



# Tecnicatura Universitaria en Programación

## Introducción a la Programación

### Índice:

<b>Introducción</b>	<b>2</b>
¿Qué hace un programador?	2
<b>Algoritmos</b>	<b>3</b>
¿Qué es un Algoritmo?	3
Características de un algoritmo	3
Partes de un algoritmo	3
Ejercicio 1	4
Ejercicio 2	4
Pseudocódigo	5
Diagrama de flujo	5
Ejercicio 3	7
<b>Datos</b>	<b>8</b>
¿Qué son las variables?	8
Tipos de datos	8
Definir una variable	9
Asignar valores a una variable	9
Leer valores en una variable	10
Mostrar valores almacenados en una variable	10
Tipos de datos	10
Ejercicio 4	10
<b>Operadores</b>	<b>11</b>
Operadores aritméticos	11
Operadores relacionales o de comparación	11
Operadores lógicos	12
Prioridad o precedencia entre operadores	13
Ejercicio 5	13
Ejercicio 6	13
<b>Estructuras de control</b>	<b>14</b>
Estructura de control condicional Si (if)	14
Ejercicio 7	15
Ejercicio 8	15
Ejercicio 9	15
PARA (FOR)	16

<b>Variables a utilizar dentro de una estructura de iteración</b>	<b>18</b>
Contador	18
Sumador o Acumulador	18
<b>PseInt</b>	<b>19</b>
Instalación y configuración	19
<b>Autores</b>	<b>21</b>
<b>Versiones</b>	<b>21</b>

## **Introducción**

Las computadoras se han convertido en un dispositivo esencial en la vida diaria de las personas y han cambiado el modo de vivir y de hacer negocios.

Constituyen una herramienta esencial en muchas áreas: empresa, industria, gobierno, ciencia, educación..., en realidad en casi todos los campos de nuestras vidas.

Son infinitas las aplicaciones que se pueden realizar con ellas: consultar el saldo de una cuenta corriente, retirar dinero de un banco, enviar o recibir mensajes por teléfonos celulares (móviles) que a su vez están conectados a potentes computadoras, escribir documentos, navegar por Internet, enviar y recibir correos electrónicos (e-mail), etc.

El papel de los programas o softwares es fundamental; sin una lista de instrucciones a seguir, la computadora es virtualmente inútil. Los lenguajes de programación nos permiten escribir esos programas y por consiguiente comunicarnos con las computadoras.

*La principal razón para que las personas aprendan lenguajes y técnicas de programación es utilizar la computadora como una herramienta para resolver problemas.*

Un programa de software es un conjunto de sentencias o instrucciones a la computadora. El proceso de escritura o codificación de un programa se denomina programación y las personas que se especializan en esta actividad se denominan programadores.

### **¿Qué hace un programador?**

El programador es en esencia un profesional que resuelve problemas de la realidad analizando, diseñando y codificando soluciones a esos problemas.

Para gar a ser un programador eficaz no basta con conocer un lenguaje de programación, se llenecesita aprender a resolver problemas de un modo riguroso y sistemático.

El proceso de resolución de un problema conduce a la escritura de un programa. Aunque el proceso de diseñar programas es, esencialmente, un proceso creativo, se puede considerar una serie de fases o pasos comunes, que generalmente deben seguir todos los programadores.

#### **Las fases de resolución de un problema son:**

1. Análisis del problema
2. Diseño del algoritmo
3. Codificación
4. Compilación y ejecución
5. Verificación
6. Depuración
7. Mantenimiento
8. Documentación

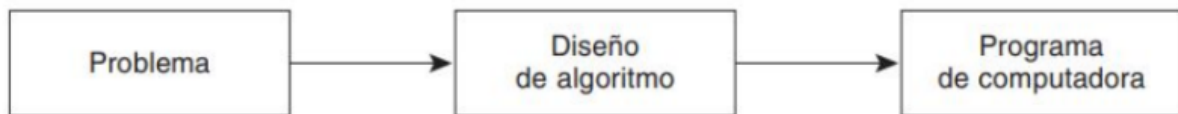
Las dos primeras fases conducen a un diseño detallado escrito en forma de algoritmo. Durante la tercera fase (codificación) se implementa el algoritmo en un código escrito en un lenguaje de programación, reflejando las ideas desarrolladas en las fases de análisis y diseño.

## Algoritmos

### ¿Qué es un Algoritmo?

Un algoritmo es un método para resolver un problema. Es decir un conjunto ordenado de pasos que permiten hallar la solución a un problema, los cuales no deben ser ambiguos.

Los algoritmos son independientes tanto del lenguaje de programación en que se expresan como de la computadora que los ejecuta. En cada problema el algoritmo se puede expresar en un lenguaje diferente de programación y ejecutarse en una computadora distinta; sin embargo, el algoritmo será siempre el mismo.



Así, por ejemplo, en una analogía con la vida diaria, una receta de un plato de cocina se puede expresar en español, inglés o francés, pero cualquiera que sea el lenguaje, los pasos para la elaboración del plato se realizarán sin importar el idioma del cocinero.

### Características de un algoritmo

- **Preciso:** Cada instrucción indica claramente lo que se debe hacer. Se debe evitar toda ambigüedad.
- **Definido:** Debe producir los mismos resultados para los mismos datos de entrada cada vez que se ejecuta.
- **Finito:** Tiene un número determinado de pasos.

### Partes de un algoritmo

1. **Entrada:** Datos que le damos al algoritmo y con la que va a trabajar para ofrecer la solución esperada.
2. **Proceso:** Conjunto de pasos para que, a partir de los datos de entrada, llegue a la solución de la situación.
3. **Salida:** Resultados, obtenidos a partir de la transformación de los valores de entrada durante el proceso.

Representaremos algoritmos a través de **diagramas de flujo o pseudocódigo**.

Veremos un ejemplo de un problema y su posible solución a través de un algoritmo.

**Problema:** Preparar un mate

*Juan quiere tomar unos mates por la mañana. Para ello llena la pava con agua y la apoya sobre la hornalla. Enciende la hornalla y mientras espera a que el agua esté lista busca el*

*termo y el mate. Prepara el mate colocando yerba hasta la mitad del mismo. Cuando escucha que la pava está lista, apaga el fuego y coloca el agua en el termo. Juan ya tiene lo necesario para tomar su primer mate por la mañana.*

El sistema para describir “escribir” un algoritmo consiste en realizar una descripción paso a paso con un lenguaje natural.

### **Algoritmo**

1. Llenar la pava con agua
2. Apoyarla en la hornalla
3. Encender la hornalla
4. Buscar el termo y mate
5. Colocar yerba hasta la mitad del mate
6. Apagar la hornalla
7. Colocar el agua caliente de la pava en el termo

### **FinAlgoritmo**

*Notar que en cada paso, se identifica sobre qué objeto se realiza la acción, es decir: no es lo mismo escribir el paso 6. “Colocar yerba hasta la mitad del mate” como “Colocar yerba” o “Colocar yerba hasta la mitad” ya que se podría prestar a conducción a que objeto hace referencia o cuanta yerba hay que colocar. **Es importante además de identificar cada paso, escribirlo con claridad.***

## **Ejercicio 1**

### **Transformar el siguiente enunciado en un algoritmo paso a paso**

*Un empleado de una tienda debe preparar un pedido para un cliente. Para ello primero lee el nombre del cliente y los artículos pedidos; luego genera un resumen del pedido. Con el resumen, busca los artículos pedidos en el depósito. Ya con todos los artículos, prepara el pedido. Una vez preparado el pedido, avisa al cliente que su pedido ya está listo.*

### **Algoritmo**

...

### **FinAlgoritmo**

## **Ejercicio 2**

### **Transformar el siguiente enunciado en un algoritmo paso a paso**

*Un empleado de facturación debe calcular la paga mensual de un empleado. Primero lee el nombre, horas trabajadas y paga x hora del empleado. Luego multiplica las horas trabajadas por la paga x hora para calcular el total por horas trabajadas. Además retiene los porcentajes de descuento por jubilación que es del 11% y por obra social que es del 5%. Una vez realizado todos los cálculos genera el recibo mensual para el empleado.*

### **Algoritmo**

...

## FinAlgoritmo

## Pseudocódigo

Mezcla de lenguaje de programación y léxico habitual en el que estamos trabajando, sin conocer lenguajes específicos.

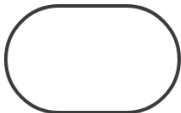



## Diagrama de flujo

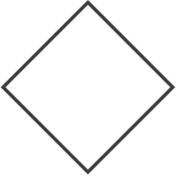
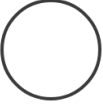

Un diagrama de flujo (*flowchart*) es una de las técnicas de representación de algoritmos más antigua y a la vez más utilizada. Describe un proceso, sistema o algoritmo informático. Se usa ampliamente en numerosos campos para documentar, estudiar, planificar, mejorar y comunicar procesos que suelen ser complejos en diagramas claros y fáciles de comprender.

Los diagramas de flujo son útiles para escribir un programa o algoritmo y explicárselo a otros o colaborar con otros en el mismo. Puedes usar un diagrama de flujo para explicar detalladamente la lógica detrás de un programa antes de empezar a codificar el proceso automatizado. Puede ayudar a organizar una perspectiva general y ofrecer una guía cuando llega el momento de codificar. Más específicamente, los diagramas de flujo pueden entre otras cosas:

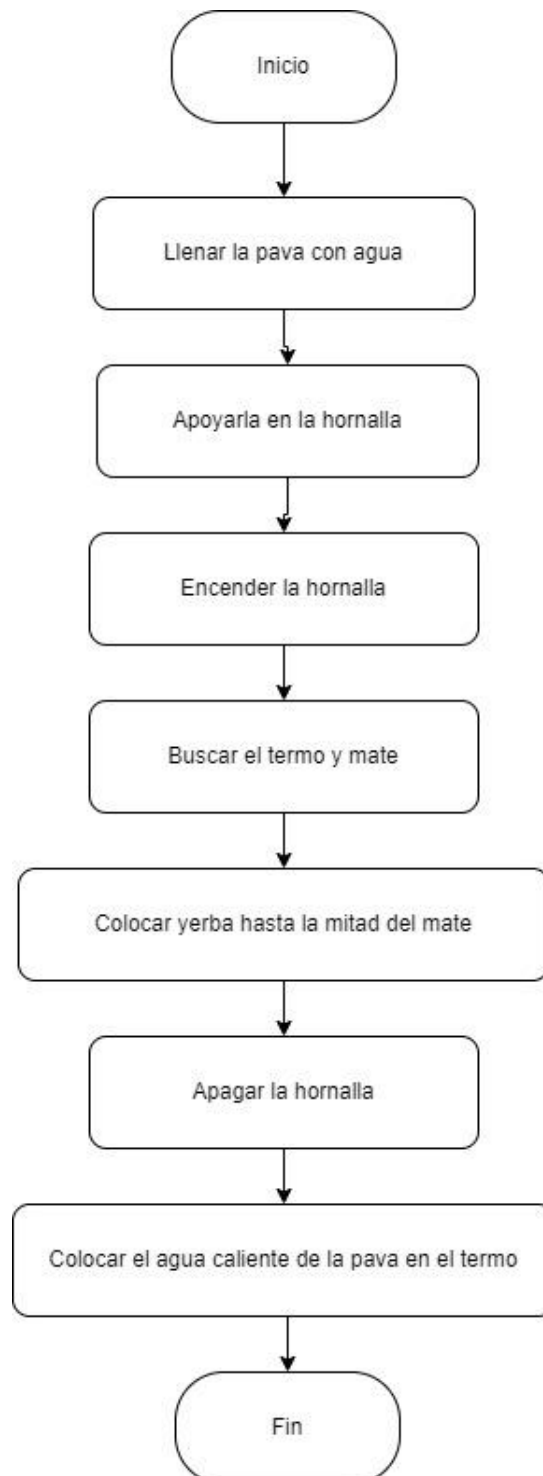
- Demostrar cómo el código está organizado.
- Visualizar la ejecución de un código dentro de un programa.

### Planilla de símbolos del diagrama de Flujo:

Símbolo	Función
	<b>Terminal:</b> representa el comienzo, “inicio”, y el final, “fin” de un algoritmo.
	<b>Proceso:</b> cualquier tipo de operación que se realiza.
	<b>Entrada/Salida:</b> cualquier tipo de “entrada” de información o “salida” de la misma.
	<b>Línea de flujo:</b> Indicador de dirección o del sentido de ejecución de los pasos.

	<b>Decisión:</b> indica operaciones lógicas o de comparación entre datos —normalmente dos— y en función del resultado de la misma determina cuál de los distintos caminos alternativos del programa se debe seguir.
	<b>Conector:</b> sirve para enlazar dos partes cualesquiera de un diagrama a través de un conector en la salida y otro conector en la entrada. Se refiere a la conexión en la misma página del diagrama.
	<b>Conector:</b> Conexión entre dos puntos del organigrama situado en páginas diferentes.

Del ejemplo para preparar un mate tenemos el siguiente diagrama de flujo:



### **Ejercicio 3**

**Enunciado:** Realizar el diagrama de flujo de los pasos escritos del ejercicio 2.

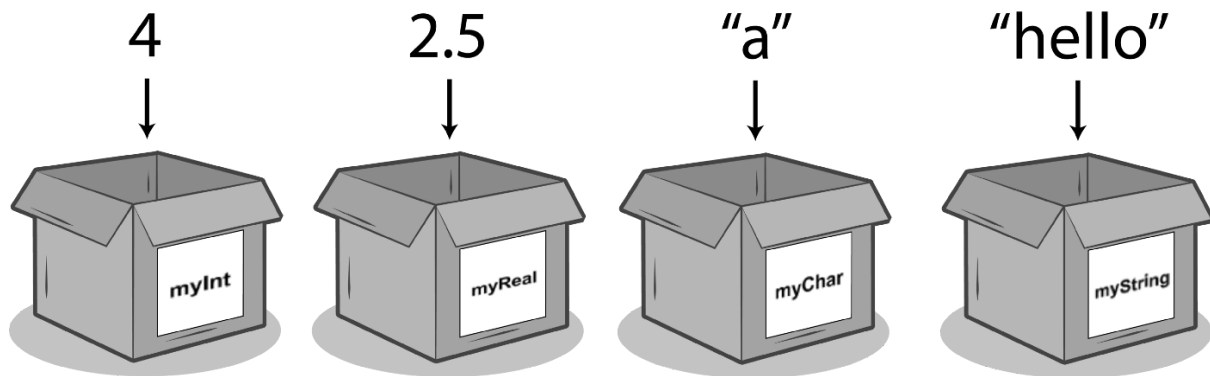


## Datos

### ¿Qué son las variables?

Una variable es un **nombre o identificador** asociado con un **dato** cuyo valor puede **cambiar** durante el inicio o fin de un algoritmo o la ejecución de un programa. Las variables pueden tener también definido un **tipo de dato** que indica los valores que puede asumir dicha variable.

Podemos pensar en las variables como “cajones” (espacios de la memoria) donde podemos guardar valores, y a su vez estos cajones pueden cambiar su contenido a lo largo del tiempo.



### Tipos de datos

Imaginemos que estamos trabajando en un programa de cobros de una cervecería y debemos trabajar con los siguientes datos:

- La cantidad de pintas de cerveza que compró.
- El precio de la pinta.
- El nombre del mozo que lo atendió.

Primero podemos identificar a estos datos con los siguientes nombres de variables:

- La cantidad de pintas de cerveza que compró → **cantidadPintas**
- El precio de la pinta → **precioPinta**
- El nombre del vendedor que lo atendió → **nombreMozo**

Notar que los nombres o identificadores para las variables se escriben sin espacios entre las palabras que conforman dicho nombre y además identifica bien el dato que representa.

**cantidadPintas** es sólo una de las muchas formas que hay de escribir nombre de variables; otras alternativas son **cantidad\_pintas** o **CantidadPintas**.

Luego de elegidos los nombres de las variables para los datos de mi programa debemos identificar el tipo de datos de dichas variables.

La cantidad de pintas siempre va a ser un número entero (ya que imaginemos que la cervecería no tiene permitido vender media pinta, solo la pinta completa).

El precio de la pinta, al ser expresado en moneda puede ser un número con coma (Real).

Y el nombre del mozo es un texto.

Tenemos los siguientes tipo de datos para las variables definidas

***cantidadPintas*** → **entero**

***precioPinta*** → **real**

***nombreMozo*** → **texto**

Si el cliente compró 6 pintas a \$8,50 y el nombre del mozo que lo atendió es Gabriel Santos, las variables asumirán los valores:

***cantidadPintas*** = 6

***precioPinta*** = 8.5

***nombreMozo*** = “Gabriel Santos”

En otro caso, el cliente compró 2 pintas a \$8,50 y el nombre del mozo que lo atendió es Julio Perez, las variables asumirán los valores:

***cantidadPintas*** = 2

***precioPinta*** = 8.5

***nombreMozo*** = “Julio Perez”

¿Qué debería hacer si quiero saber cuanto es el valor total del ticket para el cliente? Sencillamente puedo utilizar los valores guardados en las variables ya declaradas (es decir, ya creadas) para realizar la operación.

***cantidadPintas*** \* ***precioPinta***

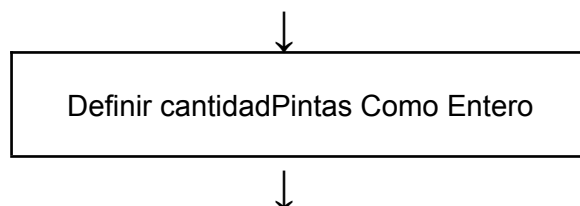
Este valor lo puedo guardar en una **nueva variable** llamada ***totalTicket*** que reciba el valor obtenido de la operación anterior.

***total Ticket*** = ***cantidadPintas*** \* ***precioPinta***

¿Cuál será el tipo de datos de la variable ***totalTicket*** , teniendo en cuenta los tipos de datos de las variables involucradas en la operación?

### **Definir una variable**

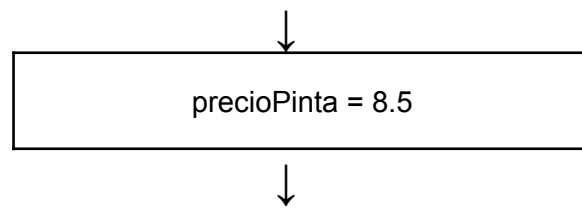
En resumen para definir una variable tenemos que indicar su nombre o identificador y su tipo de dato utilizando la instrucción **Definir** como sigue:



Una vez definida la variable podemos utilizarla en el resto de nuestro algoritmo o programa, almacenando en ella valores.

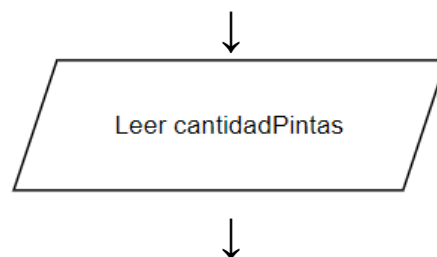
### **Asignar valores a una variable**

Asignar un valor a una variable sería lo mismo que guardar dicho valor. Para ello vamos a utilizar el = seguido del valor a almacenar como sigue:



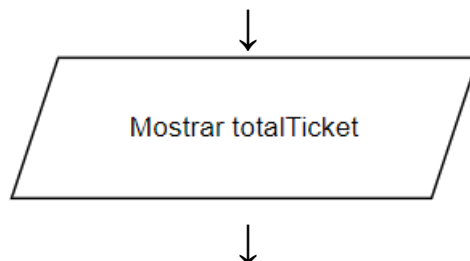
### **Leer valores en una variable**

Hay veces que al momento de escribir el algoritmo no sabemos cuál va a ser el valor que debe almacenar una variable, en ese caso en vez de asignar debemos “Leer” el valor de dicha variable. Leer significa que se ingresa al algoritmo un valor y se almacena en la variable indicada como sigue:



### **Mostrar valores almacenados en una variable**

También necesitamos saber qué valor tiene almacenado una variable y para ello debemos mostrar su valor. Para ello utilizamos las palabras Mostrar o Escribir seguido del nombre de la variable como sigue:



### **Tipos de datos**

Los tipos de datos para variables que vamos a ver son:

- **Entero:** Subconjunto de los números enteros.
- **Real:** Subconjunto de los números reales.
- **Texto o Cadena:** Cualquier conjunto de caracteres alfanuméricos.
- **Lógico o Booleano:** verdadero o falso.

## **Ejercicio 4**

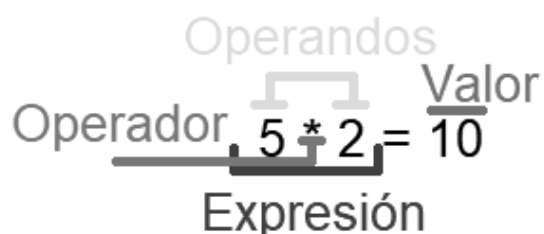
**Escribir el algoritmo para el siguiente enunciado en pseudocódigo y luego realizar el diagrama de flujo.**

*Un mozo de una cervecería toma la orden y posterior cobro del ticket a un cliente. Primero lee la cantidad de pintas que el cliente solicita. Las pintas valen todas \$10. Luego genera el ticket con el total a abonar por el cliente. El total es la multiplicación del precio de cada pinta por la cantidad ordenada. El mozo se llama Maximiliano Arco.*

## **Operadores**

Un operador es un símbolo que determina la acción que se va a realizar sobre los operandos en una expresión. Por otro lado, una expresión, aunque parezca redundante, es un conjunto de operandos y operadores que se evalúa para determinar su valor.

Veamos el siguiente ejemplo utilizando operadores aritméticos o matemáticos que ya conocemos.



La expresión anterior,  $5 * 2$ , está compuesta por los operandos 5 y 2 y el operador \*, el cual indica que sobre los operandos se debe realizar una multiplicación para obtener el valor de la expresión.

Los operandos de una expresión pueden ser números, variables o otras expresiones.

Existen diferentes tipos de operadores, a continuación se desarrollan los operadores aritméticos, lógicos y relacionales.

## **Operadores aritméticos**

Son los más conocidos porque se usan en matemática, ya que permiten realizar cálculos aritméticos sobre los operandos. En este curso, los vamos a utilizar solamente con números o variables numéricas enteras o reales.

**Los operadores aritméticos a utilizar son:**

- + (suma)
- (resta)
- \* (multiplicación)
- / (división)

## **Operadores relacionales o de comparación**

Se utilizan para comparar o relacionar operandos del mismo tipo y el resultado de esta comparación puede ser verdadero o falso, es decir un resultado de tipo lógico o booleano.

**Los operadores relacionales que vamos a utilizar son:**

- < (menor)
- > (mayor)
- <= (menor o igual)
- >= (mayor o igual)
- <>/!= (distinto)
- == (igual)

**Ejemplos:**

- 1 > 2 = false
- 3 == 3 = true
- "hola" == "hola" = true
- 5.1 >= 4.3 = true
- 9 <> 10 = true

## **Operadores lógicos**

Permiten establecer el valor de verdad de una expresión, pudiendo ser este valor únicamente verdadero o falso. Estos operadores se aplican sobre operandos de tipo booleano únicamente y el resultado también el de tipo de dato lógico o booleano.

**Los operadores lógicos que vamos a ver son:**

- y (and)
- o (or)
- no (not)

Se definen a través de **tablas de verdad** que indican el resultado al aplicar el operador, dependiendo del valor de cada operando.

AND		
X	Y	X and Y
True	True	True
True	False	False
False	True	False
False	False	False

OR		
X	Y	X or Y
True	True	True
True	False	True
False	True	True
False	False	False

NOT	
X	not X
True	False
False	True

**verdadero y verdadero = verdadero** => (idem) => true and true = true

**no falso = verdadero**

**falso o verdadero = verdadero**

En general, este tipo de operadores se utiliza en expresiones junto con operadores relacionales, cuyo resultado es un resultado true or false.

**Ejemplos:**

$1 > 2 \text{ AND } 2 < 5 = \text{False AND True} = \text{False}$

$3 == 3 \text{ OR } 4 <> 4 = \text{True OR False} = \text{True}$

$\text{NOT } 2 > 3 \text{ AND } 7 < 9 = \text{NOT False AND True} = \text{True AND True} = \text{True}$

## **Prioridad o precedencia entre operadores**

Cuando una expresión está formada por varios operadores, estos se deben evaluar en un orden determinado.

La precedencia en los operadores aritméticos es igual que en matemática, se evalúan de izquierda a derecha, primero la multiplicación y la división y luego la suma y la resta.

Los operadores relacionales tienen el mismo nivel de prioridad entre ellos.

La prioridad de evaluación de los operadores lógicos es NOT, AND y OR.

También existe una precedencia entre los tipos de operadores, los aritméticos se evalúan primero, luego los relacionales y por último los lógicos.

La excepción de precedencia es cuando la expresión tiene paréntesis, también al igual que en matemática. En este caso, los operadores se evalúan desde el paréntesis más interno o de mayor nivel hasta el más externo o de menor nivel.

Por ejemplo, en la expresión  $(5 + 1) * 3$  se evalúa primero  $5 + 1 = 6$  y luego  $6 * 3 = 18$  ya que la primera está entre paréntesis, independientemente de que la multiplicación tenga mayor prioridad.

## **Ejercicio 5**

**Enunciado:** Evaluar paso a paso y determinar el valor de las siguientes expresiones:

1.  $((2 + 3) == 5) \text{ OR } ((7 - 2) < 4)$
2.  $\text{NOT } ((8 / 2) == 4) \text{ AND } ((5 * 2) > 10)$
3.  $((10 - 5) < 6) \text{ OR NOT } ((20 / 4) >= 5)$
4.  $((5 * 5) > 24) \text{ AND NOT } ((30 / 6) == 5)$
5.  $((6 + 4) >= 10) \text{ OR } ((8 * 2) <= 20)$
6.  $\text{NOT } ((12 / 3) != 4) \text{ AND } ((15 - 5) > 7)$
7.  $((20 / 4) > 4) \text{ OR NOT } ((25 - 5) < 10)$
8.  $((7 * 3) >= 21) \text{ AND NOT } ((8 + 2) < 11)$
9.  $((9 + 1) > 10) \text{ OR } ((5 * 2) >= 10)$
10.  $\text{NOT } ((30 / 6) != 5) \text{ AND } ((8 * 3) > 21)$

## **Ejercicio 6**

**Enunciado:** Dada las siguientes variables y su respectivos valores resolver las expresiones:

A = 6, B = 3, C = 7, D = -10 y E = "hola"

1. A == B
2. A == B + 3
3. no C == 7
4. ((A + B) - 1) == C
5. ((A + B) - 1) >= C
6. ((A + B) < C
7. ((D + B) < C) y "hola" == E
8. no ("HOLA" == E)
9. falso o verdadero y falso o verdadero
10. no (D <> 10) y (A/B >=2) o falso
11. C + D \* -1 == B or B + D == C

## **Estructuras de control**

Las estructuras de control permiten controlar y modificar el flujo de ejecución de un programa o algoritmo.

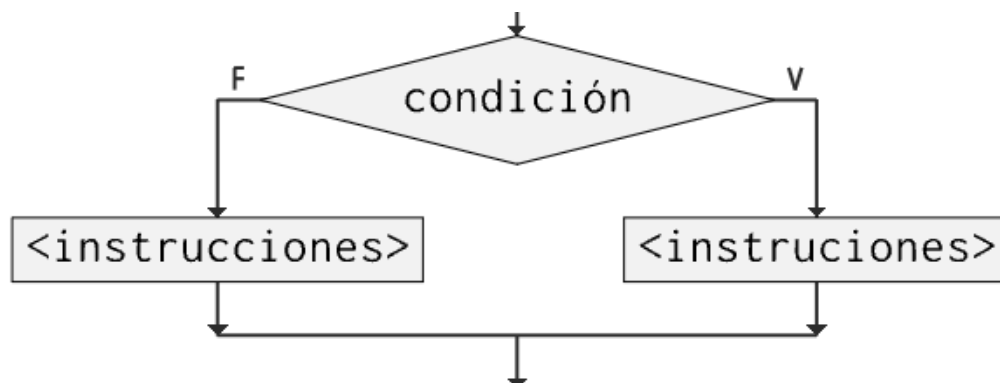
Estas estructuras serán escritas en diferente sintaxis según el lenguaje de programación que se utilice. Lo importante es conocerlas debido a que en la mayoría de los lenguajes de programación se pueden observar las mismas estructuras de control.

### **Estructura de control condicional Si (if)**

La estructura condicional **Si** o **if** se utiliza cuando un algoritmo debe tomar ciertas decisiones para realizar determinadas acciones.

Esta decisión se representa a través de la evaluación de una condición, la cual se debe determinar si la misma es verdadera o falsa. Si la condición resulta verdadera se procederá a ejecutar las sólo las instrucciones por el camino del verdadero y si la condición resulta falsa se ejecutarán sólo las condiciones por el camino del falso.

Podríamos representar la estructura if en diagrama de flujo de la siguiente manera:



En la imagen podemos observar que, luego de comenzado el programa, se debe evaluar si la condición se cumple. Dicha condición será una expresión que podría estar formada por un conjunto de operadores y variables cuyo resultado dará un valor verdadero o falso.

En el caso de que la expresión resulte verdadera, se procederá a ejecutar las instrucciones que se encuentren del lado **verdadero** del diagrama de flujo, caso contrario se ejecutarán las instrucciones del lado **falso** del mismo, ignorando aquellas instrucciones que se encuentren en el camino no elegido.

Finalmente el programa sigue su ejecución normal por fuera del bloque IF.

***Se pueden anidar múltiples estructuras condicionales IF para tomar una decisión evaluando múltiples expresiones.***

## **Ejercicio 7**

**Escribir el algoritmo para el siguiente enunciado en diagrama de flujo.**

*En la provincia de Santa Fe uno de los requisitos para que una persona pueda obtener la licencia de conducir de auto es que sea mayor a 18 años y que sepa leer. Determinar para los datos de entrada si una persona se encuentra con las condiciones suficientes para obtener una licencia de conducir en la provincia.*

## **Ejercicio 8**

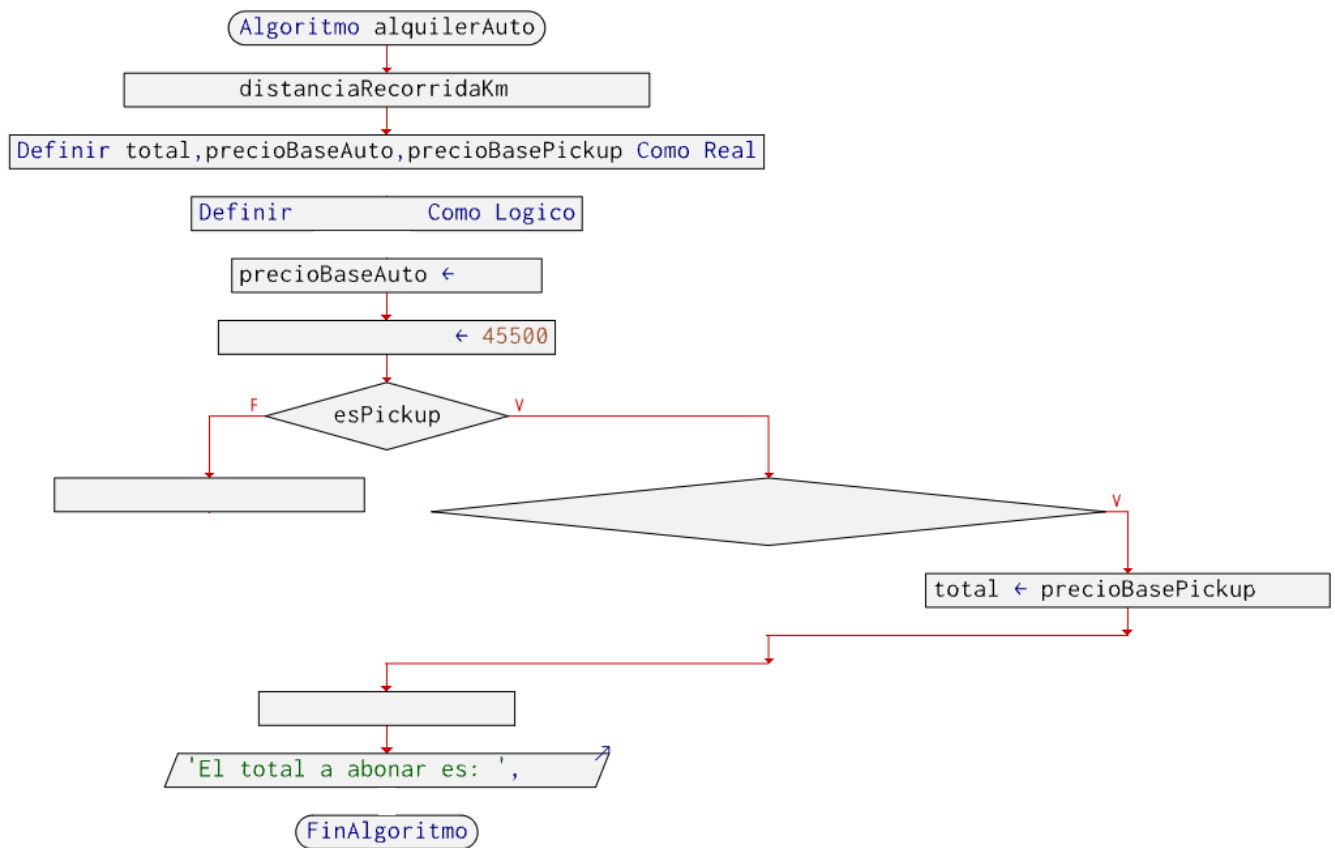
**Completar/Corregir el diagrama de flujo del siguiente enunciado**

*Una empresa de alquiler de vehículos tiene 2 tipos de vehículos: autos y pickups. Los autos tienen un precio base de alquiler de 30000 mil kilómetros y las pickups de 45000. Los vehículos se alquilan sólo por el día.  
Si un cliente alquila una pickup y recorre más de 300 kilómetros en el día se le adiciona un 25% al precio base.  
El total a abonar se muestra al cliente y el mismo incluye el IVA.*

## **Ejercicio 9**

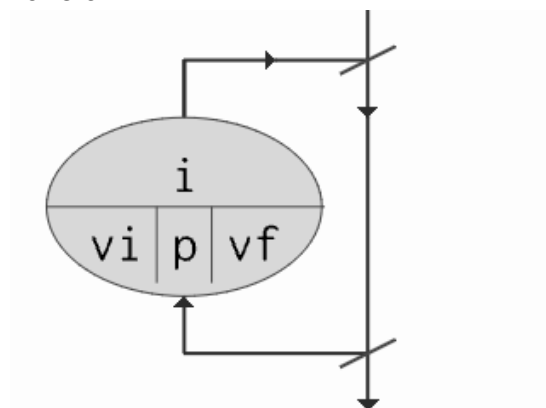
**Enunciado:** Plantee una solución más óptima para el algoritmo abajo expuesto.





## PARA (FOR)

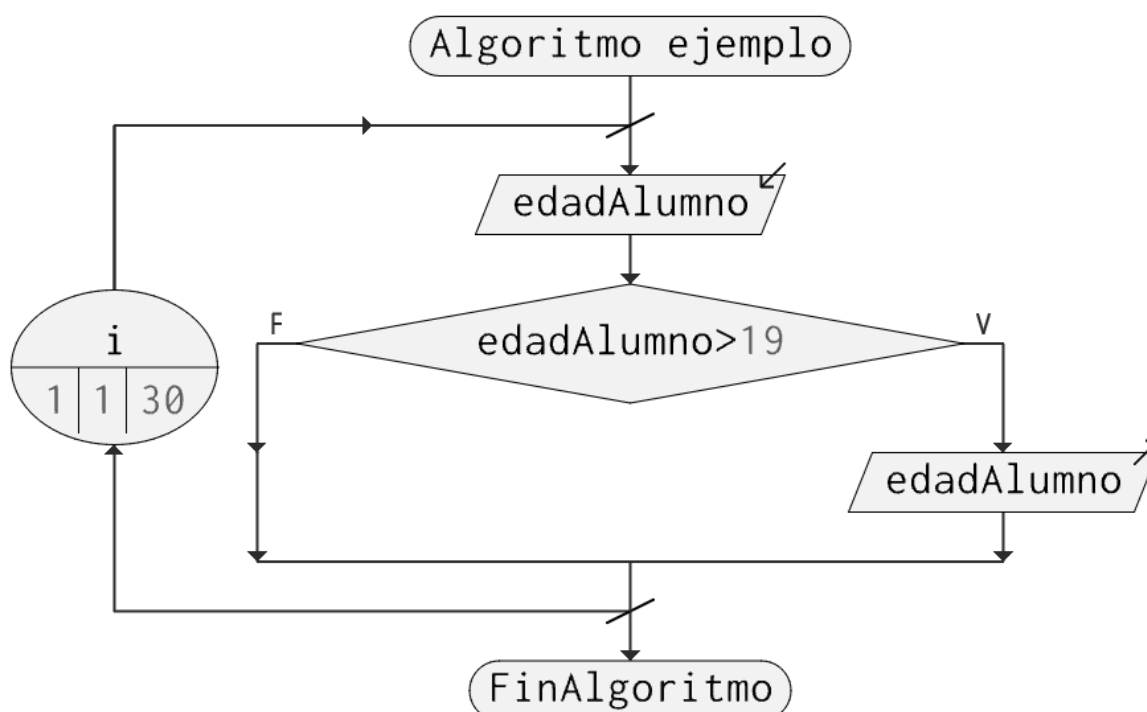
Es una estructura de control que utilizaremos para repetir un conjunto de instrucciones un número finito y conocido de veces. Para poder representarlo en el diagrama de flujo lo haremos de la siguiente manera:



En esta estructura de iteración vemos que tenemos, una variable de control que llamamos *i* que toma el valor inicial del ciclo *vi* y el ciclo se repite hasta que dicha variable llegue al valor final *vf*. La cantidad de repeticiones que realice la estructura de control depende del valor final *vf* y el incremento (paso) de la variable en cada iteración representado por *p*.

Veamos un ejemplo práctico para poder comprender mejor el concepto.

**Ejemplo:** “Se dispone de las edades de los 30 alumnos de una comisión de 1er. cuatrimestre de la TUP. Mostrar todas aquellas edades que superen los 19 años.”



La variable de control  $i$  se inicializa con el valor inicial 1 y ejecuta la acción que se encuentra dentro de la estructura de iteración (en nuestro caso la acción es una estructura de control preguntando si la edad supera o no el valor 19 establecido en el enunciado).

Cuando terminan de ejecutarse el conjunto de instrucciones dentro del bloque de iteración, se incrementa automáticamente el valor de la variable de control  $i$  en el valor indicado en el incremento (en este caso se incrementa en uno). Luego si el nuevo valor de la variable de control  $i$  no supera al valor final 30, se vuelve a ejecutar las instrucciones dentro del bloque. Sólo cuando el valor de la variable de control supera al valor final, no se vuelve más a ejecutar las acciones contenidas dentro de la estructura for y se continúa con las instrucciones que se encuentran siguiendo el flujo.

El valor de la variable de control  $i$  no debe ser modificado dentro de la iteración, pero sí puede ser utilizado con otras instrucciones siempre que no lo modifiquen.

Tener presente que no siempre es necesario que el valor inicial sea 1, el valor final sea 30 y el incremento sea 1, estos valores responden al enunciado del ejemplo.

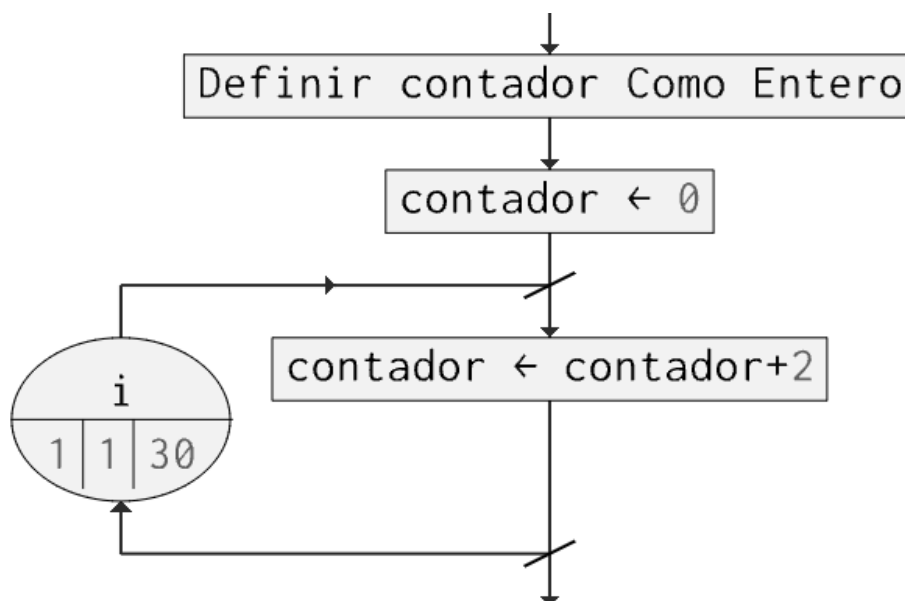
## Variables a utilizar dentro de una estructura de iteración

### Contador

Un contador es una variable cuyo valor será sobrescrito por el nuevo valor que resulte de la suma o resta entre dicho valor y un valor fijo definido previamente. El valor de la variable se verá incrementado o decrementado cada vez que se ejecute dicha expresión.

Estos son útiles para contar la cantidad de veces que se realizó una determinada acción o la cantidad de veces que ocurrió un suceso. En estos casos es común que el valor que se le suma o resta al contador sea de uno.

La variable debe ser inicializada antes de su utilización, generalmente en 0.



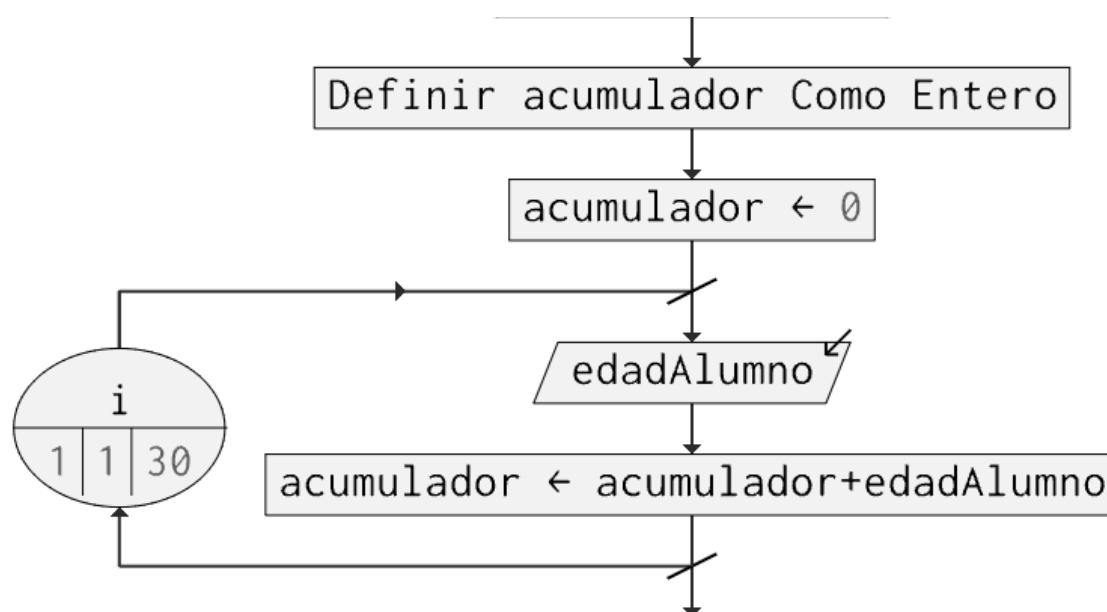
### Sumador o Acumulador

Un sumador es una variable cuyo valor será sobrescrito por el nuevo valor que resulta de la suma o resta entre dicho valor y otro valor variable. Como su nombre lo indica, suma sobre sí misma un conjunto de valores no fijos.

El sumador también debe inicializarse con un valor antes de utilizarse, generalmente en 0.

.

**La diferencia entre ambos es que el contador va aumentando su valor en una cantidad fija y preestablecida mientras que el sumador irá aumentando su valor en una cantidad variable.**



## PseInt

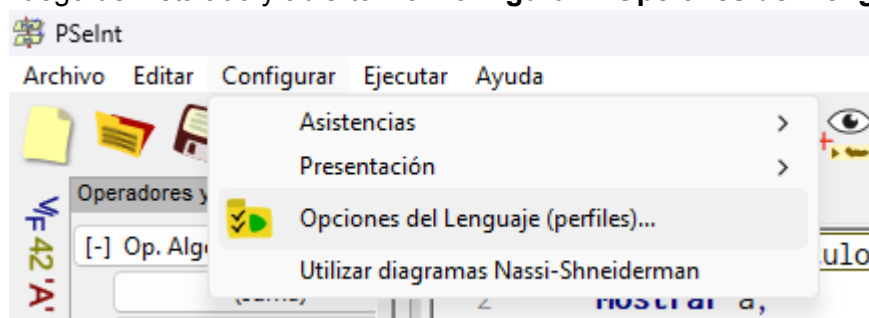
PseInt es una herramienta para asistir a un estudiante en sus primeros pasos en programación. Mediante un simple e intuitivo pseudolenguaje en español (complementado con un editor de diagramas de flujo), le permite centrar su atención en los conceptos fundamentales de la algoritmia computacional, minimizando las dificultades propias de un lenguaje y proporcionando un entorno de trabajo con numerosas ayudas y recursos didácticos.

## Instalación y configuración

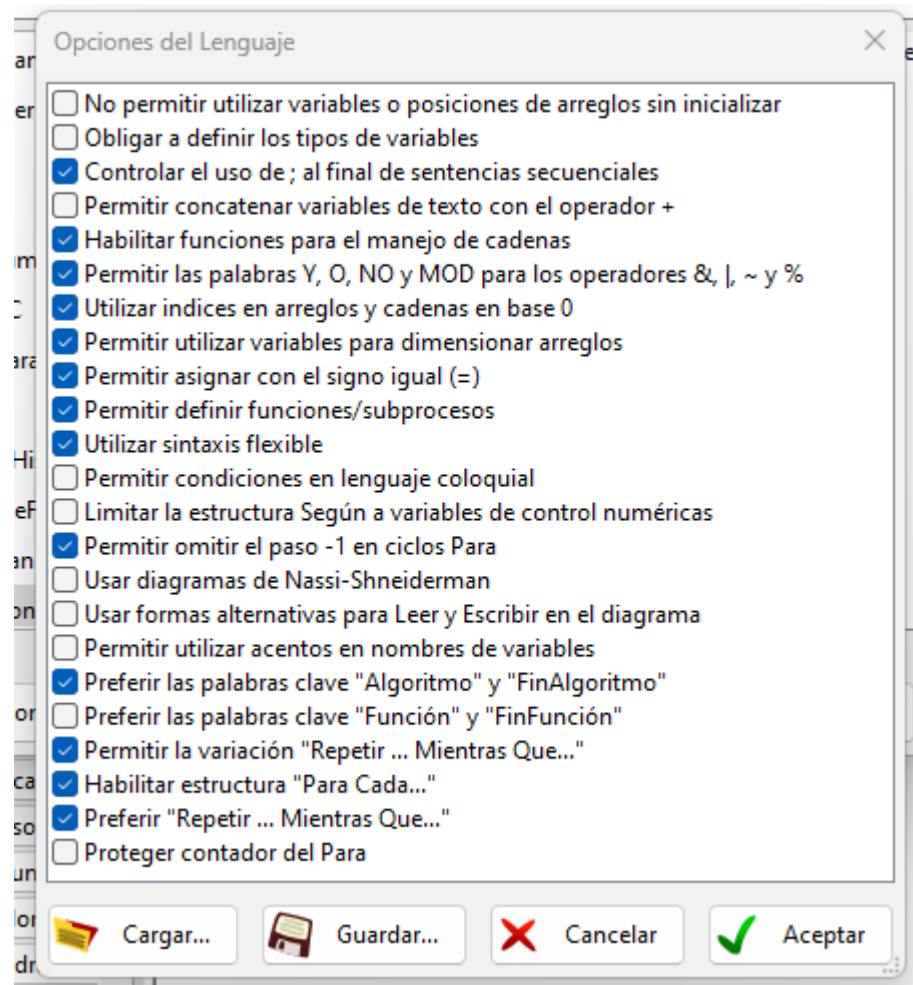
Instalar PseInt desde la web oficial, pueden optar por la versión portable que no requiere instalación. <https://pseint.sourceforge.net/>

La primera vez que abran el programa les solicitará importar un perfil, bajar el indicado en el módulo de introducción a la programación descomprimir el .zip e importar el archivo dentro.

Luego de instalado y abierto ir a **Configurar > Opciones del Lenguaje (perfiles)...**



Seleccionar **Personalizar...** y revisar que se encuentren tildadas las siguientes opciones, para ver que el perfil se importó correctamente.



## **Autores**

Gabriel Golzman
Sofía Errecarte
Natalia Fernandez
Bruno Cocitto Lopez
María Mercedes Valoni

## **Versiones**

Fecha	Versión
12/02/2023	1.0
08/03/2023	1.1 Actualizaciones
29/05/2023	2.0 Actualizaciones