3.1. Estructuras Secuenciales

La estructura secuencial es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso. Una estructura secuencial se representa de la siguiente forma:

```
Comienzo
  Accion1
  Accion2
    .
    .
  AccionN
Fin
```

3.1.1 Asignación

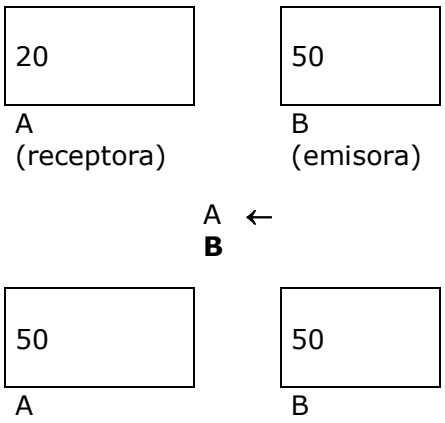
La asignación consiste, en el paso de valores o resultados a una zona de la memoria. Dicha zona será reconocida con el nombre de la variable que recibe el valor.

variable ← expresión (puede ser una variable, una constante, una expresión o fórmula a evaluar)

Ejemplos:

promedio ← suma /5
alumno ← nombre
cantidad-de-notas ← 5

Ejemplo de funcionamiento de la asignación: En las variables transfiero el valor de B a A



ASIGNACIÓN: la emisora mantiene su valor y la receptora lo modifica, el dato de la emisora reemplaza al valor anterior

Se pueden asignar operaciones Ej. A← B * 2

La asignación se puede clasificar de la siguiente forma:

- ✓ ***Simples:*** Consiste en pasar un valor constante a una variable (a←15)
- ✓ ***Contador:*** Es una variable que se incrementa, cuando se ejecuta, en una unidad o en una cantidad constante

contador ← contador + 1
multiplo ← multiplo + 3

- ✓ ***Acumulador:*** Es una variable que se incrementa en una cantidad variable

suma ← suma + numero

Donde "numero" es una variable que recibe distintos valores

Considerar:
- ✓ Una variable del lado derecho debe tener valor antes de que la sentencia se ejecute. Si "numero" no tiene valor antes de: suma ← suma + numero
  Se produce un Error LÓGICO. Se dice que "numero" no se ha **inicializado.**
- ✓ A la izquierda de una sentencia de asignación sólo puede haber variables. No puede haber operaciones.

Nota: la operación de asignación es una **operación destructiva** debido a que el valor almacenado en una variable se pierde o destruye y se sustituye por el nuevo valor de asignación. Ejemplo
numero ← 16
numero ← -23
numero conservará el último valor asignado, en este caso -23

3.1.2 Entrada

La entrada de datos consiste en recibir desde un dispositivo de entrada (p.ej. el teclado) un valor. Esta operación se representa en pseudocodigo como sigue:

Leer (a)
Leer (b)
Donde "a" y "b" son las variables que recibirán los valores

3.1.3 Salida

Consiste en mandar por un dispositivo de salida (p.ej. monitor o impresora) un resultado o mensaje. Este proceso se representa en pseudocodigo como sigue:

Mostrar ("El resultado es:", R)

Donde "El resultado es:" es un mensaje que se desea aparezca y R es una variable que contiene un valor.

3.2 Estilo de programación

Antes de escribir los algoritmos de los ejercicios en Pseudocódigo considerar:

Un programa legible y comprensible	Más fácil de:	- Entender - Corregir - Mantener

Seguir las siguientes sugerencias:

1. **SANGRADO O INDENTACIÓN**. En cada estructura, y alineando las instrucciones (sentencias) dentro de cada una de ellas y dentro de todo el algoritmo (Comienzo, Fin)
2. **LÍNEAS EN BLANCO**. Dejarlas entre partes importantes o que estén lógicamente separadas. Recomendable luego de cada estructura (Repetitiva o Selectiva), luego de declaración de variables.
3. **COMENTARIOS**. Parte importante de la documentación de un programa que permite mayor comprensión del mismo. (luego lo haremos en C) En Pseudocódigo entre **{ }**
4. **NOMBRES SIGNIFICATIVOS DE IDENTIFICADORES**. que representen aquello que estamos tratando. Si son palabras compuestas usar guión común. Todos deben comenzar con una letra.
5. **CADA SENTENCIA EN UNA LÍNEA DISTINTA**. Al colocar una nueva sentencia comenzar una nueva línea, incluso las palabras claves de las estructuras en líneas separadas.
6. **ESPACIOS ENTRE ELEMENTOS DE UNA SENTENCIA**. lo hace más legible. Por ejemplo:
   suma ← suma + numero