Pseudocódigo

2.1 ¿Qué es el Pseudocódigo?

Es un lenguaje creado especialmente para la realización de algoritmos; la característica principal de éste es que se pensó para el entendimiento del humano y no el de la máquina. Por ello es que se considera un lenguaje sencillo. Como todos los algoritmos se deben ejecutar en una máquina, es necesario traducir el pseudocódigo a un lenguaje de programación, siendo considerado un borrador, por esto es utilizado en textos donde no está definido un lenguaje de programación en particular, haciendo de él uno universal.

El pseudocódigo es considerado un lenguaje de **alto nivel** (*pie de página:* Se caracteriza por ser más entendible por el humano que los de un nivel inferior) y posee una estructura secuencial.

2.2 Características.

- Se puede ejecutar en un computador, por medio de un IDE (pie de página: Entorno de desarrollo integrado).
- Al ser sencillo, facilita la realización y comprensión de un algoritmo.
- No depende de un lenguaje en particular.

Pero a su vez, comparte características con lenguajes de programación formales, tales como:

- **Las variables:** Espacios de memoria que pueden cambiar de contenido a lo largo de la ejecución de un programa y se accede mediante un identificador (nombre).

  Constantes como su nombre lo indica, no cambian en el transcurso de la ejecución del programa.

- **Entradas y Salidas (INPUT/OUTPUT):** Las entradas son los datos que el usuario ingresa para procesar en el programa. Y las salidas son las respuestas que el programa entrega al usuario.

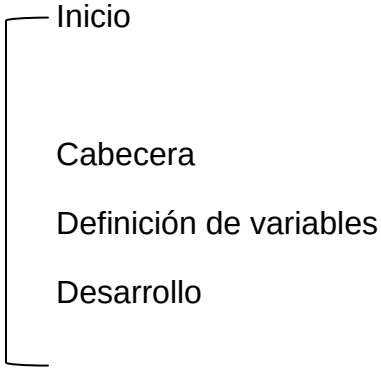
- **Declaración de variables:**Se define el tipo de dato de la variable a utilizar en el algoritmo, esta puede ser entero, real, carácter, etc.

- I**nstrucciones de control:** Principalmente utilizaremos dos, instrucción de control “SI” e instrucción de control “MIENTRAS”. Más adelante se detallarán.

2.3 Sintaxis.

La sintaxis del pseudocódigo es de carácter personal, ya que existen muchas formas de escribirlo, pero en esta ocasión utilizaremos la siguiente.

2.3.1 Estructura.



- **Cabecera:** En esta sección se encuentran las variables y constantes.

- **Definición de variables:** En esta sección se declaran las variables a utilizar en el programa junto con su tipo (ver 1.3.2.1).

- **Desarrollo:** En esta sección se encuentran todas las instrucciones correspondientes al programa.

2.3.2 Instrucciones.

2.3.2.1 Asignación.

Se utiliza para dar valores a las variables creadas en el programa. Estas pueden ser valores numéricos, expresiones algebraicas, e incluso otras variables detalladas a lo largo del texto.

Sintaxis:

<nombre_variable> = Contenido

Donde:

- **nombre_variable** = Es el identificador anteriormente mencionado.

- **Contenido** = Dato, número o expresión que se quiere asignar a la variable.

2.3.2.2 Instrucciones de entrada y salida.

- Leer: Esta instrucción tiene por objetivo ingresar los datos del usuario al programa para que éste, al ejecutarse, los procese para un posterior uso.

- Escribir: Esta instrucción muestra por pantalla, tanto valores referenciados en el programa, como textos añadidos a éstos. Los textos mencionados se incluyen ente comillas (Escribir “HOLA”).

2.3.2.3 Instrucción de selección (Si).

Estas instrucciones condicionan la ejecución de trozos de código que se encuentren en su interior, presentando expresiones, que de cumplirse permiten la ejecución de las líneas de programa que estén a continuación.

Sintaxis:

```
Si(condición) Entonces
        Instrucciones;
Fin Si
```

Además existe una instrucción llamada “Sino”, que se utiliza para ejecutar algunas líneas de código cuando la condición del “Si” no se cumple.

Sintaxis:

```
Si(condición) Entonces
        Instrucciones;
Sino Entonces
        Instrucciones;
Fin Si
```

2.3.2.4 Instrucción de Iteración (Mientras).

Pertenece al grupo de las sentencias de bucle. Ya que ejecuta sus instrucciones infinitas veces siempre y cuando su condición sea verdadera. Por ende se verifica la condición del “Mientras”; Si esta es verdadera se ejecutan todas susinstrucciones y se vuelve a preguntar la condición. Si esta se cumple, el ciclo vuelve a funcionar, caso contrario, se abandona el bucle.

Sintaxis:

```
Mientras(condición) Hacer
        Instrucciones;
Fin Mientras
```

A continuación, presentamos algunos de los símbolosmás utilizados para la elaboración de un algoritmo.

Signo	Operación	Ejemplo
>	Mayor que	25 > 10
<	Menor que	0 < 10
>=	Mayor o igual	20 >= 20
<=	Menor o igual	40 <= 41
==	Igual que	30 == 30
!=	Distinto que	0 != 1

También, presentamos algunos operadores aritméticos básicos:

Operador	Significado	Ejemplo
+	Suma	1 + 1
-	Resta	2 – 1
/	División	½
*	Multiplicación	2*2
%	Resto de una división	1%2

2.4 Ejercicios Resueltos.

I. Del Algoritmo:

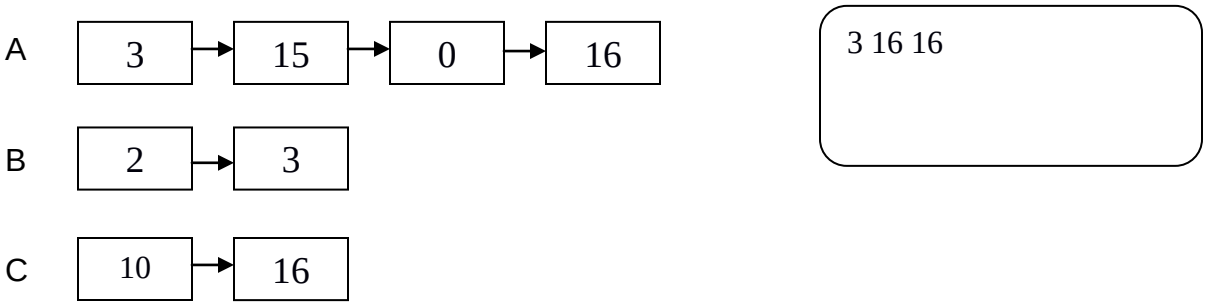
```
1. A=3
2. B=2
3. C=10
4. Escribir(A)
5. A=A+B+C
6. B=B+1
7. C=A+1
8. Escribir(C)
9. A=A-15
10. A=A+C
11. Escribir(A)
```

a) ¿Qué valor tiene la variable A en la línea 4?

Solución:

Memoria

1. **Pantalla**



Su valor es 16.

b) ¿Qué valor tiene la variable A en la línea 9?

Solución:

Su valor es 0.

c) ¿Qué pasaría si en vez de poner en la línea 11`imprimir(A)`, ponemos`imprimir("A")`?

Solución:

Por pantalla se mostraría el carácter A, ya que la encerrarlo entre comillas activamos el modo texto.

II. Del Siguiente Algoritmo

```
1.  A=2
2.  si A>=1 entonces
3.       Imprimir(A+1)
4.  sino
5.       imprimir(AMOD2)
6.  fin si
```

a) ¿Que sucede en la línea 3?

Solución:

Se imprime el valor 3 por pantalla.

b) ¿Que se lanza por pantalla en la línea 5?

Solución:

0 (cero), ya que el resto entre la división de 2 y 2 es 0.

c) ¿Que pasaría en la línea 3, si en la línea 1 coloco A=3?

Solución:

Se lanzaría por pantalla el valor 4

d) ¿Quépasaría si en la línea 5 coloco A/2?

Solución:

Se lanzara por pantalla el valor 1.

III. Cree una secuencia que al ingresar una contraseña, esta verifique si es correcta o no. En el caso que no sea correcta envié un mensaje al usuario para que este la vuelva a ingresar, caso contrario mande un mensaje para indicarle que esta correcta.

Solución:

Contraseña correcta: 15789

```
1. Entero Contrasena
2. leer(Contrasena)
3. Mientras Contrasena!=15789 entonces
4.       imprimir(“Contraseña incorrecta, vuelva a ingresar”)
5.       leer(Contrasena)
6. fin Mientras
7. imprimir(“Contraseña Correcta”)
```

IV. Cree una secuencia que considerando un numero aleatorio, el usuario en cinco intentos lo adivine (hint: utilice la función azar() para el numero aleatorio).

Solución:

```
1.  Entero numero, numero_a_adivinar, intentos
2.  intentos=1
3.  Numero_a_adivinar=azar()
4.  Leer(numero)
5.  Mientras intentos<=5 entonces
6.  Si numero==numero_a_adivinar entonces
7.  Escribir("Estas en lo correcto")
8.           Intentos=6
9.       Sino
10.          intentos=intentos+1
11. Leer(numero)
12. Fin Mientras
```