



TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN

SEMINARIO DE NIVELACIÓN de TUP 2025

MÓDULO ALGORITMIA

Profesores:

Ing. Axel Robles – Lic. Martín Meaurio – Lic. Gabriel Rivero



CLASE 1

Contenidos:

- ▶ Definiciones: Programación, Algoritmos, Estados, Acciones.
- ▶ Herramientas: Lenguaje Coloquial, Tabla Visual de Contenidos, Diagrama de Flujo. Pseudocódigo.
- ▶ Práctica: Problemas informales.

Desarrollo

¿Qué es programar?

Según el diccionario de la Real Academia:

- ▶ Establecer o planificar el programa de una serie de actividades. "Este año, además de básquetbol, se programarán diversas actividades, haciendo hincapié en la práctica de otros deportes como el voleibol o la natación".
- ▶ Dar las instrucciones necesarias a una máquina para que realice su función de manera automática. "Programar la lavadora; programar el video".

En definitiva, tenemos un dispositivo o conjunto de dispositivos capaces de comprender una orden o una serie de órdenes para efectuar una tarea.

Este dispositivo o conjunto de dispositivos lo llamaremos **procesador**. El procesador es capaz de alguna manera de comprender y ejecutar las acciones que se le solicitan.

Por este motivo diremos que la **Programación** es el conjunto de instrucciones reconocibles por una máquina de propósito general que la convierte en una máquina particular.

En la programación imperativa, estas instrucciones son las *acciones* que debe realizar el *procesador*. Una **acción** queda definida por su efecto. Es decir, la acción provoca que el procesador pase de un estado a otro. Por ejemplo, cuando se le indica a nuestra mascota "sentarse", ésta cambia de parado a sentado. Es decir, en el momento en que conozco el estado inicial y el estado final al que quiero llegar, puedo determinar qué acción se debe realizar.

Por lo tanto, ¿Qué es una Acción?

- ▶ Es un acontecimiento producido por un *Actor* que tiene un *Tiempo Finito* (período), produce un *resultado definido y preciso* y produce cierta transformación. Por ejemplo: Aprobar el Cursillo o Seminario de Ingreso.

Para validar que una acción se produjo, hay que verificar si hubo transformación

sobre los elementos que participan en la misma, ya para esto debemos observar y conocer el estado de los elementos antes y después de realizar la acción.

Muchas veces, una acción puede incluir múltiples acciones internas, por lo tanto, podemos afirmar que existen diferentes acciones:

- ▶ *Acciones Simples:* Se pueden realizar directamente.

Ejemplo: Estudiar las unidades incluidas en cada parcial. Realizar las actividades obligatorias. Asistir a las clases obligatorias.

- ▶ *Acciones Complejas:* No se pueden realizar directamente sino a través de una descomposición en acciones simples (menos complejas).

Ejemplo: Rendir los parciales.

Programar, en definitiva, es poder escribir el conjunto de estas acciones de modo tal que nuestro procesador, a través de sucesivos cambios de estado, llegue al estado final deseado.

Cuando se escribe el conjunto de acciones sin especificar un procesador en particular y empleamos lenguaje coloquial o algún diagrama decimos que escribimos un **algoritmo**.

Un algoritmo es:

- Una fórmula para resolver un problema.
- Un conjunto ordenado de pasos para resolver un problema en particular.
- Un conjunto *finito de acciones* o secuencia de operaciones que ejecutadas en un determinado *orden* resuelven un problema.

Ejemplo de Algoritmo - Sumar 2 números enteros:

1. Iniciar
2. Leer los números A y B.
3. Realizar la suma de los números A + B y el resultado asignarlo a C.
4. Escribir el resultado de C.
5. Salir.

En el mundo de la programación existen dos modos de representar un algoritmo cualquiera: el pseudocódigo y el diagrama de flujo. La diferencia fundamental entre ambos métodos radica en el modo en cómo se representa dicho algoritmo.

Pseudocódigo:

El pseudocódigo es una forma de escribir los pasos que va a realizar un programa de la forma más cercana al lenguaje de programación que vamos a utilizar posteriormente. Es como un falso lenguaje, pero en nuestro idioma, en el lenguaje humano y en español.

Generalmente este pseudocódigo no puede ejecutarse en una computadora (a veces si), ya que no está estructurado para que pueda ser interpretado por una PC, debido a que como lo mencionamos y su propio nombre así lo indica, se trata de un código falso. El pseudocódigo es un código escrito para que pueda ser interpretado a simple vista por los usuarios, no por el procesador de un dispositivo. Cabe destacar que también se le llama “Lenguaje de descripción algorítmico”.

En la actualidad el pseudocódigo es una de las formas más sencillas y eficaces de demostrar y comprender el funcionamiento de un programa de software, y aprender a utilizarlo correctamente nos permitirá llevar a cabo su programación de manera mucho más eficaz y rápida. Es decir que la principal aplicación del pseudocódigo es en la programación de software.

Ejemplo de Pseudocódigo:

```
algoritmo Sumar
variables
  entero a, b, c

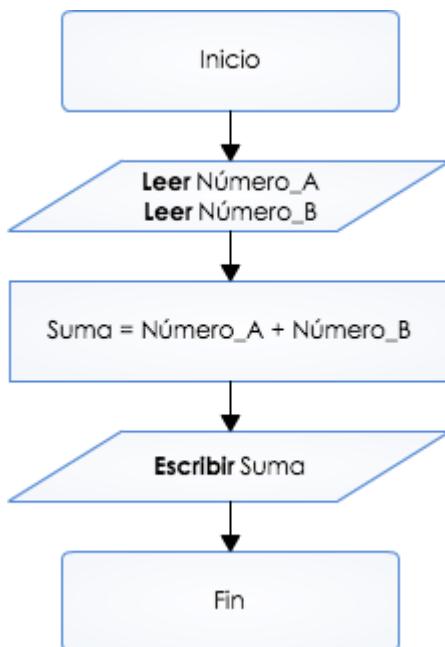
inicio
  escribir ( "Introduzca el primer número (entero): " )
  leer( a )
  escribir ( "Introduzca el segundo número (entero): " )
  leer( b )
  c ← a + b
  escribir ( "La suma es: ", c )
fin
```

Diagrama de Flujo:

El *Diagrama de Flujo* representa el algoritmo mediante un diagrama utilizando diversa simbología gráfica, denominados “bloques”, en las cuales se describen las acciones que debe ejecutar el algoritmo, conectados entre sí mediante líneas indicando el orden en que deben ser llevadas a cabo las instrucciones. Además de ello, el diagrama de flujo siempre contiene dos bloques fundamentales, el de inicio y el de final.

Ejemplo de Simbología y Diagrama de Flujo:

Símbolo	Nombre	Función
	Inicio / Final	Representa el inicio y el final de un proceso
	Línea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa la lectura de datos en la entrada y la impresión de datos en la salida
	Proceso	Representa cualquier tipo de operación
	Decisión	Nos permite analizar una situación, con base en los valores verdadero y falso



Retomando el concepto de algoritmo, veamos algún ejemplo de cómo es el proceso de resolución:

Ejemplo Práctico 1:

Se desea preparar una taza de té.

Se determinan los elementos necesarios para iniciar el algoritmo.

Datos de entrada: Tetera, Taza, Bolsita de Té.

Resultado de salida del algoritmo: Taza de Té.

Algoritmo:

1. Tomar la tetera.
2. Llenarla con agua.
3. Encender el fuego.
4. Poner la tetera en el fuego.
5. Esperar a que hierva el agua.
6. Tomar la bolsita de té.
7. Introducirla en la tetera.
8. Esperar 1 minuto.
9. Servir el té en la taza.

Ejemplo Práctico 2:

Se desea calcular la longitud de cable que se precisa adquirir para amarrar una antena de 100 m. Se sabe que las sujetaciones son 3, que se amarran a $\frac{4}{5}$ de la altura total en la antena y a 20 m de la base en el suelo. Además, se debe prever un 10% adicional para las sujetaciones.

Tenemos entonces la siguiente situación:

Debemos calcular la hipotenusa de un triángulo rectángulo cuya base es 20 m, cuya altura es $\frac{4}{5}$ de 100 m y este valor incrementarlo en 10%.

Nuestro Estado Inicial es:

- Altura de la antena $H=100$ m
- Altura del triángulo $h= ?$ Debemos calcularla
- Base del triángulo $b= 20$ m
- Largo del cable $L=?$ Debemos calcularlo

Podemos pasar a un estado intermedio donde sepamos la Altura del triángulo. La acción es: $h = (4*100) / 5$

Nuestro estado actual es:

- Altura de la antena $H=100$ m
- Altura del triángulo $h= 80$ m
- Base del triángulo $b= 20$ m
- Largo del cable $L=?$ Debemos calcularlo

Podemos ahora pasar a un siguiente estado donde podemos conocer el largo de uno de los tensores sin tener en cuenta el adicional para sujeciones. La acción es (raíz cuadrada) $L = \sqrt{b^2+h^2}$.

Nuestro estado actual es:

- Altura de la antena $H=100$ m
- Altura del triángulo $h= 80$ m
- Base del triángulo $b= 20$ m
- Largo del cable $L= 82,462$ m

Luego realizar la siguiente acción $L=L*3$

Nuestro estado actual es:

- Altura de la antena $H=100$ m
- Altura del triángulo $h= 80$ m
- Base del triángulo $b= 20$ m
- Largo del cable $L = 247,386$ m

Ya tenemos el largo del cable para los tres tensores. Aún nos falta calcular el adicional para las sujeturas. La acción es $L=L+L*10/100$

Nuestro estado actual es:

- Altura de la antena $H=100$ m
- Altura del triángulo $h= 80$ m
- Base del triángulo $b= 20$ m
- Largo del cable $L = 272,125$ m

Es seguro que no vamos a comprar esa cantidad, sino que compraremos 272 m. Hay formas para que el procesador tenga en cuenta estos detalles, pero eso es tema para más adelante.

Este algoritmo, en lenguaje matemático, sería:

$$1. h = (4*100) / 5$$

2. $L = \sqrt{b^2 + h^2}$

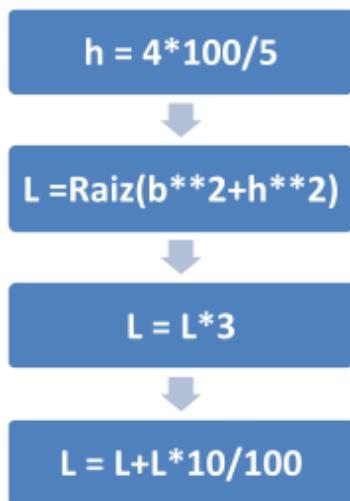
3. $L = L * 3$

4. $L = L + [(L * 10) / 100]$

Este algoritmo en lenguaje coloquial sería:

1. Calcular la altura de sujeción
2. Calcular la Hipotenusa del triángulo
3. Triplicar el valor de la Hipotenusa
4. Incrementar en 10% el último valor

También se puede expresar mediante un diagrama conocido como Diagrama de Flujo.



Ejercitación Clase 1

1. Especificar la acción principal y los estados iniciales y finales. Determinar los estados intermedios y las acciones que componen la acción principal.

Llamar a un amigo por teléfono público.

2. Especificar la acción principal y los estados iniciales y finales. Determinar los estados intermedios y las acciones que componen la acción principal.

Venir a la Facultad

3. Especificar la acción principal y los estados iniciales y finales. Determinar los estados intermedios y las acciones que componen la acción principal.

Preparación de mate amargo

4. Especificar la acción principal y los estados iniciales y finales. Determinar los estados intermedios y las acciones que componen la acción principal.

Cambiar una lámpara quemada

5. Indique los estados de las variables antes y después de cada operación.

```
X := 1
Y := 2
Z :=((X*Y)+4)/Y
X := Z ** Y
```

6. Indique los estados de las variables antes y después de cada operación.

```
A:= 1
B:= 5
C:= 6
X:= A
A:= B
B:= C
C:= X
```

CLASE 2

Contenidos:

- Constantes y Variables.
- La acción de Asignación.
- Composición de Acciones. Secuencia.
- Tipos de Datos y Operadores.
- Práctica: Problemas de composición secuencial y asignación.

Desarrollo

En cualquier programa siempre necesitaremos hacer cálculos, usar información, procesarla y mostrarla. En la mayoría de los casos, necesitaremos un lugar *temporal* en el cual guardar parte de esa información, incluso toda. Todos los lenguajes de programación nos permiten guardar datos en la memoria, para que cuando los necesitemos, podamos tomarlos, modificarlos y volverlos a guardar para usarlos más tarde.

El que se llamen de esta forma es porque podemos hacer que el contenido de ese lugar de almacenamiento varíe o cambie, es como si tuviésemos una serie de cajas y en ellas pudiésemos guardar cosas, con la salvedad de que en cada caja sólo puede haber una cosa a la vez; aunque también veremos cómo hacer que el contenido pueda variar y que varíe dependiendo de lo que contenía antes.

En el desarrollo anterior hemos usado letras para identificar elementos cuyo valor incluso era desconocido. Estas letras representan una **variable**.

Una **variable** puede identificarse por una letra o más por ej. ALTURA. Cuando hacemos referencia a la variable ALTURA en una operación, hacemos referencia al contenido de la variable. Así, si escribimos:

Máximo := ALTURA - 1,

Si ALTURA contiene el valor 6,

Máximo pasa a valer 5.

Asimismo, en el campo de la programación es importante resaltar el concepto de constantes y variables y cuál es la diferencia entre ellas:

- Una **constante** es un dato cuyo valor no puede cambiar durante la ejecución del programa. Recibe un valor en el momento de la asignación y éste permanece inalterado durante todo el programa.
En este sentido, es igual que una variable, ya que representa una zona de memoria en la cual se almacena un dato, pero éste no puede modificarse una vez asignado. Las constantes se utilizan para simplificar la programación. Por ejemplo, en un

programa se puede definir como constante el valor de **Pi**. Aunque es posible utilizar directamente el valor numérico, es más fácil usar el nombre **Pi**, que escribir el valor 3,1415926 cada vez que sea necesario.

Se recomienda usar un dato constante cuando voy a utilizar un valor definido y estático varias veces dentro de un programa.

- Una **variable** es un nombre asociado a un elemento de datos que está situado en posiciones contiguas de la memoria principal, y su valor puede cambiar durante la ejecución de un programa.

Una variable puede contener diferentes tipos de valores, de allí que cada variable, al ser **declarada** debe definir de qué **tipo** es. Es decir, qué rango o conjunto de valores puede almacenar.

Tipos de Datos:

En pseudocódigo vamos a utilizar los siguientes tipos de datos:

- **Datos Numéricos:** Son aquellos que representan una cantidad o valor determinado. Su representación se lleva a cabo en los formatos ya conocidos (enteros, punto y fracciones decimales si éstas existen). Estos pueden representarse en dos formas distintas:
 - *Enteros:* Es un conjunto finito de los números enteros. Los enteros son números completos, no tienen componentes fraccionarios o decimales y pueden ser negativos y positivos. Algunos ejemplos son: 10, 5, 0, -234, -3 ... Por ejemplo, un dato que represente la edad de una persona debe ser siempre del tipo Entero.
 - *Reales:* Consiste en un subconjunto de los números reales. Estos números siempre tienen un punto decimal y pueden ser positivos o negativos. Un número real consta de una parte entera y una parte decimal. Algunos ejemplos son: 0,345, 12.4, -231.005 Por ejemplo, un dato que represente el peso de una persona debe ser siempre del tipo Real.
- **Datos Alfanuméricos:** Son los datos que representan información textual (palabras, frases, símbolos, etc.). No representan valor alguno para efectos numéricos, o sea no podemos operar algebraicamente con ellos. Pueden distinguirse porque están delimitados por apóstrofes o comillas Los valores que acepta son los siguientes:

Ejemplos: "Hola Mundo" "30 de noviembre de 1974" "Enunciado cualquiera".

En el lenguaje aritmético no tenemos mayores dificultades. Los nombres expresan variables y los números constantes. Pero cuando trabajamos en un campo más extendido debemos contemplar el hecho que "ALTURA" exprese la cadena de caracteres y no el nombre de una variable. Este simple hecho de que las cadenas se indiquen entre comillas permite al procesador distinguir ALTURA de "ALTURA". El primero es el nombre de una variable y el segundo la cadena de caracteres.

- **Lógicos:** También se le denomina Booleano, es aquel dato que solo puede tomar uno de dos valores: Falso y Verdadero. Se utiliza para representar las alternativas (si/no) a determinadas condiciones. Por ejemplo, cuando se quiere saber si un valor entero es primo, la respuesta será verdadera o falsa, según si el número ingresado cumple con la condición de primo o no.

En los Lenguajes de Programación, los tipos de variables son:

- **Byte:** permite almacenar valores enteros de 0 a 255
- **Entero Corto (short):** permite almacenar valores enteros de 0 a 65.535
- **Entero Largo (double):** permite almacenar valores enteros de 0 a 4.294.967.295
- **Decimal:** permite almacenar números con decimales. En algunos lenguajes se debe indicar la cantidad total de dígitos y cuántos son decimales.
- **Booleano (bool):** permite solo dos valores Verdadero o Falso
- **Char:** permite almacenar un solo carácter.
- **Cadena (String):** permite almacenar caracteres. En algunos lenguajes se debe indicar el largo.

Cada lenguaje de programación especifica los tipos de variables aceptados y el comportamiento de las mismas. Algunos lenguajes no aceptan “mezclar” tipos y otros lo permiten pero producen truncamientos.

Es claro que el tipo de cada variable determina las operaciones que pueden efectuarse con ellos. Así en el caso de las variables numéricas se pueden efectuar todas las operaciones aritméticas. En el caso de las cadenas es posible efectuar la operación de concatenación (utilizando el símbolo & o +) que consiste en crear una cadena más larga donde la primera parte de la cadena es la correspondiente al primer operador y la segunda al segundo operador.

Por ejemplo:

1. nombre := “Juan”
2. apellido := “Perez”
3. apeynom := apellido + ”,” + nombre

El contenido de **apeynom** será “Perez, Juan”

Asignación

En general cuando se opera con variables, es preciso distinguir la operación de comparación de la operación de asignación. La asignación es una acción que resuelve las operaciones de la derecha del símbolo de asignación y el resultado lo almacena en la variable a la izquierda del símbolo de asignación.

Así en el caso de **A := B + C**, la acción calcula la suma de los contenidos de B y C y lo almacena en A.

Téngase en cuenta que el resultado almacenado dependerá del tipo de la variable receptora. Así es que, si B es un decimal y C un real y A un entero, el resultado almacenado en A será el resultado de la operación, pero solo su parte entera, truncándose la parte decimal.

Ejemplo B = 2,90 y C = 0,23

La operación da como resultado 3,13 pero si A es un entero contendrá el valor 3.

Es importante resaltar que en programación se deben distinguir tres etapas fundamentales en la vida de una variable:

- **Declaración:** Esta es la primera fase en la vida de cualquier variable. La declaración se realiza en la sección que comienza con la palabra **Ambiente** y se debe indicar a qué tipo pertenece la variable.
- **Inicialización:** Esto no es más que darle un valor inicial a una variable. Así como lo primero que se hace con una variable es declararla, lo siguiente tiene que ser iniciarla. Principalmente, existen dos maneras de otorgar valores iniciales a variables: mediante la operación de asignación como vimos anteriormente y mediante el proceso de entrada de datos por teclado, método que se va a profundizar más adelante.
- **Utilización:** Una vez declarada e iniciada una variable, es el momento de utilizarla. Esta es la parte que presenta un mayor abanico de posibilidades como ser: incrementar su valor, participar en una expresión y otras utilidades que ya iremos descubriendo.

En pseudocódigo:

- **Declaración:** Para declarar variables, lo único que debemos hacer es indicar el nombre de la variable y su tipo (numérico, lógico y cadena), antes del inicio del programa.

Ej: suma: entero

- **Instrucción de asignación:** Escribe sobre una variable el valor de una expresión. Así:

variable := expresión

Donde una expresión es una combinación de valores, variables y operadores. Los siguientes son algunos ejemplos de expresiones:

```
a := 5
b := c*d+(c-f)*m
z :=(x+y)/(w+s)
s:=(a/b)^3
```

Instrucciones de entrada y salida: Para que un programa pueda interactuar con el usuario debe haber un conjunto de instrucciones que permitan especificar tal interacción, y estas son las instrucciones de entrada y salida.

- **Instrucciones de entrada:** Permite tomar uno o más datos de un medio externo (comúnmente el teclado) y asignarlos a una o más variables, su representación en pseudocódigo es con la palabra reservada **Leer**, como se ve en el siguiente ejemplo:

```
Leer(var1, var2, ..., varN)
```

- **Instrucciones de salida:** Permite mostrar variables o constante en un medio externo (comúnmente la pantalla), y además imprimir mensajes al usuario al momento de solicitar un valor que sea ingresado por teclado en conjunto con la instrucción **Leer**. En pseudocódigo la instrucción asociada a la salida se realiza con la palabra reservada **Escribir** y tiene la siguiente forma:

```
Escribir("Texto1",var1, ..., varN)
```

Ejemplo:

Realizar un programa en pseudocódigo que resuelva la suma de dos números solicitados al usuario.

Solución:

La implementación en pseudocódigo del algoritmo se muestra a continuación:

```
Algoritmo Ejemplo
Ambiente
primerNumero, segundoNumero, suma: Entero
Proceso
    Escribir ("Ingrese un número")
    Leer (primerNumero)
    Escribir "Ingrese otro número"
    Leer (segundoNumero)
    suma := primerNumero+segundoNumero
    Escribir ("La suma de ambos es: ", suma)
FinAlgoritmo
```

Como se puede observar en el ejemplo, en la línea del Ambiente se realiza la declaración de las variables que se van a necesitar para resolver el problema en cuestión, paso seguido se debe informar al usuario por pantalla que debe ingresar los valores de entrada por teclado con la

sentencia **Escribir**, para luego asignar dichos valores a una variable con la sentencia **Leer**. Finalmente se almacena el resultado de la operación en una variable denominada suma, y se muestra el resultado por pantalla.

Operadores

Un operador es el símbolo que determina el tipo de operación o relación que habrá de establecerse entre los operandos para alcanzar un resultado.

Los operadores se clasifican en:

- **Aritméticos:** Son aquellos que permiten la realización de cálculos aritméticos. Utilizan operandos numéricos y proporcionan resultados numéricos.

Operador	Operación
+	Suma
-	Resta
*	Multiplicación
/	División real

- **Relacionales o de Comparación:** Son aquellos que permiten la comparación de dos valores. Utilizan operandos numéricos o alfanuméricos y proporcionan resultados lógicos.

Operador	Significado
<	Menor que
>	Mayor que
=	Igual que
<=	Menor o igual que
>=	Mayor o igual que
<>	Diferente de

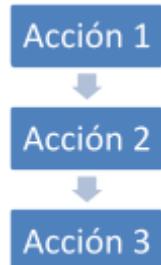
- **Lógicos:** Son aquellos que permiten la combinación de condiciones para formar una sola expresión lógica.

Operador	Relación
-	Negación (No)
^	Conjunción (Y)
v	Disyunción (O)

Estructuras secuenciales

Es importante comprender de qué manera el procesador ejecutará las acciones. En los casos estudiados hasta el momento, cuando uno establece una sucesión de acciones, éstas se ejecutan en el orden en que fueron secuenciadas. Es decir, de la primera a la última.

Se puede representar con un diagrama de flujo de acciones:



Harán que el procesador ejecute primero la *Acción 1*, luego la *Acción 2* y finalmente la *Acción 3*. En otras palabras, el estado inicial de la *Acción 2* será el estado final de la *Acción 1* y el estado inicial de la *Acción 3* será el estado final de la *Acción 2*.

Este tipo de estructura se denomina composición de acciones en secuencia o estructura en secuencia.

Ejemplo:

Ei B=? , C=? , A=?

•B:= 5

E2 B=5 , C=? , A=?

•C:= B*2

E3 B=5 , C=10 , A=?

•A:=C+B

Ef B=5 , C=10 , A=15

Ejercitación Clase 2

1. Indique los estados de las variables antes y después de cada operación.

1. $H := 3,14$
2. $D := 5$
3. $R := H*D$
4. $R := \text{redondear}(R)$

2. Dados los estados iniciales y finales, escribir las acciones.

Ei: $a=10$ $b=3$ $P=?$

Ef: $a=7$ $b=10$ $P=21$

No existe ninguna otra variable. Se debe tener en cuenta que asignaciones del tipo $a := a + b$, **primero** evalúan la expresión de la derecha y después efectúan la asignación.

3. Dados los estados iniciales y finales, escribir las acciones.

Ei: $a=6$ $b=15$ $P=?$

Ef: $a=9$ $b=6$ $P=27$

4. Suponga que se define la variable TOTAL como de tipo entero y las variables CANTIDAD e IMPORTE como de tipo real.

Indique el valor contenido por TOTAL luego de los siguientes pasos de un algoritmo.

```
CANTIDAD := 2
IMPORTE := 0,98
TOTAL := CANTIDAD * IMPORTE
```

5. Suponga que se define la variable TOTAL como de tipo real y las variables CANTIDAD e IMPORTE como de tipo entero.

Indique el valor contenido por TOTAL luego de los siguientes pasos de un algoritmo.

```
CANTIDAD := 2
IMPORTE := 1,02
TOTAL := CANTIDAD * IMPORTE
```

6. Suponga que se define la variable VALOR como de tipo entero y las variables DIVISOR y DIVIDENDO como de tipo real.

Indique el valor contenido por VALOR luego de los siguientes pasos de un programa.

```
DIVISOR := 2
DIVIDENDO := 5
VALOR := DIVIDENDO/DIVISOR
```

7. Supongamos una máquina de calcular donde sólo se pueden hacer operaciones de una a la vez. Escriba un algoritmo que realice la siguiente operación:

- a. $(10 \times 4 + 5)/2$
- b. $(\sqrt{(A+B) * C - D}) / [a * (c-a)]^n$ donde $n=2$

CLASE 3

Contenidos:

- Composición de Acciones: Acciones condicionadas. Estructura de Decisión. Decisión Múltiple.
- Práctica: Problemas con Acciones Condicionadas.

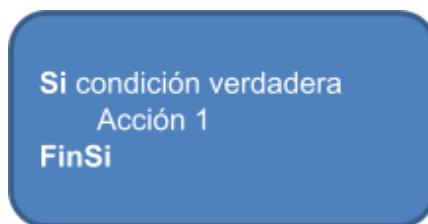
Desarrollo

La estructura de acciones en secuencia es una estructura que permite solucionar un cierto número de problemas. Pero hay situaciones en las que una acción sólo debe ejecutarse bajo determinadas condiciones. Otras donde se debe elegir una acción u otra acción en función de estas condiciones.

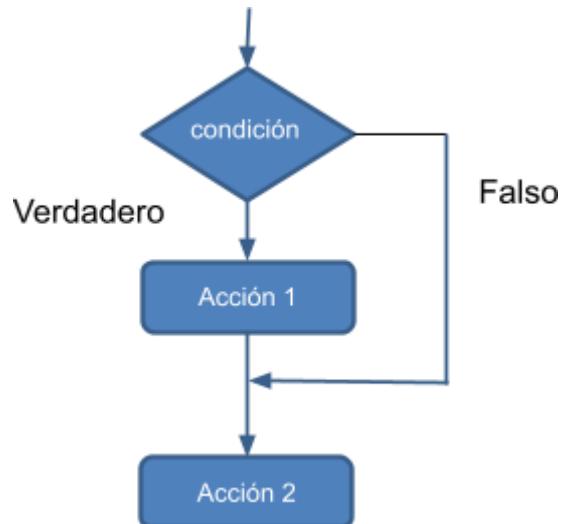
No es algo distinto a lo que habitualmente hacemos en nuestra vida cotidiana, permanentemente elegimos hacer determinadas cosas en función de prioridades y condiciones. Elegimos caminar o tomar un colectivo, tomar té o café. Si tenemos sed bebemos agua, si tenemos hambre elegimos comer una fruta o un yogur. Estas decisiones son tomadas en función de condiciones que nos impulsan en uno u otro sentido.

En el armado de un algoritmo también se presentan situaciones en que se debe decidir si se ejecuta una u otra acción. Para ello se emplean las estructuras de decisión.

Una estructura de decisión parte de una pregunta por el estado de algunas variables, en función de ello se ejecuta (o no) una determinada acción.



Su representación en Diagrama de Flujo sería:



Por ejemplo, supongamos que tenemos dos variables A y B, y que dependiendo que A sea mayor que B se realiza la resta A-B y en caso contrario se efectúa la resta B-A, resulta:

En pseudocódigo:

```
Si A > B  
    C := A - B  
Sino  
    C := B - A  
FinSi
```

Debe prestarse mucha atención a la tabla de verdad de la expresión booleana de la condición. En el ejemplo sencillo que hemos presentado el hecho de que sea falso el resultado de **A > B**, no significa que **B** sea mayor que **A**, ¡Pueden ser iguales!

Las expresiones pueden ser complejas mediante el empleo de conjunciones (\wedge) o disyunciones (\vee). Por ejemplo, **A > B \wedge B > C** en este caso para que la condición sea verdadera ambos términos deben ser verdaderos, y en la expresión **A > B \vee B > C** para que sea falsa ambos términos deben ser falsos.

Así recordemos:

A > B	B > C	A > B ∧ B > C
V	V	V
V	F	F
F	V	F
F	F	F

A > B	B > C	A > B ∨ B > C
V	V	V
V	F	V
F	V	V
F	F	F

Muchas veces es preferible, en lugar de una expresión muy compleja, efectuar una estructura de decisión anidada. Así tenemos:

```

Si A > B ∧ B > C
    Acción 1
Sino
    Acción 2
FinSi

```

```

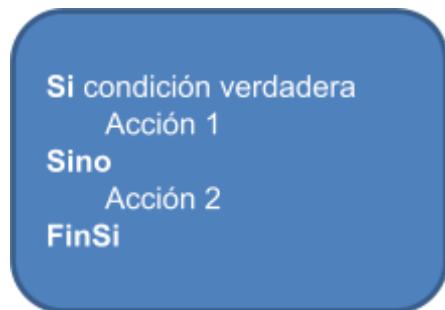
Si A > B
    Si B > C
        Acción 1
    Sino
        Acción 2
    FinSi
Sino
    Acción 2
FinSi

```

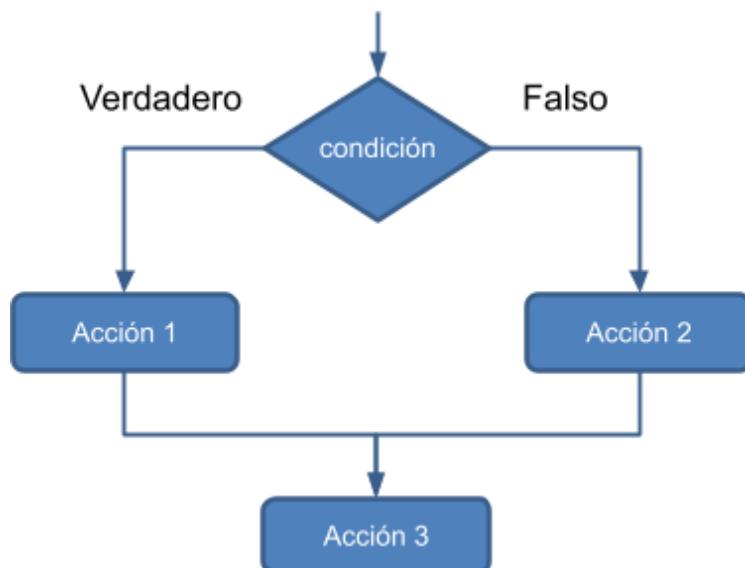
Observamos que la Acción 1 sólo se ejecuta cuando ambas condiciones son verdaderas.

Las estructuras condicionales también pueden ser *dobles*. Estas permiten elegir entre dos opciones o alternativas en función del cumplimiento o no de una determinada condición.

En pseudocódigo se representa:



Su representación en Diagrama de Flujo sería:



Ejercitación Clase 3

1. Elaborar un algoritmo para determinar el tipo de un triángulo dado el tamaño de sus lados.
2. Elaborar un algoritmo para determinar el mayor de cuatro valores.
3. Elabore un algoritmo para representar la acción de servir la merienda. El comensal puede tomar té, café, café con leche, té con leche y puede o no ponerle azúcar, si le pone puede ponerle una, dos o tres cucharadas.
4. El sistema de registración de IVA debe asignar la letra a un comprobante de venta. Para ello se debe tener en cuenta que la empresa emisora del comprobante emite comprobantes 'C' en el caso que la misma revista el carácter de Responsable No Inscripto (RNI), Monotributista (M) o Exento (E), cualquiera sea el carácter fiscal del comprador. Si la empresa emisora es Responsable Inscripto (RI), emitirá un comprobante 'A' si el comprador es RI o es RNI, pero emitirá un comprobante 'B' si el comprador es M o E.
5. Escriba un algoritmo que acepte tres números y luego los devuelva ordenados decrecientemente.
6. Se desea calcular el promedio de votos a obtener por cinco partidos políticos. Se realiza una encuesta acumulando los votos obtenidos por cada partido y el número de votos indecisos. Realizar un algoritmo que determine el porcentaje de votos de cada partido y el de indecisos, respecto del total de encuestados.
7. La nota final en la escuela de informática se obtiene en función de 3 notas. La nota final del primer examen, la nota del segundo examen y una nota de concepto del profesor. La nota del primer examen se pondrá como 30% de la nota final, la del segundo examen como el 50% y la de concepto como el 20% de la nota final. Elabore un algoritmo que en base a esas tres notas calcule la nota final.
8. La Mutual del personal del Banco, a pedido de un socio, canjea un plazo fijo, pagándole el capital inicial. Luego, al cobrarse el plazo fijo, le paga los intereses descontando los días correspondientes y \$ 0,50 por gastos. Efectúe un algoritmo que liquide el pago de la diferencia entre el capital inicial y capital con intereses (el interés es de 1,5% a 30 días). Se cuenta con los datos: Capital inicial, Capital con interés, Cantidad de días de anticipo.
9. El valor de la sección (S) de un conductor se determina en función de la corriente eléctrica (I) y de la conductividad (C) del material. Además, a la sección obtenida se le incrementa un 25% por razones de seguridad. Tenga en cuenta que $(C=I/S)$. Realizar un algoritmo que calcule la sección.
10. Escriba un algoritmo que determine el precio del peaje a abonar por el pasajero en función de los km que va a recorrer, sabiendo que hasta 10 km el precio es \$ 200, hasta 20 km, el precio es \$ 300, hasta 40 km, el precio es \$400 y hasta 80 km el precio es \$500, si supera los 80 Km el costo es de \$600.

11. En una empresa el valor del pasaje se cobra en función de los destinos. El destino 1 paga \$200 y a partir de allí existen 7 destinos más. Cada destino tiene un costo de 15% más que el anterior. Escriba un algoritmo que devuelva el valor a pagar en función del número de destino elegido.
12. Escriba un algoritmo que acepte una fecha como día, mes y año (en números) y devuelva la fecha en formato “largo”. Por ejemplo, se ingresa 25 (día), 10 (mes) y 2003 (año) y devuelve el texto: “25 de Octubre de 2003”. Tenga en cuenta que deberá controlar que la fecha sea válida.
13. Escriba un algoritmo que solicite la fecha de nacimiento de una persona y la fecha de hoy y me devuelva la edad de la persona. Tenga en cuenta que deberá controlar que la fecha sea válida.
14. En el ejercicio anterior tenga en cuenta que la edad se expresa en años salvo el caso de menores de un año que se expresa en meses y en el caso que tenga menos de un mes que se expresa en días.

CLASE 4

- Composición de Acciones: Acciones que se repiten. Estructura de Iteración. Iteración con condición de continuación. Iteración con condición de terminación.
- Práctica: Problemas con Iteración.

Desarrollo

En la solución de problemas (como en la vida cotidiana) muchas veces es necesario repetir una acción un cierto número de veces, donde cada vez que se ejecuta se produce una aproximación al resultado esperado. Es decir, cada vez que se ejecuta la acción se llega a un nuevo estado que al arribar a las condiciones apropiadas será el estado final deseado.

Tomemos el caso de la multiplicación. Podemos interpretar que **AxB** consiste en sumar **B** veces el valor de **A**. Es decir que **M:=M+A** debe ser ejecutado **B** veces partiendo con **M=0**, donde **M** es una variable temporal en donde almacenaremos el valor temporal para cada ciclo de la multiplicación.

Debemos entonces contar con alguna estructura que le indique al procesador esta actividad. Nótese que podemos razonar así: La primera vez tenemos **M:=0** que es el resultado de **A*0**, para conseguir el resultado de **A*1** debo hacer **M:=M+A**. Ahora tengo el resultado para **A*1**, y para conseguir el valor de **A*2**, debo nuevamente hacer **M:=M+A**. Y así sucesivamente hasta que lleguemos al resultado **A*B**.

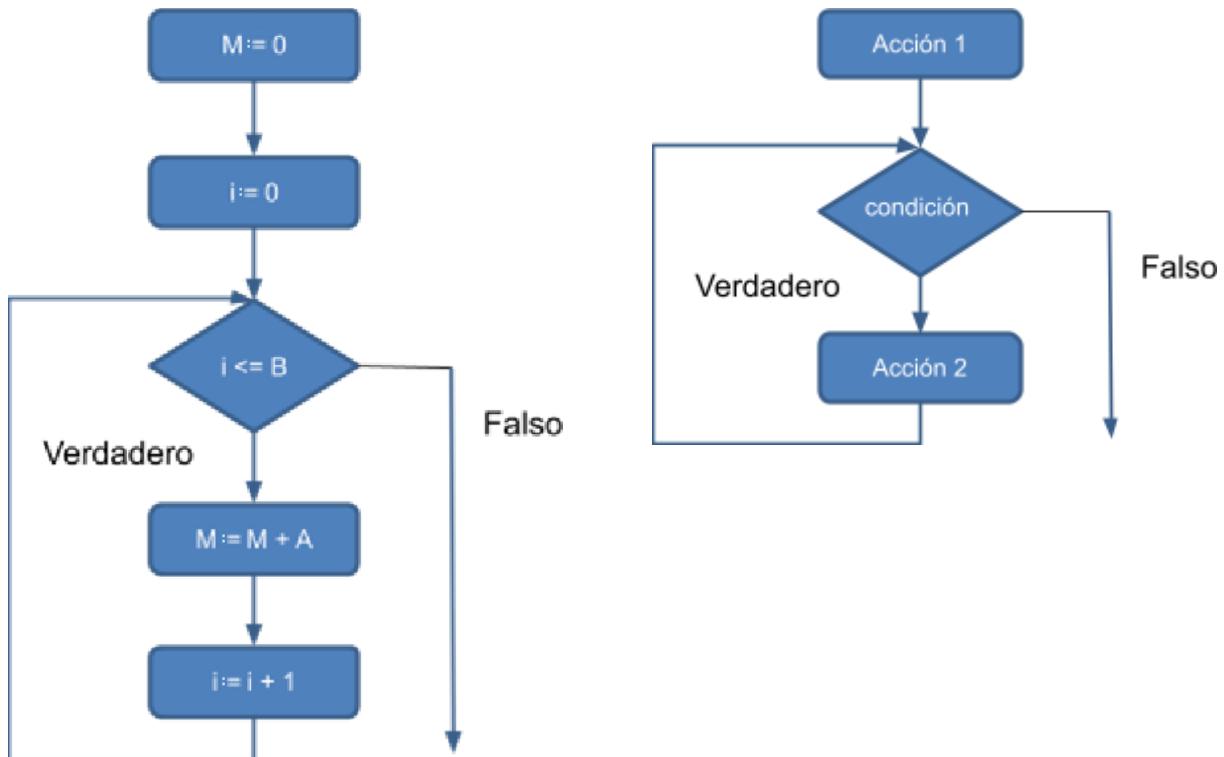
Está claro que, para un valor de **i** cualquiera, conocemos el valor de **A*(i-1)**, efectuamos **M:=M+A** y ahora tenemos el valor de **A*i**. Si hacemos **i:=i+1**, estamos en el mismo estado en el que empezamos para un **i** mayor.

La estructura de este tipo se conoce como iteración. Y el algoritmo para calcular **AxB** en pseudocódigo sería:

```
M:= 0
i:= 1
Mientras ( i <= B ) Hacer
    M:= M + A
    i:= i + 1
FinMientras
```

Obsérvese que siempre antes de ejecutar la acción conozco **M** para **i-1** y luego de las acciones regreso a ese estado. Cuando **i:=B+1**, entonces conozco **M** para **i-1**, es decir para **B**. En esas condiciones **i>B** y termina la iteración.

En diagrama de flujo sería:



Se estructura en pseudocódigo sería:

```

Accion1
Mientras (condición verdadera) Hacer
    Accion2
FinMientras
    
```

La mayoría de los lenguajes poseen varias estructuras de iteración.

Aparte de la que hemos visto, que se denomina de **Repetición con Condición al Inicio** (o con condición de terminación) o **Pretest** o **Mientras**, muchos tienen una estructura de **Repetición con Condición al Final** (o con condición de continuación), también conocida como **Posttest** o **Repetir**.

Esta estructura sería:

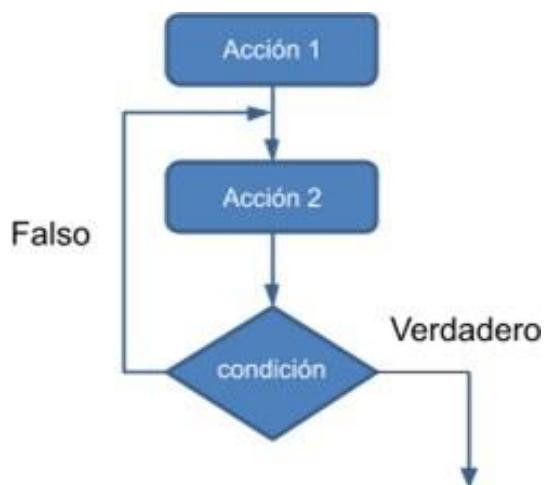
```

Accion1
Repetir
    Accion2
    Hasta que condición
  
```

Nótese que en esta estructura la Acción2 se realiza por lo menos una vez, en el caso de la estructura **Pretest** puede no ejecutarse nunca.

También debemos advertir que el Postest sale del ciclo cuando la condición es **verdadera**, a diferencia del Pretest, que sale cuando es **falsa**.

Su representación en Diagrama de Flujo sería:



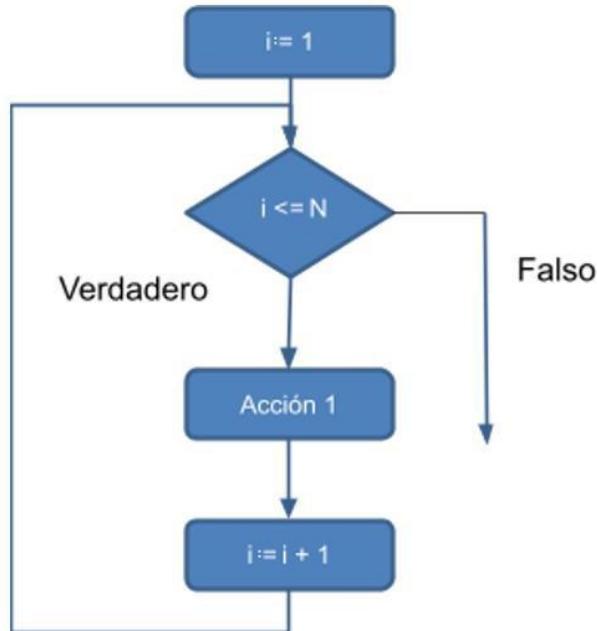
Por último, una estructura muy difundida es aquella en que la cantidad de repeticiones está establecida. Este tipo de repeticiones se conocen como **Manejada por contador** o **Para**. Un ejemplo de esta estructura en pseudocódigo sería el siguiente caso:

```

Para i:=1 hasta X (,paso)
    Accion1
FinPara
  
```

La estructura permite ir de 1 en 1 o de 2 en 2 o en general de **n** en **n (paso)**. El valor inicial de **i** también puede ser cualquier entero. Es muy práctico para aquellos casos donde es posible determinar el número de repeticiones necesarias, pero no es posible de usar cuando la condición de terminación o continuación es aleatoria.

Su representación en diagrama de flujo sería similar a las anteriores, pero hay que considerar que la condición sólo evalúa el valor de una sola variable que se incrementa de n en n .



Ejercitación Clase 4

1. Escriba un algoritmo que pida la base y el exponente de una potencia y calcule el valor resultante, bajo la premisa que el resultado es la multiplicación de la base por sí misma tantas veces como indica el exponente. Tenga en cuenta que si el exponente es 0 el valor de la potencia es 1 cualquiera sea la base.
2. Si en el ejercicio anterior no contempló exponentes negativos, replantee el ejercicio para contemplar este caso.
3. Elaborar un algoritmo para ordenar de mayor a menor cuatro valores.
4. Describa y especifique en qué se diferencia la iteración con condición al inicio y con condición al final.
5. Se dispone de una lista de personas con número de documento, apellido y nombre y número de teléfono. Dicha lista se encuentra ordenada por el número de documento. Cree un algoritmo que efectúe una búsqueda de un número dado.
6. Escriba un algoritmo para calcular el factorial de un número dado.
7. Escriba un algoritmo que, dado un número, muestre y cuente los números pares desde 1 hasta el número dado.
8. Escriba un algoritmo que, dado un número, muestre y cuente los números pares desde 51 hasta el número dado.
9. Un **palíndromo** (del griego *palin dromein*, volver a ir hacia atrás) es una palabra, número o frase que se lee igual hacia adelante que hacia atrás. Si se trata de un número, se llama capicúa. Habitualmente, las frases palindrómicas se resienten en su significado cuanto más largas son. Genere un algoritmo que permita introducir letra por letra un texto de hasta 20 caracteres y determine si es un palíndromo.