etapa se plantea en base a la estrategia, el conjunto de acciones que permitirán resolver el problema, mediante pseudocódigo, diagrama de flujo, etc

El algoritmo define CÓMO hacerlo

Programa

Un algoritmo codificado empleando un lenguaje de programación interpretable por una computadora constituye un programa.

Código Ejecutable

Para poder probar un programa escrito en un lenguaje de programación de Alto Nivel es necesario generar un código ejecutable.
Este proceso puede efectuarse mediante la
COMPILACION

o mediante la
INTERPRETACIÓN

del código fuente que editó el programador.

El Proceso de Compilación

Es una traducción del código fuente a un código ejecutable por la computadora.

El resultado de compilar un archivo o programa fuente es un nuevo archivo llamado imagen ejecutable.

Los archivos ejecutables pueden ser directamente utilizados por el usuario mediante una simple llamada desde el sistema operativo (por ejemplo un doble clic en un entorno gráfico).

El archivo ejecutable ya no requiere del compilador ni del entorno que permitió su creación y puede ser utilizado en cualquier computadora (de plataforma compatible a la admitida por el compilador).

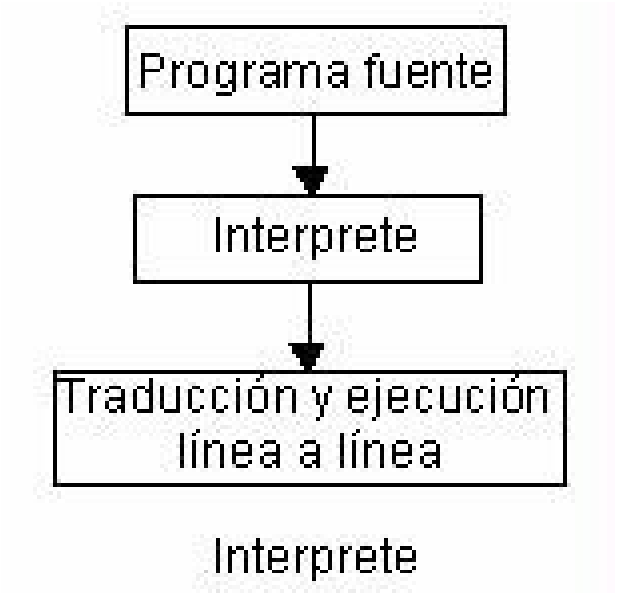
- **Código Ejecutable**

Fases de la compilación

El Proceso de Interpretación

En este caso el intérprete procesa instrucción por instrucción, sin generar un código ejecutable completo.

Cada vez que el usuario necesita ejecutar el programa deberá llamar al intérprete para que lo ejecute línea por línea.



Ventajas y Desventajas de Compiladores e Intérpretes

Ventajas

Compilación	Interpretación
Errores de sintaxis antes de la ejecución.	Ejecución en una sola etapa.
Velocidad de ejecución.	Proceso interactivo de depuración.
Protección del código.	

Desventajas

Compilación	Interpretación
Proceso en varias etapas y a menudo engorroso.	Errores de sintaxis detectados durante la ejecución.
Depuración laboriosa.	Baja velocidad de ejecución.
	Código abierto.

Depuración de Programas

Depuración o "debugging" significa eliminar los errores de un programa. En ciertos casos esa depuración es sencilla, pero a menudo constituye un proceso penoso. Esto depende de los tipos de errores de un programa.

Errores de un Programa

i) Errores en Tiempo de Compilación: son aquellos en los que se infringen las reglas que definen la estructura de las declaraciones y sentencias. Estos errores son denominados también errores de sintaxis.

ii) Errores en Tiempo de Ejecución: los programas contienen estos errores cuando, a pesar de contar con sentencias sintácticamente válidas, se producen errores al ejecutar estas sentencias.

iii) Errores de Lógica: en muchos casos el programa arroja resultados incorrectos a pesar de que no posea errores de sintaxis y tampoco errores al ejecutarse. En general se trata de los casos en que el programa no realiza las tareas que originalmente se solicitaron en la definición del problema.

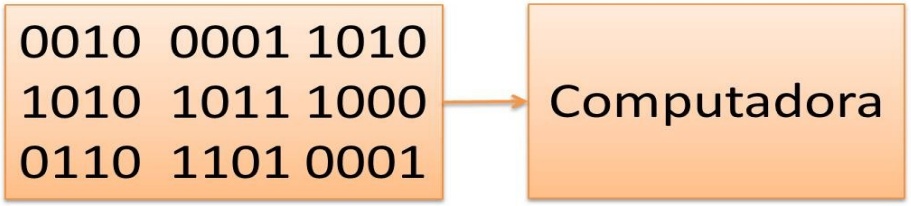
Lenguajes de Programación

En base a la similitud de estos lenguajes de programación respecto de nuestro lenguaje natural se los puede clasificar en 3 tipos: Lenguajes de Máquina, Lenguajes de Bajo Nivel (Ensambladores) y Lenguajes de Alto Nivel.

Lenguaje Máquina

El **lenguaje de máquina** son las instrucciones nativas de una **computadora** en particular

Los lenguajes de máquina son dependientes de la máquina



Por lo general los lenguajes máquina consisten en cadenas de números (**0s y 1s**) que instruyen a las computadoras para realizar sus operaciones más elementales

Ejemplo de codificación en Lenguaje Máquina

```
00110000 01000001 00000010   ; Carga el número 2 en el registro A

00110001 01000010 00000110   ; Carga el número 6 en el registro B

00000010 01000001 01000010   ; Suma los números en los registros
A y B
```

Lenguaje de Bajo Nivel (Ensambladores)

INSTRUCCIONES ARITMETICAS

ADD	Suma Acumulador
ADDC	Suma con Acarreo
DA	Ajuste decimal del acumulador
DEC	Decremento del operando
DIV	División Acumulador por B
INC	Incrementa la variable
MUL	Multiplicación Acumulador por B
SUBB	Resta con acarreo

Ejemplo de codificación en Lenguaje de Bajo Nivel

```
section .data
  number1   db  2      ; Definición de la variable number1 y asignación
del valor 2
  number2   db  6      ; Definición de la variable number2 y asignación
del valor 6
  result    db  0      ; Definición de la variable result y asignación
del valor 0

section .text
  global _start

_start:
  mov  al, [number1]   ; Mueve el valor de number1 al registro AL
  add  al, [number2]   ; Agrega el valor de number2 al registro AL
  mov  [result], al    ; Mueve el valor de AL a la variable result

  ; Salida del programa
  mov  eax, 1          ; Cargar el código de salida 1 en el registro
EAX
  xor  ebx, ebx        ; Limpiar el registro EBX
  int  0x80            ; Llamar al servicio del sistema para salir
```

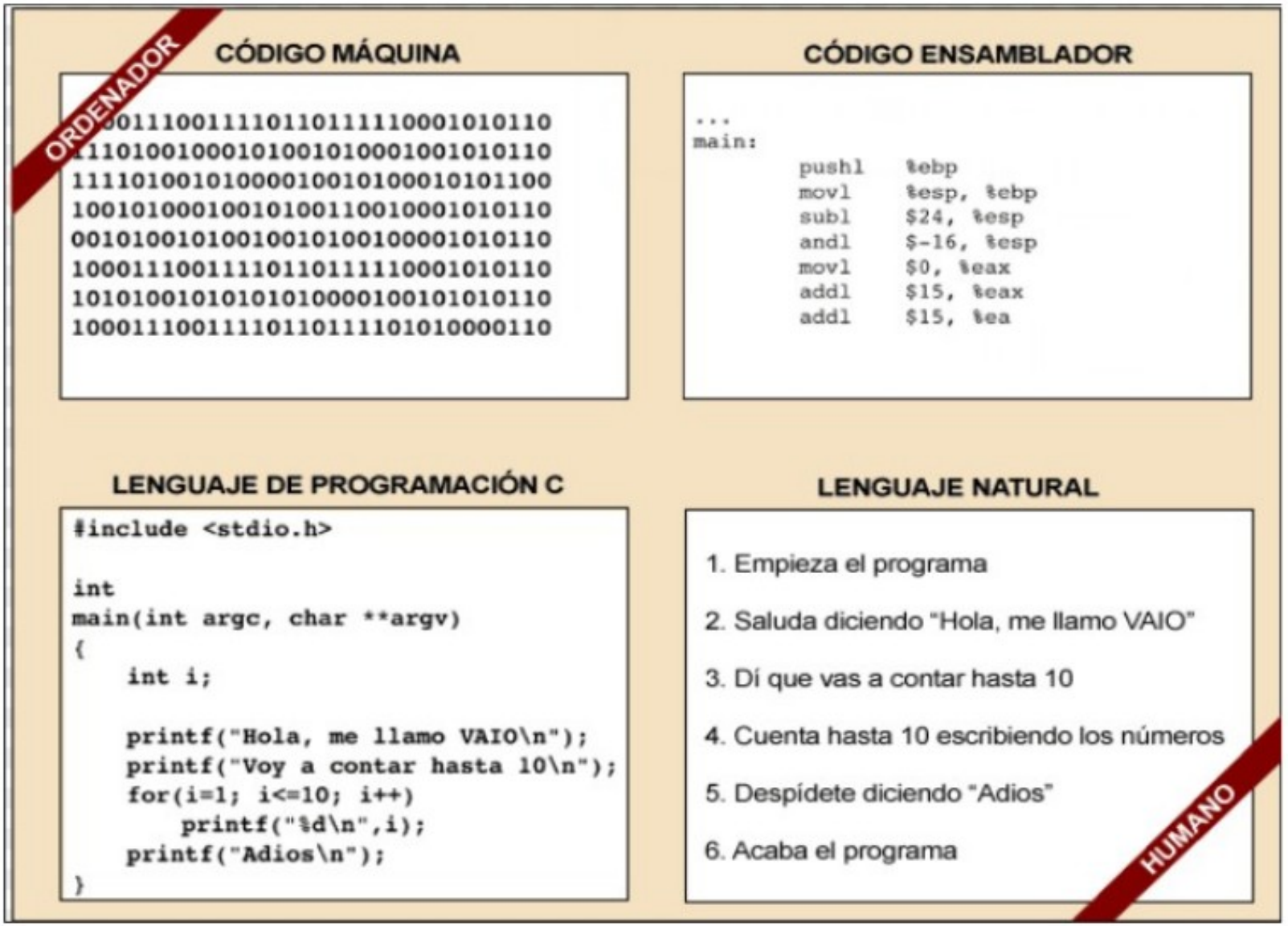
Lenguajes de Alto Nivel

Lenguaje de Alto Nivel

Ejemplos

- Java
- C
- C++
- C#
- FORTRAN
- PROLOG
- LISP
- PL/I
- SMALLTALK
- PowerBuilder
- BASIC
- Visual Basic
- Pascal
- Object Pascal
- ADA
- COBOL
- ALGOL
- CLIPPER
- Objective-C
- Phyton
- Modula 2
- Logo
- Ruby
- Eiffel
- Clarion
- Delphi
- Perl
- PHP
- Ocaml

Comparativa de lenguajes



Documentación de Programas

- Documentación Interna
  - Comentarios
    - Identificación (objetivo, e/s, fecha, autor…)
    - Rastros de diseño
    - Verificación
  - Nombres adecuados de elementos (constantes, variables, subprogramas)
- Documentación Externa
  - Resultados de las fases de
    - Análisis (documento de especificación, pruebas de escritorio)
    - Diseño (diseño modular, algoritmo, trazas)
    - Codificación (pruebas de ejecución)