APRENDE A PROGRAMAR DESDE CERO

Por Mayra Moyano

Aprende a programar desde cero

¿Qué es programar?

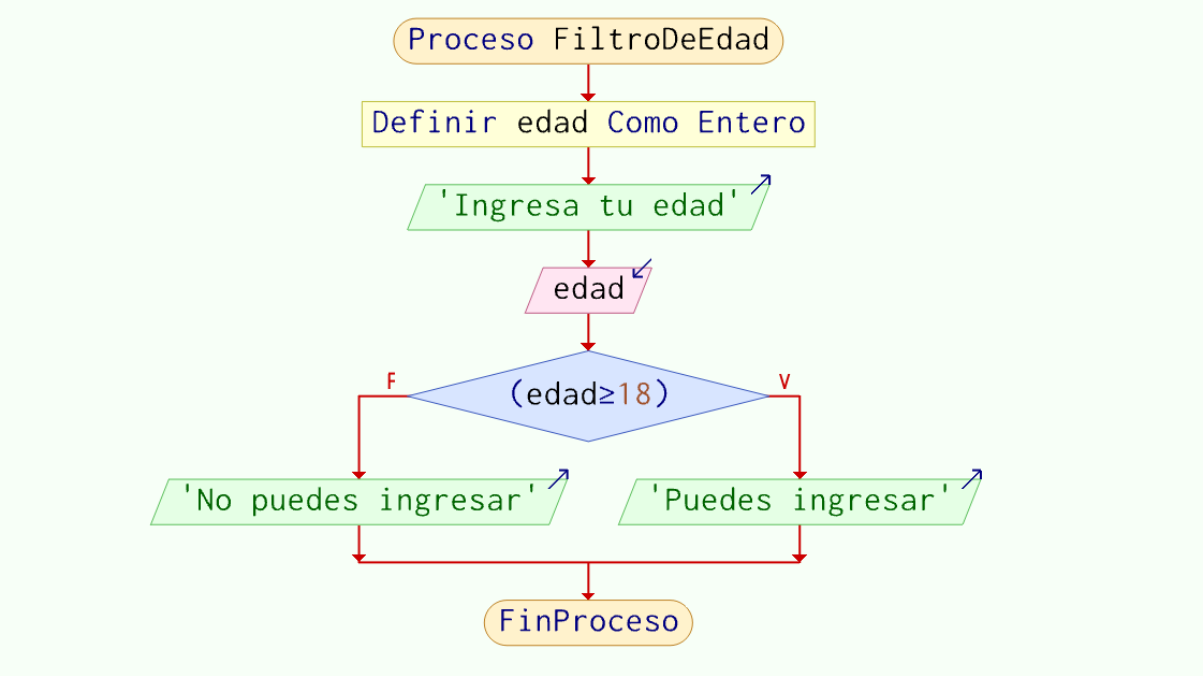
Programar es decir **¿qué hacer?** a una maquina mediante instrucciones.

Estas instrucciones o pasos lógicos se siguen para poder llevar a cabo una tarea o resolver un problema. Este proceso se denomina como **ALGORITMO**.

La forma de representar gráficamente este algoritmo es mediante un **DIAGRAMA DE FLUJO.**

Un diagrama de flujo muestra la secuencia de pasos necesarios para poder llevar a cabo una tarea o resolver un problema. Mediante símbolos y conectores.

Ejemplo de un diagrama de flujo:



Bien, ¿qué podemos entender de este diagrama?

Bueno este diagrama grafica un algoritmo que intentaba resolver una tarea. Esta tarea era que la persona ingresara su edad y a partir de eso el programa iba a tomar la decisión si la persona podía ingresar o no al establecimiento.

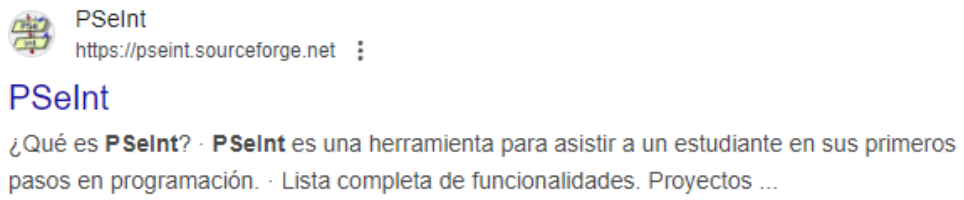
La condición que debía cumplir la persona era tener o ser mayor de 18 años. Si la cumplía podía ingresar y sino no podía ingresar.

Bueno una vez que entendamos estos conceptos pasemos al siguiente nivel.

PSEUDOCODIGO

Para esta nueva etapa vamos a descargar el siguiente programa:

https://pseint.sourceforge.net/?page=descargas.php



PSeInt es un programa que trabaja con seudocódigo.

Y este es una forma de escribir los pasos que realizará un programa de la forma más cercana al lenguaje humano para traducirlo a un lenguaje de programación.

Bien comencemos...

¿Qué es una variable?

Una **variable** es un espacio de almacenamiento en la memoria. Para explicarlo más simple te brindare un ejemplo de que es.

Supongamos que tengo un **COFRE** (nombre de la variable) y dentro de este cofre quiero guardar un **MAPA DEL TESORO** (valor).

Bueno esto hace una variable almacena un valor.

Así se vería por ejemplo la declaración de una variable en Python.

Cosas que debemos de saber sobre las variables:

- No puede tener espacio entre caracteres.
- No pueden iniciar con números.
- No pueden incluir caracteres especiales.
- No puede llevar el mismo nombre que una palabra reservada.
- El nombre de la variable debe de ser acorde al tipo de dato que va almacenar. Por ejemplo, no puedo llamar a mi variable Letras y que sirva para almacenar un número.
- Las variables pueden cambiar de valor durante la ejecución del programa.

Formas de escribir una variable:

- **PascalCasing**: la palabra letra de cada palabra va en mayúscula
- **camelCasing**: la primera letra de cada palabra va en mayúsculas, excepto la primera
- **sanke_case**: lleva un guion bajo para separar las palabras

Veamos los tipos de datos ahora…

Tenemos el tipo de dato:

- **ENTERO** que este guarda números naturales. Por ejemplo **2, 34, 27, …**
- **REAL** este guarda números que conlleven “,”. Por ejemplo **2,4**
- **CARACTER** guarda texto, caracteres especiales, etc. Por ejemplo **“Hola Mundo”**
- **LOGICOS** solo trabaja con datos booleanos. Que son **True** o **False**.

Antes de continuar con el ejercicio debemos de saber lo siguiente:
¿QUE ES UN USUARIO? Un usuario es la persona que va a usar el programa.

Resuelve el siguiente ejercicio:

1. Estoy haciendo un dulce de leche y me pide agregar 1,5 ¿Qué tipo de dato es 1,5?
2. Quiero crear un programa donde el usuario me ingrese su edad ¿Qué tipo de dato me va a ingresar?

¿Lograste resolverlo? Bien continuemos entonces…

Otro aspecto a saber para ejecutar el programa en PseInt debemos de apretar el siguiente icono:

Pongamos en práctica estos 2 ejercicios en seudocódigo.

Cosas que debemos de saber:

- Para declarar una variable en seudocódigo debemos usar la palabra **DEFINIR**.
- Debemos de colocar a lado de la variable el tipo de dato que va almacenar la variable y siempre en cualquier línea de código que escribas vas a colocar el “;” (punto y coma).

```
Proceso programaEdad
    Definir edad Como Entero;
```

Aquí acabamos de definir una variable.

Pero, ¿cómo continuaría el programa si quisiera saber la edad del usuario?

Bueno otras cosas que debemos de saber es:

- La palabra **ESCRIBIR** nos servirá para hacer visualizar un texto al usuario (recordemos el punto y coma al final) además de que para escribir en texto siempre debemos de usar “”.
- Para que el usuario pueda ingresar su edad desde el programa usaremos la palabra **LEER** y el nombre de la variable que almacenara ese dato.

```
Proceso programaEdad
    Definir edad Como Entero;
    Escribir "Ingrese su edad";
    Leer edad;
    Escribir "Usted tiene ", edad;
FinProceso
```

Así se vería al final el programa. Le agregamos un plus para formular una respuesta más completa.

MAS EJERCICIOS:

Créame un programa en seudocódigo que haga lo siguiente:

- Pida un nombre al usuario.
- Pida un numero par.
- Pida una calificación.
- Pida un color.

El siguiente paso es conocer los **OPERADORES**. Pero, **¿que son los operadores?**

Los operadores nos permiten realizar operaciones dentro del programa. Por el momento conoceremos 3.

- Operadores aritméticos
- Operadores relacionales
- Operadores lógicos

OPERADORES ARITMETICOS: Nos permite realizar operaciones matemáticas.

OPERADOR ARITMETICO	SIMBOLO	EJEMPLO
Potencia	**^**	**5^2 =**
Suma	**+**	**5+2 =**
Resta	**-**	**5-2 =**
Multiplicación	*****	**5*2 =**

División	**/**	**5/2 =**
Modulo/Residuo	**%, MOD**	**5%2 =**
División baja - devuelve el entero de la división	**//**	**5//2=**

OPERADORES RELACIONALES: Compara dos valores y de esta comparación nos da un valor lógico.

OPERADOR RELACIONAL	SIMBOLO	EJEMPLO
Mayor que	**>**	**6>3 true**
Menor que	**<**	**13<11 false**
Mayor o igual que	**>=**	**6>=6 true**
Menor o igual que	**<=**	**4<=3 false**
Diferente	**<>, !=**	**60!=40 true**
Igual	**=,==**	**9==10 false**

OPERADORES LOGICOS: Establecen relaciones entre valores lógicos.

OPERADOR LOGICO	SIMBOLO
AND (conjunción)	Y / &&
OR (disyunción)	O / ||
NOT (negación)	NO / =

OPERADOR LOGICO “AND”:

Recordemos que este operador solo nos devolverá un valor lógico como **TRUE** o **FALSE**.

Cuando tengamos el operador **AND** necesita que las 2 condiciones se cumplan para que la proposición ser verdadera.

Por ejemplo, se representaría algo así en la vida real.

Te pasare a buscar a las 7 y te llevare flores.

Para que esta condición sea verdadera necesita ser verdad que me pasara buscar a las 7 y me llevara flores.

TABLA DE VERDAD CON AND:

TRUE	**&**	**TRUE**	**= TRUE**
FALSE	&	TRUE	= FALSE
TRUE	&	FALSE	= FALSE
FALSE	&	FALSE	= FALSE

OPERADOR LOGICO "OR":

Para que esta proposición se cumpla debe ser al menos una verdad. Por ejemplo, sigamos con el mismo ejemplo de la vez anterior.

Te pasare a buscar a las 7 o te llevare flores.

Debe al menos cumplir una de las 2 condiciones. O venirme a buscar a las 7 o traerme flores.

TABLA DE VALOR CON OR:

TRUE	**||**	**TRUE**	**= TRUE**
FALSE	**||**	**TRUE**	**= TRUE**
TRUE	**||**	**FALSE**	**= TRUE**
FALSE	||	FALSE	= FALSE

OPERADOR LOGICO "NOT":

En este caso con el operador NOT estamos negando la condición. La negación de una proposición es verdadera cuando dicha proposición es falsa y viceversa.

Continuando con el ejemplo seria así:

No **te pasare a buscar a las 7**.

Es falsa, pero si decimos:

No **no te pasare a buscar a las 7**.

En este caso sería verdadera.

TABLA DE VALOR "NOT"

NOT	TRUE	FALSE
NOT	**FALSE**	**TRUE**

Bueno ahora pongamos en práctica lo aprendido hasta ahora.

Crearemos un programa que pida al usuario 2 números y este lo sume y muestre el resultado al usuario.

Aclaración:
Si vamos a trabajar con más variables que compartan el tipo de dato que van almacenar en seudocódigo debemos de sepáralos por ";" (comas)

```
Proceso sumar
    Definir num1, num2, resultado Como Entero;
    Escribir "Ingrese el primer numero";
    Leer num1;
    Escribir "Ingrese el segundo numero";
    Leer num2;
    resultado ← num1 + num2;
    Escribir "Total: ", resultado;
FinProceso
```

¿Qué fue lo primero que hicimos?

1. Definir la variable.
2. Pedir al usuario que ingrese un numero
3. Guardar ese número dentro de la variable **num1**.
4. Nuevamente pedimos al usuario ingresar un número.
5. Esta vez este número lo almacenamos en la variable **num2**.
6. Luego usamos la variable **resultado** para guardar el resultado de la suma entre **num1** y **num2**.
7. Por último, le mostramos el usuario el resultado final de la suma.

Bueno ahora es tu turno de hacerlo. Crea lo siguiente:

- Crea un programa que ahora en lugar de sumar los números los reste.
- Crea un programa donde el usuario ingrese 2 números y este diga si el primer número que ingreso es mayor o no.
- Crea un programa donde ingrese 3 calificaciones el usuario y este las sume y luego las divida para sacar el promedio final.

RESPUESTAS:

```
Proceso restar
    Definir num1, num2, resultado Como Entero;
    Escribir "Ingrese el primer numero";
    Leer num1;
    Escribir "Ingrese el segundo numero";
    Leer num2;
    resultado ← num1 - num2;
    Escribir "Total: ", resultado;
FinProceso
```

Explicación:

1. Definir la variable.
2. Pedir al usuario que ingrese un numero
3. Guardar ese número dentro de la variable **num1**.
4. Nuevamente pedimos al usuario ingresar un número.
5. Esta vez este número lo almacenamos en la variable **num2**.
6. Luego usamos la variable **resultado** para guardar el resultado de la resta entre **num1** y **num2**.
7. Por último, le mostramos el usuario el resultado final de la resta.

```
Proceso esMayor
    Definir num1, num2, resultado Como Entero;
    Escribir "Ingrese el primer numero";
    Leer num1;
    Escribir "Ingrese el segundo numero";
    Leer num2;
    Escribir "Es: ", num1 > num2;
FinProceso
```

Explicación:

1. Definir la variable.
2. Pedir al usuario que ingrese un numero
3. Guardar ese número dentro de la variable **num1**.
4. Nuevamente pedimos al usuario ingresar un número.
5. Esta vez este número lo almacenamos en la variable **num2**.
6. Para mostrarle si es verdad o no el usuario usamos el operador de comparación (>).

```
Proceso calificaciones
    Definir nota, nota2, nota3, resultado  Como Real;
    Escribir "Ingrese la primera nota";
    Leer nota;
    Escribir "Ingrese la segunda nota";
    Leer nota2;
    Escribir "Ingrese la tercera nota";
    Leer nota3;
    resultado ← (nota + nota2 + nota3) / 3;
    Escribir "Promedio final: ", resultado;
FinProceso
```

Explicación:

1. Definimos las variables como un dato de tipo real (porque a veces podemos tener notas como 3.4 y así).
2. Luego le pedimos al usuario ingresar su primera nota y la guardamos en variable nota. Continuamos así hasta ya tener las 3 notas guardadas en sus respectivas variables.
3. Como podemos ver lo que hice fue almacenar el resultado en la variable correspondiente, pero para sumar las notas las encerré entre (). Esto lo hice así porque si no le agrego el paréntesis lo que va hacer el programa es resolver primero la división y luego la suma lo que me dará un resultado incorrecto, pero si agrego los paréntesis estoy dando prioridad a que se resuelva primero lo que está en paréntesis y luego lo demás.
4. Y para finalizar le muestro al usuario su promedio final.

Jerarquía de operaciones:

1. Paréntesis
2. Exponentes
3. Multiplicación, división, modulo
4. Suma y resta

CONCEPTOS BASICOS

Para continuar el curso necesitamos tener en cuenta los siguientes conceptos:

Bueno ya sabemos que es programar, que es un algoritmo y que es un diagrama de flujo, pero veamos qué es lo que engloba todo esto…

¿Qué es el Software?

Es un conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

¿Y qué es el Hardware?

Son las partes físicas, tangibles, de un sistema informático, sus componentes eléctricos, electrónicos y electromecánicos.

¿Como se relacionan estos 2 conceptos? Bueno, tanto como el software como el hardware se necesitan para ser funcional. Ya que el software necesita un medio para ejecutarse y el hardware necesita un programa que le diga que hacer.

Anteriormente vimos que son las variables y que estas son capaces de cambiar de valor durante la ejecución del programa.

Pero existe otro tipo de variable que se las conoce como **CONSTANTES**. Estas variables cumplen la misma función que una variable solo que este valor no podrá ser modificado a lo largo del programa.

Otra cosa que vimos fueron las TABLA DE VERDAD. Pero **¿que son las tablas de verdad?**

Los valores de verdad de una proposición pueden graficarse mediante una tabla.

En la carrera de Desarrollo de Software tenemos una materia llamada **“Elementos de Matemática y Lógica”**, este nombre puede cambiar de acuerdo a la universidad, pero lo que enseña esta materia es como se introduce la matemática en el pensamiento lógico. Porque si bien para programar no hay que ser un genio de las matemáticas estas si las lleva.

Continuemos con la explicación:

Los resultados válidos en Matemática son aquellos que se pueden demostrar. La manera de hacer demostraciones depende de lo que se quiera demostrar y también de la "forma" del enunciado.

Los enunciados son **proposiciones**, se afirma algo de todos o de algunos elementos de un conjunto.

Una **proposición lógica** es una oración declarativa de la cual puede conocerse si es verdadera o falsa.

Bien recordemos el ejemplo:

Te pasare a buscar a las 7 o te llevare flores.

Esta es una proposición donde tenemos 2 condiciones que en este caso debe ser al menos una verdadera para ser verdad.

TRUE	**||**	**TRUE**	**= TRUE**
FALSE	**||**	**TRUE**	**= TRUE**
TRUE	**||**	**FALSE**	**= TRUE**
FALSE	||	FALSE	= FALSE

Veamos algunos ejercicios: De acuerdo a lo que ustedes piensan coloquen si creen que estas proposiciones son verdaderas o falsas (hagan las tablas de verdad si lo es necesario)

- **Afuera esta soleado y está lloviendo**
- **No no te iré a buscar en mi camioneta.**
- **Te comprare un paraguas o un impermeable (compro el impermeable).**
- **No me gusta el café.**

Respuestas:

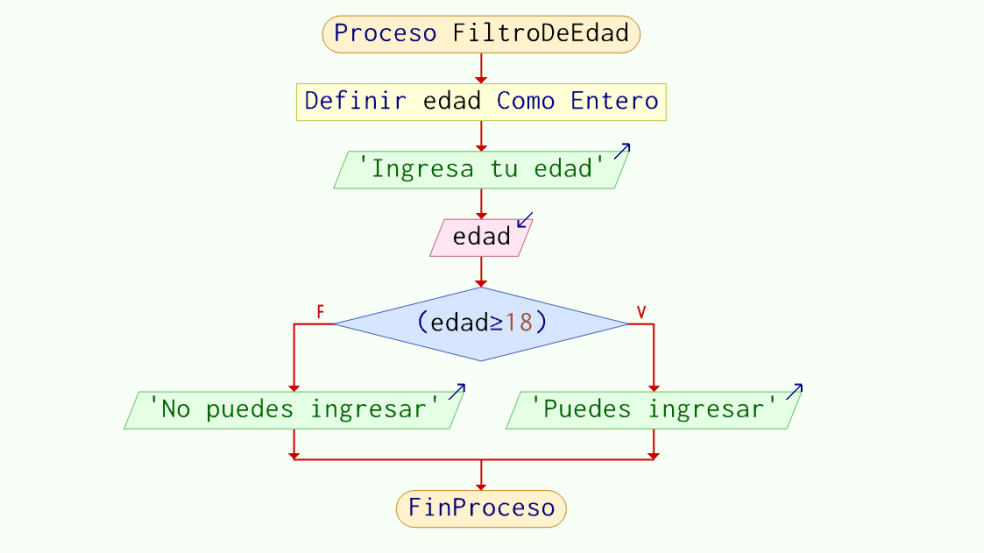
- Falso
- Verdadero
- Verdadero
- Falso

ESTRUCTURAS CONDICIONALES

Ahora si veamos las estructuras condicionales:

Las estructuras condicionales son instrucciones que permiten modificar el flujo de ejecución de las sentencias de un programa.

Recuerdan el siguiente diagrama de flujo:



Este diagrama muestra una estructura de condición. Como anteriormente explicamos este algoritmo lo que debía hacer es que de acuerdo a tu edad te iba a mostrar un mensaje de si ingresabas o no a establecimiento.

Si la persona tenía o era mayor de 18 años la condición era verdad por la que iba a tomar ese camino, pero sino era verdad iba a tomar el camino de la condición falsa.

En seudocódigo esto se escribiría así:

```
Proceso filtroDeEdad
    Definir edad Como Entero;
    Escribir "Ingrese su edad";
    Leer edad;
    si edad ≥ 18 Entonces
        Escribir "Puedes ingresar";
    SiNo
        Escribir "No puedes ingresar";
    FinSi
FinProceso
```

1. Lo primero que hicimos fue definir la variable de tipo entero.
2. Lo segundo fue mostrarle el usuario el mensaje para que sepa que debe escribir su edad.
3. Luego guardamos ese dato en la variable edad.
4. Ahora empezamos a trabajar con esto de las condiciones.

Dice: si edad >= 18 entonces (ósea si su edad es mayor o igual a 18 entonces ejecútame lo siguiente)

Si es verdad se va ejecutar el mensaje “Puedes ingresar”.

Pero SiNo, si la condición no es verdad ejecútame el siguiente mensaje.

Hagamos los siguientes ejercicios: como siempre debajo te dejare las respuestas de dichos ejercicios.

- Escribe un programa para decirme si un número es par y sino es par que mande un mensaje al usuario aclarándole que no es un numero par.
- Escribe un programa donde el usuario ingrese un número y este le diga si es un numero positivo o negativo.
- Escribe un programa para validación de usuario donde el usuario ingrese su nombre de usuario y su contraseña y si estas llegan a ser verdaderas se le dejara el ingreso y sino le enviaras un mensaje diciendo “Contraseña incorrecta”.

RESPUESTAS:

1.

```
Proceso parOimpar
    Definir num Como Entero;
    Escribir "Escribe el numero y te digo si el numero par o es impar";
    Leer num;
    si num % 2 == 0 Entonces
        Escribir "Es un numero par";
    SiNo
        Escribir "Es un numero impar";
    FinSi
FinProceso
```

2.

```
Proceso parOimpar
    Definir num Como Entero;
    Escribir "Escribe el numero y te digo si el numero par o es impar";
    Leer num;
    si num > 0 Entonces
        Escribir "Es un numero positivo";
    SiNo
        si num < 0 Entonces
            Escribir "Es un numero negativo";
        FinSi
        Si num == 0 Entonces
            Escribir "El numero es 0";
        FinSi
    FinSi
FinProceso
```

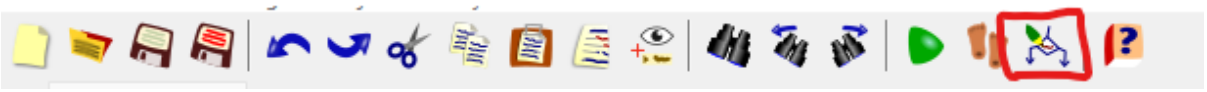
3.

```
Proceso usuar
    Definir usuario, contrasena, usuario1, contrasena1 Como Caracter;
    usuario1 ← "Rata";
    contrasena1 ← "1234";
    Escribir "Ingrese su usuario";
    Leer usuario;
    si usuario == usuario1 Entonces
        Escribir "Ingrese su contraseña";
        Leer contrasena;
        si contrasena == contrasena1 Entonces
            Escribir "Ingresando...";
        SiNo
            Escribir "Contraseña incorrecta";
        FinSi
    FinSi
FinProceso
```

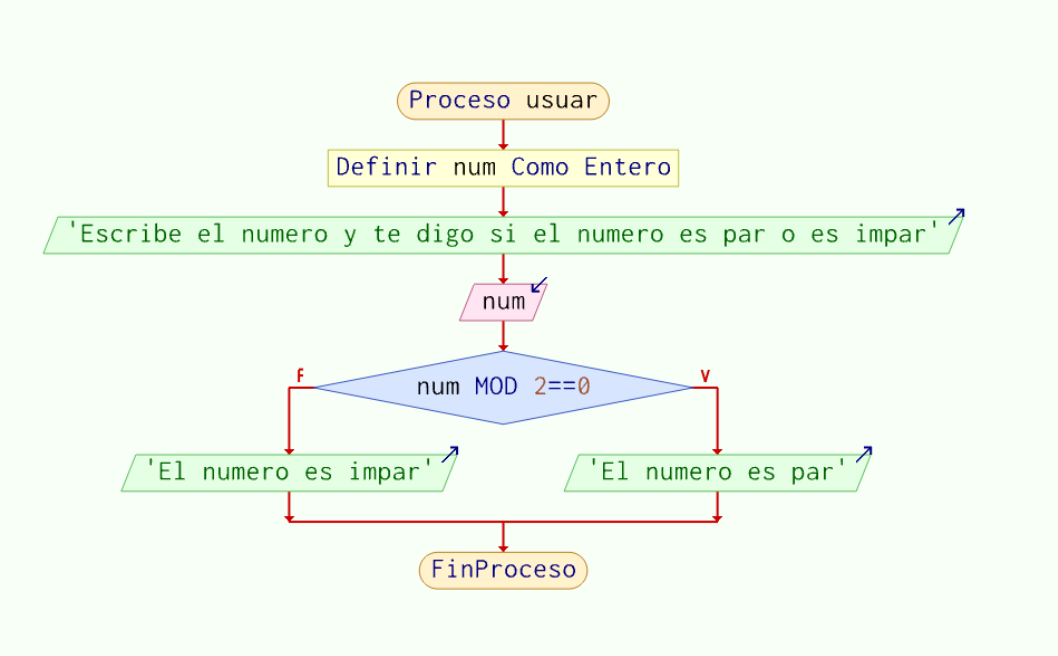
¿Qué fue lo que hicimos en este último programa?

Declaramos la variable antes de que el usuario ingrese un valor. Es decir que esa variable llamada “usuario1” no podrá ser modificada a lo largo del programa por lo que se vuelve una **variable constante**.

Revisemos ahora sus diagramas. Recuerden que pueden verlo apretando el siguiente icono:



Bueno miremos el del programa 1.



Un diagrama de flujo es una representación gráfica de un algoritmo o proceso, que

utiliza símbolos y conectores para mostrar la secuencia de pasos necesarios para

llevar a cabo una tarea o resolver un problema.

Veamos como estas figuras geométricas nos cuentan sobre el flujo del programa.

- **Inicio y Fin**: Representan el inicio y el final del proceso. Suelen representarse por círculos, óvalos u otras formas específicas.
- **Proceso**: Representa una acción o una serie de acciones que se realizan en el proceso. Se suele representar mediante un rectángulo.
- **Decisión**: Representa una condición o una pregunta que se debe evaluar para tomar una ruta u otra en el proceso. Se representa comúnmente por un rombo o un diamante y se utiliza para representar bifurcaciones en el flujo del proceso.
- **Conector**: Representa un punto de unión entre diferentes partes del diagrama de flujo.
- **Flechas**: Representan la dirección del flujo del proceso, indicando la secuencia en la que se llevan a cabo las operaciones.

Estas figuras geométricas, estas fechas, símbolos comunican al lector la secuencia de pasos.

Veamos otro tema que quedo suelto por allí “Las palabras reservadas”.

¿Que son las palabras reservadas son? Son palabras que cumplen una función en el programa. En PseInt son:

https://www.abrirllave.com/pseudocodigo/palabras-clave.php

A través de ese enlace podrá ver todas las palabras reservadas de PseInt.

Continuemos:

Anteriormente vimos cómo podemos dirigir el flujo del programa a través de funciones que nos brinda el software. ¿Pero cómo se llama lo que estábamos haciendo?

Instrucciones de control:

- Alternativas (selectivas)
- Repetitivas (iterativas)
- De salto (de transferencia)

Tipo de Instrucción alternativa:

Permite seleccionar, en el flujo de control de un programa, la o las siguientes instrucciones a ejecutar, de entre varias posibilidades.

Existen tres tipos de instrucciones alternativas:

- Doble (Si SiNo)
- Simple (Si)
- Múltiple (Según)

Tipo de Instrucción repetitivas:

Una instrucción de control repetitiva permite ejecutar una o más instrucciones varias veces.

Existen tres tipos de instrucciones repetitivas:

- Mientras
- Repetir
- Para.

Tipo de Instrucción de salto:

Son un lastre de los orígenes de la programación. De hecho, en programación estructurada, todos los programas se pueden escribir utilizando tres tipos de estructuras de control:

- Secuencial.
- De selección (alternativas).
- De iteración (repetitivas).

Así pues, todos los programas que utilizan instrucciones de salto, pueden ser reescritos sin hacer uso de ellas.

Las instrucciones de control de salto permiten realizar saltos en el flujo de control de un programa, es decir, permiten transferir el control del programa, alterando bruscamente el flujo de control del mismo.

Existen cuatro tipos de instrucciones de salto: (no están disponibles en PseInt)

- Interrumpir (romper, salir, terminar...).
- Continuar
- Ir_a
- Volver

Hagamos un par de ejercicios para probar estas funciones. (Observar la explicación de cada programa antes de ejecutar).

1. Crea un programa para un auto que **según** el color del semáforo este avance, pare o baje la velocidad.
2. Crea un programa para calcular la suma de los primeros números enteros positivos.
3. Crea un programa que muestre la tabla de multiplicar de un número n ingresado por el usuario, desde 1 hasta 10.
4. Crea un programa para contar cuántos números pares hay desde 1 hasta un número entero positivo n ingresado por el usuario.

```
Proceso coche
    Definir color Como Entero;
    Escribir "¿Cual es el color del semaforo? 1) ROJO, 2) AMARILLO, 3) VERDE";
    Leer color;
    Segun color Hacer
        1:
            Escribir "Parando el vehiculo...";
        2:
            Escribir "Bajando la velocidad...";
        3:
            Escribir "Siguiendo adelante...";
        De Otro Modo:
            Escribir "Respuesta no identificada...";
    FinSegun
FinProceso
```

¿Qué fue lo que hicimos?

1. Definimos la variable como un numero entero (porque para utilizar la función según debemos de colocar las opciones en índice).
2. Le pedimos que identifique el color de acuerdo el número y guardamos la respuesta en una variable.
3. Y SEGUN la respuesta lo iba a dirigir a un mensaje. Si su respuesta no encajaba con ninguna de las 3 (De otro Modo) le indica el siguiente mensaje “Respuesta no identificada...”

```
Proceso sumal
    Definir num, suma, i Como Entero;
    Escribir "Ingrese un numero entero positivo";
    Leer num;
    suma ← 0;
    i ← 1;

    Mientras i ≤ num Hacer
        suma ← suma + i;
        i ← i + 1;
    FinMientras
    Escribir "La suma de los primeros: ", num, " numeros enteros positivos es: ", suma;
FinProceso
```

¿Qué fue lo que hicimos?

1. Declaramos las variables como un tipo de dato entero.
2. Le pedimos al usuario ingresar un numero entero positivo.
3. Guardamos ese número en la variable num.
4. En la variable como “suma” guardamos el valor de 0 y en la variable “i” guardamos el valor 1.
5. Ejecución, MIENTRAS **i** sea menor que **num** hacer lo siguiente: En cada iteración del ciclo, se suma el valor de **i** a la variable. Después de sumar **i** a **suma**, incrementamos **i** en 1. Esto asegura que en la siguiente iteración sumemos el siguiente número.
6. El ciclo mientras termina cuando **i** es mayor que **num**. En este punto, habremos sumado todos los números desde 1 hasta **num**.
7. Escribir "La suma de los primeros ", num, " números enteros positivos es: ", suma: Muestra el resultado final al usuario. El mensaje incluye el valor de n y el resultado de la suma.

Veamos un ejemplo:

Supongamos que ingreso el número 5. Por lo que ahora nuestras variables tienen los siguientes valores:

Num = 5

I = 1

Suma = 0

Iteraciones del Ciclo Mientras:

1. Iteración: suma = 0 + 1 = 1, i = 2
2. Iteración: suma = 1 + 2 = 3, i = 3
3. Iteración: suma = 3 + 3 = 6, i = 4
4. Iteración: suma = 6 + 4 = 10, i = 5
5. Iteración: suma = 10 + 5 = 15, i = 6

El ciclo termina cuando i es mayor que 5 ósea por ejemplo 6.

Es decir, el programa nos mostrara al finalizar el siguiente mensaje: "La suma de los primeros 5 números enteros positivos es: 15".

Como vimos lo que realiza la función mientras es crear un ciclo. Pero ojo porque nosotros en este caso le colocamos una condición para que este ciclo no siga trabajando (la condición es que debe ser menor o igual en este caso que 5 si es mayor termina el programa).

```
Algoritmo TablaMultiplicarPara
    Definir n, i Como Entero;
    Escribir "Ingrese un número entero positivo:";
    Leer n;

    Para i ← 1 Hasta 10 Hacer
        Escribir n, " * ", i, " = ", n * i;
    FinPara
FinAlgoritmo
```

¿Qué fue lo que hicimos?

1. Declaramos la variable como un dato de tipo entero.
2. Pedimos que se ingrese un número y guardamos ese número en la variable n.
3. Utilizamos la función PARA decir si i es 1 increméntalo hasta llegar a 10.
4. Luego hace que n se multiplique por i.

Ejemplo de la ejecución:

Supongamos que ingreso el número 2. Por lo que me va a mostrar lo siguiente:

- 2 * 1 = 2
- 2 * 2 = 4
- 2 * 3 = 6
- 2 * 4 = 8
- 2 * 5 = 10
- 2 * 6 = 12
- 2 * 7 = 14
- 2 * 8 = 16
- 2 * 9 = 18
- 2 * 10 = 20

```
Algoritmo ContarNumerosPares
    Definir n, contador, i Como Entero;
    Escribir "Ingrese un número entero positivo:";
    Leer n;
    contador ← 0;
    i ← 1;

    Repetir
        Si i % 2 = 0 Entonces
            contador ← contador + 1;
        FinSi
        i ← i + 1;
    Hasta Que i > n

    Escribir "Hay ", contador, " números pares entre 1 y ", n;
FinAlgoritmo
```

¿Qué fue lo que hicimos?

1. Declaramos la variable como un dato de tipo entero.
2. Pedimos que se ingrese un número y guardamos ese número en la variable n.
3. Declaramos que la variable contador será 0 y que la variable i será igual a 1.
4. Luego usamos la función de “REPETIR” para decir que, si i es par, y si es así, se incrementa el contador. Luego se incrementa i en 1.
5. El ciclo se repite hasta que i sea mayor que n.

Ejemplo:

Supongamos que ingreso el número 4. Por lo que ahora nuestras variables tienen los siguientes valores:

N = 4

I = 1

Contador = 0

Iteraciones del Ciclo Mientras:

1 = impar

2 = par

3 = impar

4 = par

Salida: Hay 2 números pares entre 1 y 4.