

Algoritmos, Diagramas de Flujo y Pseudocódigo

Universidad Nacional de Mar del Plata
Facultad de Ingeniería

Departamento de Informática
Materia Fundamentos de la Informática

Equipo Docente:
<ul style="list-style-type: none">- Lic. Fernando Genin- Lic. Delia Esther Benchoff- Ing. Estefany Cujano- Lic. Andrea Alende- Erik Borgnia- Wenceslao Mateos
Primera edición. Version 1.3. Año 2020

Contenido

Algoritmos	4
Pasos para la resolución de un problema.....	4
Características fundamentales que debe cumplir todo algoritmo.....	5
Representaciones de algoritmos	6
Operadores.....	7
Operadores de asignación	7
Aritméticos	7
Relacionales	7
Lógicos	8
Identificadores	9
Constantes	9
Variables	9
Definición de identificadores.....	10
Tipos de datos.....	10
Consideraciones:	10
Diagrama de flujo	12
Simbología	12
Estructuras de control	17
Selectivas.....	17
Repetitivas	19
Comparación entre ciclo Mientras, ciclo Repetir...Hasta que y ciclo Para	20
Contadores y acumuladores.....	21
Buenas prácticas para la construcción de algoritmos.....	22
Documentación	22
Tipos de variables	22
Validaciones.....	24
Prueba de escritorio	27
¿Cuándo es conveniente hacer la prueba de escritorio?.....	27
Errores frecuentes	29

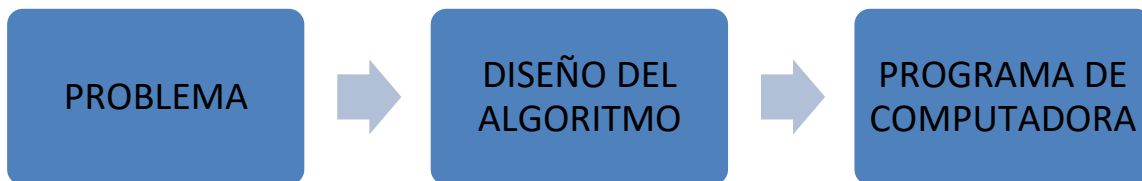
En Algoritmos representados con Diagramas de Flujo	29
En Pruebas de Escritorio	31
Ciclos o Bucles infinitos	31
Pseudocódigo	33
Sentencias	34
Comentarios	38
Errores frecuentes	38
En Algoritmos representados en Pseudocódigo	38
En la realización de la Prueba de Escritorio	39
En ejercicios de Detección de errores	39
Ciclos o Bucles infinitos	40
Reglas bien definidas en su lógica	40
Sin ambigüedades	41
Ejercicios resueltos	41
Notas	54
PSeINT:	54
Opciones del lenguaje:	54
Link de acceso a ejemplos de algoritmos representados con Pseudocódigo:	54
Bibliografía	54

Algoritmos

Un algoritmo es un método para resolver un problema¹. Aunque la popularización del término ha llegado con el advenimiento de la era informática, algoritmo proviene de Mohammed al-KhoWârizmi, matemático persa que vivió durante el siglo IX y alcanzó gran reputación por el enunciado de las reglas paso a paso para sumar, restar, multiplicar y dividir números decimales; la traducción al latín del apellido en la palabra algorismus derivó posteriormente en algoritmo.

Un algoritmo es un conjunto finito de reglas bien definidas en su lógica de control que permiten la solución de un problema en una cantidad finita de tiempo. En la resolución del problema con las reglas mencionadas, el algoritmo realiza un conjunto de pasos cuya ejecución para dar la solución del problema puede ser ejecutada manualmente, mecánicamente o utilizando una máquina de procesamiento electrónico de datos.²

La resolución de un problema exige el diseño de un algoritmo que resuelva el mismo. La propuesta para la resolución de un problema es la siguiente:



Pasos para la resolución de un problema

1. Diseño del algoritmo, describe la secuencia ordenada de pasos, sin ambigüedades, que conducen a la solución de un problema dado. (Análisis del problema y desarrollo del algoritmo).
 - a. Validación del algoritmo.
 - b. Documentación del algoritmo.
2. Expresar el algoritmo como un programa en un lenguaje de programación adecuado. (Fase de codificación).
3. Ejecución y validación del programa por computadora.

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. Por ejemplo, la receta para preparar una caipiriña de kiwi se puede expresar en español, inglés o francés, pero cualquiera que sea el lenguaje, los pasos para la elaboración, se realizarán sin importar el idioma del barman.

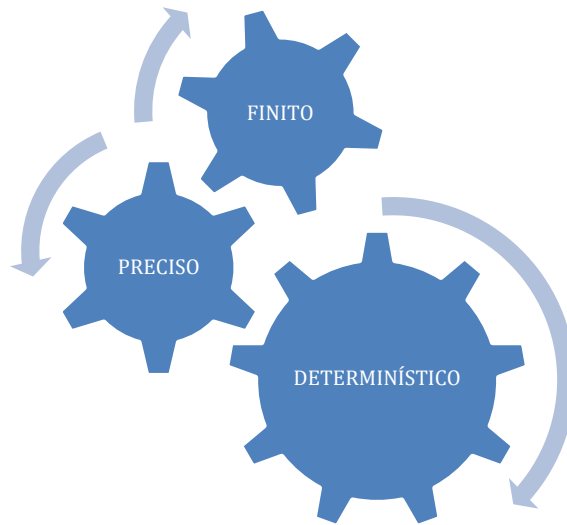
¹ JOYANES AGUILAR, Luis. Fundamentos de programación. Mc Graw Hill. España. 2003. pp. 15.

² HERRERA-GOMEZ-PORTILLA. Diseño y construcción de algoritmos. Universidad del Norte. 2016. pp. 53

Un lenguaje de programación es un medio para expresar un algoritmo y una computadora es un procesador para ejecutarlo. Tanto el lenguaje de programación como la computadora son los medios para obtener un fin: conseguir que el algoritmo se ejecute.

Características fundamentales que debe cumplir todo algoritmo

- Un algoritmo debe ser preciso e indicar el orden de realización de cada paso.
- Un algoritmo debe ser determinístico. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- Un algoritmo debe ser finito. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos.



La definición de un algoritmo debe describir tres partes: Entrada, Proceso y Salida. Por ejemplo, en el cálculo de la edad de una persona, conociendo su año de nacimiento, la definición del algoritmo, quedaría de la siguiente manera:

- **Entrada:** la edad de la persona, información del año de nacimiento y la fecha actual.
- **Proceso:** realizar la diferencia del año actual menos el año de nacimiento.
- **Salida:** visualización del resultado generado. Es decir, el resultado es la edad.

Se debe considerar que el algoritmo, que posteriormente se transformará en un programa de computadora, debe considerar las siguientes partes:

- Una descripción de los datos que serán manipulados.
- Una descripción de acciones que deben ser ejecutadas para manipular los datos.
- Los resultados que se obtendrán por la manipulación de los datos.

Representaciones de algoritmos

Un algoritmo es algo puramente conceptual que necesita una forma de representación, que permita comunicarlo a otra persona y convertirlo en un programa.

Existen diversas formas de representar un algoritmo. Vamos a enfocarnos en las siguientes:

- Diagrama de flujo
- Pseudocódigo

Operadores

Todos los símbolos que representan enlaces entre cada uno de los argumentos que intervienen en una operación se llaman operadores, y se utilizan para construir expresiones. Los operadores se clasifican:

De asignación

- Aritméticos
- Relacionales
- Lógicos

Operadores de asignación

Mediante este operador, " \leftarrow ", se almacenan valores en una variable. Estos valores pueden ser constantes o ser resultado de una expresión. La sintaxis de la operación de asignación es:

variable \leftarrow expresión

Aritméticos

Se utilizan para formar expresiones cuyo resultado será un valor numérico. Junto con las variables de tipo numérico dan lugar a las expresiones aritméticas.

- Suma (+)
- Resta (-)
- Multiplicación (*)
- División (/)
- Residuo (MOD)

Estos operadores son binarios, es decir, admiten dos operandos: uno a la izquierda y otro a la derecha, y tienen un único resultado.

Relacionales

Se utilizan para formar expresiones booleanas, es decir, expresiones que al ser evaluadas producen un valor booleano: VERDADERO o FALSO.

- Igual (=)
- Menor que (<)
- Mayor que (>)
- Menor o igual que (<=)
- Mayor o igual que (>=)
- Diferente a (<>)

Operador	Operación	Ejemplo	Resultado
=	Igual que	"A"="B"	FALSO
<	Menor que	7<15	VERDADERO
>	Mayor que	7>15	FALSO

<=	Menor o igual que	15<=22	VERDADERO
>=	Mayor o igual que	15>=22	FALSO
<>	Diferente a	"A"<>"B"	VERDADERO

Lógicos

Combinan sus operandos (proposiciones simples o compuestas) de acuerdo con las reglas del álgebra de Boole. El objetivo es producir un nuevo valor (FALSO o VERDADERO) que se convierta en el valor de la expresión. Las condiciones para control de flujo son expresiones de este tipo. Los operadores booleanos que utilizaremos básicamente son:

- Y lógico

Es también un operador binario. La expresión formada tendrá valor VERDADERO cuando ambos operandos tengan este mismo valor; en caso contrario, la expresión tendrá valor FALSO. Es el operador lógico de conjunción.

- O lógico

Es un operador binario, son necesarios dos operandos para producir un resultado. La expresión que se forma producirá un valor VERDADERO cuando al menos uno de sus operandos tenga este mismo valor; de otra forma, el valor de la expresión será FALSO.

Identificadores

Los identificadores son los nombres que se les asignan a los objetos, los cuales se pueden considerar como variables o constantes, éstos intervienen en los procesos que se realizan para la solución de un problema, por consiguiente, es necesario establecer qué características tienen.

Para establecer los nombres de los identificadores se deben respetar ciertas reglas que establecen cada uno de los lenguajes de programación, para el caso que nos ocupa se establecen de forma indistinta según el problema que se esté abordando, sin seguir regla alguna, generalmente se utilizará una o varias letras.

El nombre del identificador, debe ser representativo del objeto que se va a identificar, por ejemplo si queremos almacenar el sueldo de una persona, no sería intuitivo utilizar un identificador llamado <var1>, por el contrario sería conveniente utilizar algunos de los siguientes nombres: <s>, <sueldo>, <sueldo_empleado>.

Limitaciones: al momento de escoger el nombre del identificador, debemos tener en cuenta que el mismo no puede:

- Disponer de espacios en blanco. Por ejemplo <sueldo empleado>
- Llamarse de igual manera que una palabra reservada. Por ejemplo <int>
- Usar acentos y letras ñ. Por ejemplo <sueldo_básico>
- Empezar con un número, pero pueden finalizar con un número. Por ejemplo <num2>
- Contener símbolos tipográficos como: !, * ' ¿ ?

Constantes

Un identificador se clasifica como constante cuando el valor que se le asigna a este identificador no cambia durante la ejecución o proceso de solución del problema. Por ejemplo, en problemas donde se utiliza el valor de PI, si el lenguaje que se utiliza para codificar el programa y ejecutarlo en la computadora no lo tiene definido, entonces se puede establecer de forma constante estableciendo un identificador llamado PI y asignarle el valor correspondiente de la siguiente manera:

PI = 3,1416

Variables

Los identificadores de tipo variable son todos aquellos objetos cuyo valor cambia durante la ejecución o proceso de solución del problema. Por ejemplo, el sueldo, el pago, el descuento, etcétera, que se deben calcular con un algoritmo determinado, o en su caso, contar con el largo (L) y ancho (A) de un rectángulo que servirán para calcular y obtener su área. Como se puede ver, tanto L como A son variables que se proporcionan para que el algoritmo pueda funcionar, y no necesariamente se calculan dentro del proceso de solución.

- **Banderas:** las banderas son un tipo de variable, ya que su valor puede cambiar durante la ejecución. Es una variable que vale "verdadero" o "falso", y que típicamente se usa para realizar controles.

Definición de identificadores

Para que las variables o constantes estén correctamente definidas, hay que especificar:

- Su nombre.
- El tipo de dato: carácter, entero, real, lógico, etc.
- Su valor inicial, el cual es opcional, porque al no inicializar la variable, esta tomará el valor previamente almacenado en el conjunto de bytes que le corresponda.

Tipos de datos

Los identificadores (constantes y variables) almacenan información, la cual puede ser de distinto tipo de dato. Todos los identificadores utilizados, deben tener un tipo de dato definido.

Si bien existen múltiples tipos de datos, en este documento, solo veremos los que se detallan a continuación:

Tipo de dato	Definición	Ejemplo
Texto	Acepta cualquier tipo de caracteres.	nombre, empresa, club, bebida.
Entero	Solo acepta número enteros.	cant_alumnos, nota(*), modelo_celular
Real	Acepta números con decimales.	sueldo, nota(*), precio
Lógico	Solo acepta dos estados.	True o False

(*) la variable nota, puede ser definida como tipo de dato <entero>, para el caso que se quiera trabajar con notas enteras como por ejemplo 5, 8, 9 y también se encuentra definida como tipo de dato <real>, para el caso que se quiera trabajar con notas que tengan fracción, por ejemplo 5,5 o 6,33.

Cuando una variable se ha declarado de un cierto tipo de dato, solamente puede asignársele datos del mismo tipo. Por ejemplo, una variable que ha sido declarada de tipo entero no puede almacenar datos de tipo lógico.

Si se intenta asignar a una variable un valor de un tipo de dato que no corresponde con su declaración inicial, se produce un error de tipo.

Consideraciones:

El tipo ENTERO es una especialización que sólo permite almacenar valores enteros; cualquier valor numérico no entero que se lea o asigne en una variable de este tipo será truncado o producirá un error.

Una variable de tipo LOGICO sólo puede tomar los valores VERDADERO y FALSO, pero cuando se lee una variable ya definida como lógica, el usuario puede ingresar también las abreviaciones V y F, ó 0 y 1.

TEXTO se utiliza para definir variables de tipo carácter. Estas pueden contener cero, uno o más caracteres arbitrarios y no tienen una longitud máxima. Si se declara una variable de este tipo y en una lectura el usuario ingresa un número o un valor lógico, se asignará una cadena que contiene el texto ingresado, ejemplo: "1", "VERDADERO", etc.

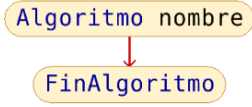

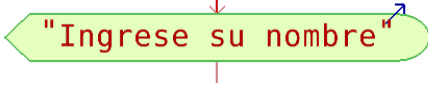
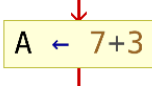
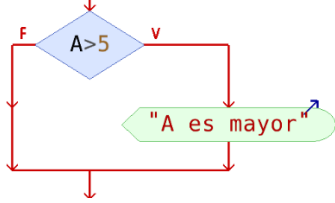
Si se intenta asignar a una variable ya definida con un tipo de dato un valor que no se corresponde con el tipo de dato definido, dependiendo del software utilizado podría producirse un error en tiempo de ejecución. Por ejemplo, ingresamos el valor "Pedro" en la variable edad definida como tipo de dato número.

Diagrama de flujo

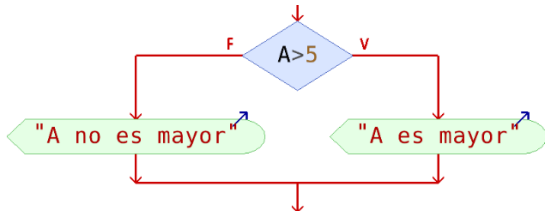
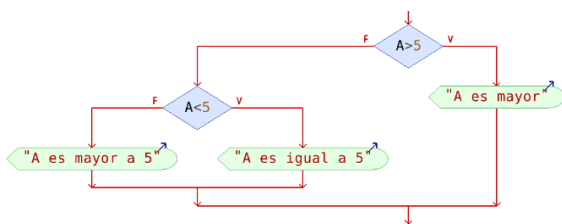
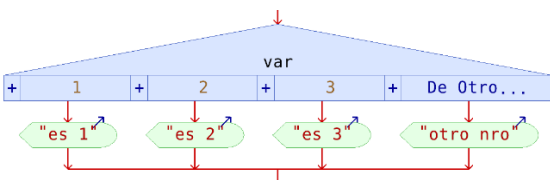
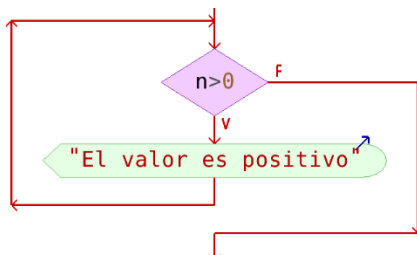
También conocido como flowchart es una técnica de programación de representación de algoritmos antigua y muy utilizada. Un diagrama de flujo, es un “un diagrama que utiliza los símbolos (cajas) estándar mostrados en la tabla 1 y que tiene los pasos de un algoritmo escritos en esas cajas unidas por flechas, denominadas líneas de flujo, que indican la secuencia en que se debe ejecutar”.³

Simbología

La simbología utilizada a continuación se corresponde con el software PSeInt.

SÍMBOLO	FUNCIÓN
	Terminal: dispone de una doble representación: <ul style="list-style-type: none"> - Inicio: representa el comienzo del algoritmo. - Fin: representa el final del algoritmo.
	Lectura: representa el ingreso de una variable por parte del usuario.
	Escritura: representa la salida de un texto o una variable por pantalla. <u>Importante:</u> si el texto que se va a mostrar tiene palabras reservadas PSeInt generará un error en tiempo de ejecución. Se recomienda colocar todo el texto entre comillas dobles o simples.
	Asignación: se utiliza para darle un valor a una variable
	Estructura selectiva simple: Se evalúa la condición y si se cumple se realizan las acciones correspondientes.

³ JOYANES AGUILAR, Luis. Fundamentos de programación. Mc Graw Hill. España. 2003. p. 59

SÍMBOLO	FUNCIÓN
 <pre> graph TD Start(()) --> Cond{A > 5} Cond -- F --> Act1[A no es mayor] Cond -- V --> Act2[A es mayor] Act1 --> Exit(()) Act2 --> Exit </pre>	<p>Estructura selectiva doble: Se evalúa la condición, si se cumple se realizan determinadas acciones y si no se cumple se realizan otras acciones diferentes.</p>
 <pre> graph TD Start(()) --> Cond1{A > 5} Cond1 -- F --> Cond2{A < 5} Cond1 -- V --> Act3[A es mayor] Cond2 -- F --> Act1[A es mayor a 5] Cond2 -- V --> Act2[A es igual a 5] Act1 --> Exit(()) Act2 --> Exit Act3 --> Exit </pre>	<p>Estructura selectiva compuesta: Se utiliza cuando se necesita evaluar más de una condición con un intervalo de valores.</p>
 <pre> graph TD Start(()) --> Cond{var} Cond -- 1 --> Act1[es 1] Cond -- 2 --> Act2[es 2] Cond -- 3 --> Act3[es 3] Cond -- De Otro... --> Act4[otro nro] Act1 --> Exit(()) Act2 --> Exit Act3 --> Exit Act4 --> Exit </pre>	<p>Estructura selectiva múltiple: Se utiliza para evaluar una variable, la cual puede tomar diferentes valores exactos, no en intervalos.</p>
 <pre> graph TD Start(()) --> Cond{n > 0} Cond -- V --> Act[El valor es positivo] Act --> Cond Cond -- F --> Exit(()) </pre>	<p>Ciclo Mientras: este ciclo se utiliza para repetir determinadas acciones mientras la condición que se está evaluando sea verdadera. Finaliza cuando la condición pasa a ser falsa.</p>

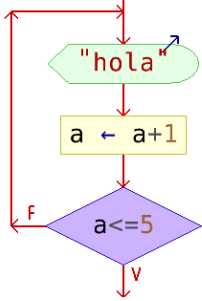
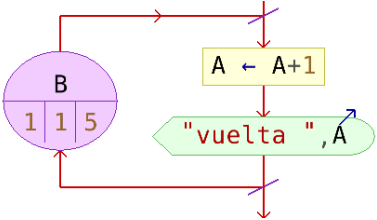
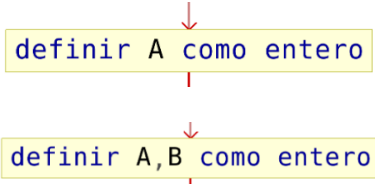
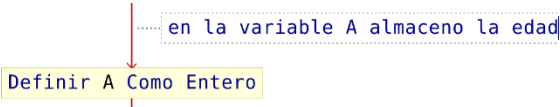
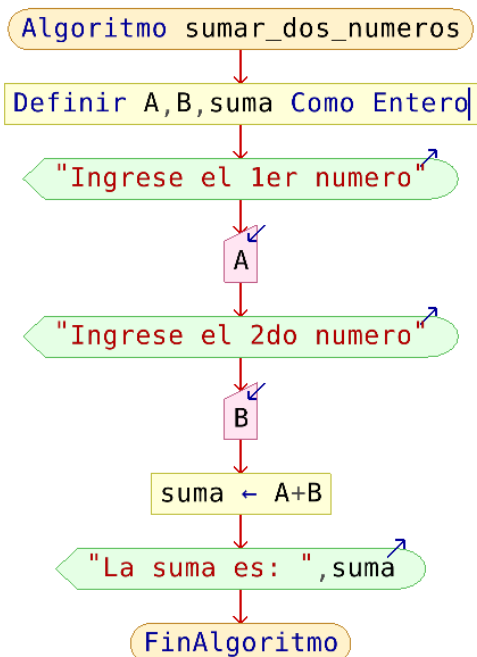
SÍMBOLO	FUNCIÓN
	Ciclo Repetir...Hasta que: este ciclo asegura que por lo menos se ejecutará una vez, y posteriormente se evaluará la condición para ver si se repite la ejecución. El ciclo finaliza cuando la condición es verdadera.
	Ciclo Para: se utiliza este ciclo cuando se sabe de antemano la cantidad de repeticiones que se van a realizar. En este caso se va a repetir 5 veces.
	Tipo de dato: Definir el tipo de dato de una variable
	Comentarios: sirve para agregar texto de explicación o ayuda, que sirve para interpretar el diagrama o pseudocódigo.

Tabla 1

Ingresar 2 números por teclado y mostrar por pantalla la suma de los mismos.



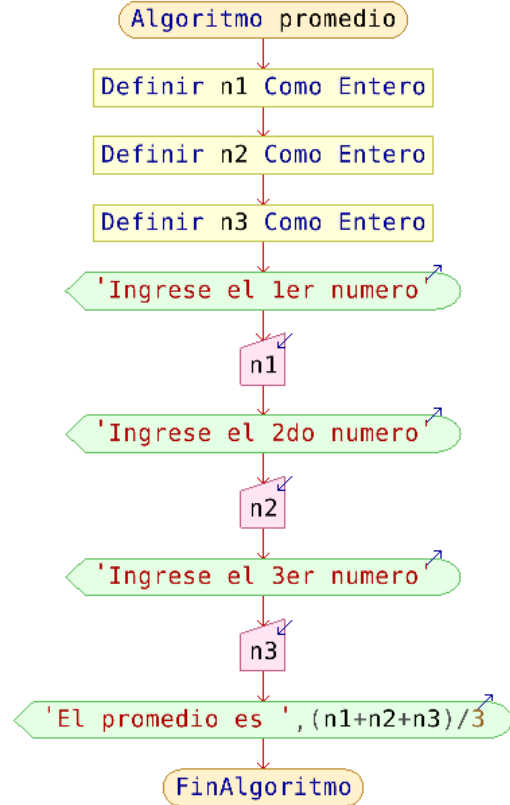
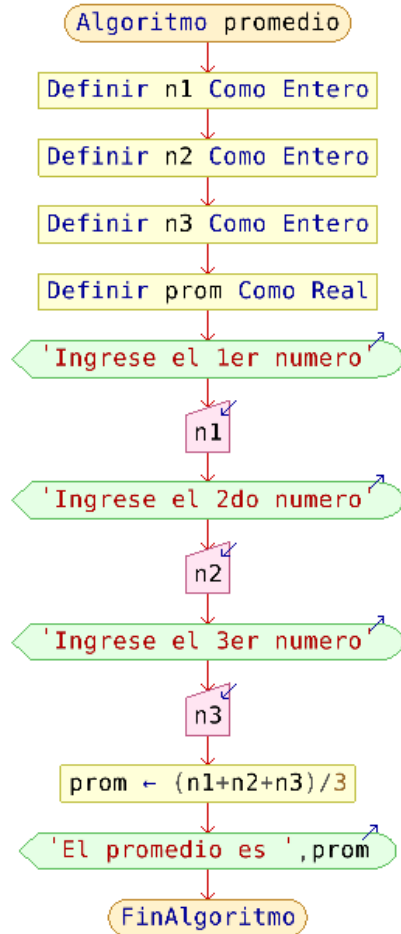
Consideraciones:

- Es necesario definir el tipo de dato de todas las variables a ser utilizadas.
- Los mensajes de salida están encerrados entre comillas dobles, debido a que la palabra 'numero' es reservada en PSeInt.
- El resultado es almacenado en la variable suma.

Actividad

- Identificar las variables utilizadas.
- Resolver el ejercicio anterior utilizando solo dos variables.

Ingresa 3 números por teclado y mostrar por pantalla el promedio de los mismos.



Actividad

- Identificar y analizar las diferencias de cada solución.
- Identificar las ventajas y desventajas de cada solución.

Estructuras de control

Las estructuras de control se clasifican en:

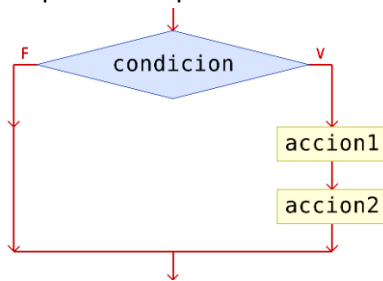
- Selectivas
 - Simples
 - Dobles
 - Compuestas
 - Múltiples
- Repetitivas
 - Mientras (While)
 - Repetir ...Hasta que (Do While)
 - Para (For)

Selectivas

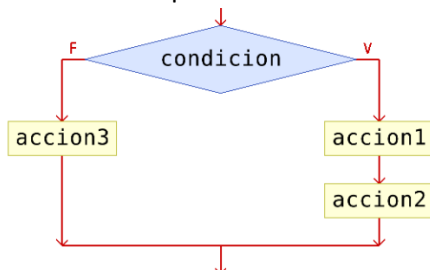
La estructura de control selectiva, se refiere a una decisión que se debe tomar en la lógica de control del algoritmo. La “decisión” implica la solución a una pregunta que se estructura en la condición. Si la resolución a la pregunta es verdadera, entonces se ejecutan las estructuras lógicas que se escriben en la condición verdadera del algoritmo.

A continuación, se describen las distintas variantes de las estructuras de control selectivas:

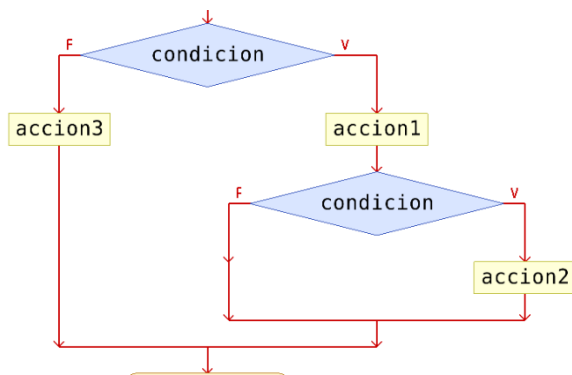
- Simples: no disponen de acciones por la opción falsa de la decisión.



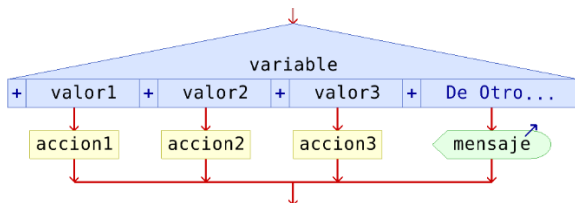
- Dobles: disponen de acciones en ambas opciones de la decisión.



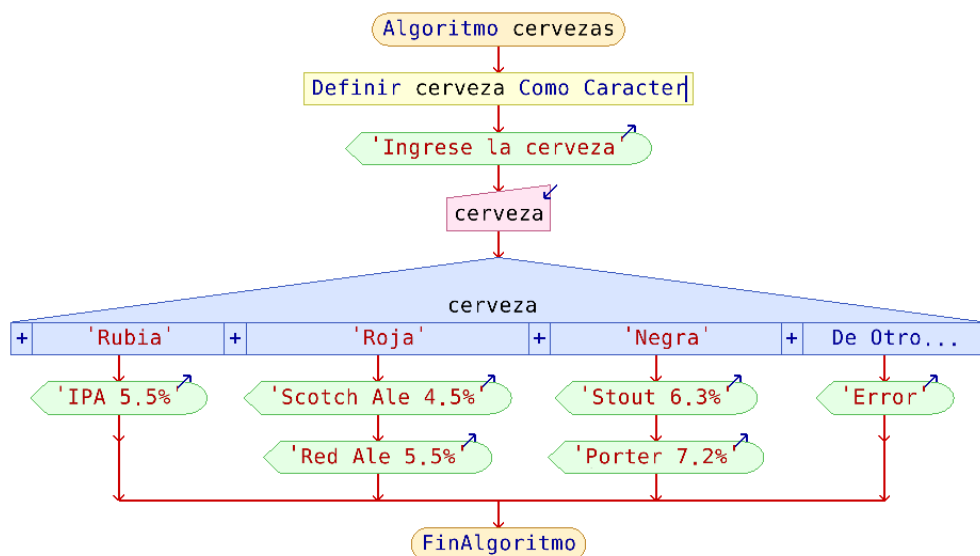
- Compuestas: se trata de aquellos casos en los cuales la opción verdadera o falsa de la decisión, dispone de una estructura de control selectiva.



- Múltiples: consiste en evaluar una expresión, en la cual **una variable** podrá tomar n valores distintos, 1, 2, 3,..., n y según el resultado de uno de estos valores en la condición, se realizará una de las n acciones, en donde el flujo del algoritmo seguirá solo un determinado camino entre los n posibles.



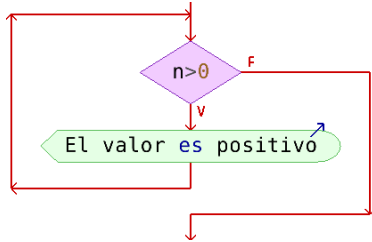
Desarrollar el algoritmo para una APP que permita mostrar la carta de cervezas disponibles en una Cervecería, en donde al ingresar se debe seleccionar el tipo (rubia, roja ó negra) de cerveza y mostrar las opciones disponibles.



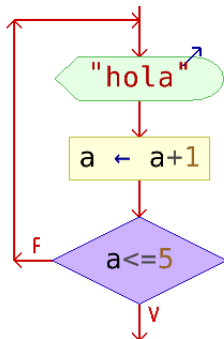
Repetitivas

- Mientras (While): permite la realización de un conjunto de sentencias, las cuales se ejecutan cuando la condición del Mientras que es verdadera. Cuando la condición del Mientras es falsa se finaliza el ciclo y se continúa con el flujo del algoritmo.

Es posible que un ciclo nunca se ejecute, debido a que la condición del Mientras es falsa en la primer iteración.

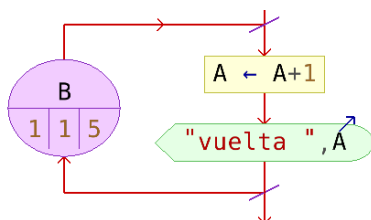


- Repetir...Hasta que (Do While): la condición del ciclo se evalúa al finalizar el mismo, motivo por lo cual las instrucciones se ejecutan por lo menos una vez. Cuando la condición del ciclo es falsa se termina el ciclo y se continúa con el flujo del algoritmo.



- Para (For):

El ciclo Para, permite la ejecución de un conjunto de sentencias de una forma repetitiva que se ubican dentro del Para, comenzando con un valor inicial, terminando en un valor final, y ejecutando el conjunto de sentencias internas del Para con un incremento de una constante o variable.



El ciclo Para está compuesto de 3 segmentos:

- Inicialización de la variable de control: es el valor inicial.
- Incremento: es un valor entero, que servirá para incrementar o decrementar el valor de la variable de control.
- Condición de control: es el valor final, que generará que el ciclo termine.

Comparación entre ciclo Mientras, ciclo Repetir...Hasta que y ciclo Para

La diferencia entre el ciclo “Mientras” y “Repetir...Hasta que”, es que independientemente de la condición de corte, el ciclo “Repetir...Hasta que” siempre se ejecutará como mínimo una vez.

Otra diferencia entre estos dos ciclos, es que en el ciclo “Mientras” la evaluación se hace al inicio, en cambio en el ciclo “Repetir...Hasta que” se realiza al final del mismo.

La diferencia entre el ciclo “Para” respecto de los ciclos “Mientras” y “Repetir...Hasta que”, es que en el primero se conoce de antemano el número de iteraciones a realizar, y en los otros dos se desconoce el número de iteraciones.

A continuación, se presenta un ejercicio práctico que es resuelto de dos formas, aplicando el ciclo “Mientras” y el ciclo “Repetir...Hasta que”.

Ingresar una serie de números y para cada uno de ellos mostrar por pantalla si el mismo es positivo o negativo. Para continuar con la carga del siguiente número se debe ingresar la letra s	Resolución aplicando el ciclo Repetir...Hasta que
<p>Resolución aplicando el ciclo Mientras</p>	<p>Resolución aplicando el ciclo Repetir...Hasta que</p>

Contadores y acumuladores

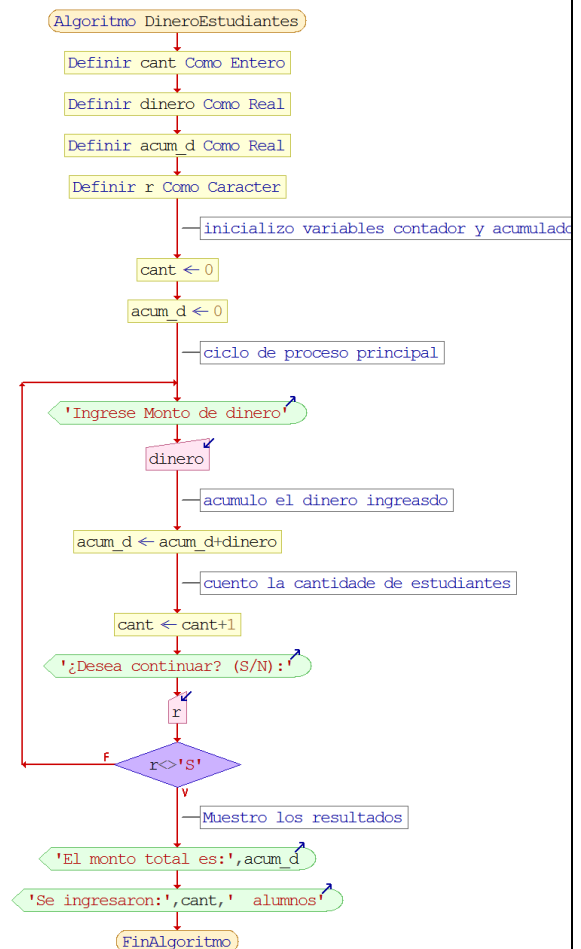
Un **contador** es una variable en memoria cuyo valor se incrementa o decrementa en un valor fijo (en cada iteración de un bucle). Un contador suele utilizarse para contar el número de veces que itera un bucle. La inicialización consiste en poner el valor inicial de la variable que representa al contador. Generalmente se inicializa con el valor cero.

Un **acumulador** es una variable en la memoria cuya misión es almacenar cantidades variables. Se utiliza para efectuar sumas sucesivas. La principal diferencia con el contador es que el incremento o decremento de cada suma es variable en lugar de constante como en el caso del contador. Al igual que los contadores, es conveniente inicializarla. Generalmente con el valor cero.

Contar la cantidad de alumnos que ingresan al aula y el monto total de dinero en sus billeteras. El ingreso continua cuando ingresa la letra S en la pregunta Desea continuar?

Ayuda:

Deberíamos disponer de un contador, que se incremente en 1 cada vez que detecta el ingreso de un alumno. También vamos a necesitar un acumulador, que sume el importe de cada billetera en una variable.



Buenas prácticas para la construcción de algoritmos

Documentación

La documentación de diagramas de flujo se utiliza para registrar las variables utilizadas en un algoritmo, las cuales se clasifican en variables de entrada, variables de proceso y variables de salida. De cada variable es necesario almacenar el nombre, la descripción y el tipo de dato.

Para documentar las variables, se utilizará una tabla como la que se muestra a continuación

NOMBRE VARIABLE	DESCRIPCION	TIPO DE DATO	TIPO DE VARIABLE		
			E	P	S
Variable1	Altura	Entero	x		
Variable2	Base	Entero	x		
Variable3	Superficie	Real		x	x
Variable4	Promedio	Real		x	

Tipos de variables

Las variables de entrada, son aquellas que se utilizan para ingresar valores en el algoritmo mediante algún dispositivo físico de entrada (teclado, mouse, etc.)

Análogamente existen las variables de salida, que se utilizan para mostrar valores por algún dispositivo físico de salida (impresora, monitor, etc.)

Las variables de proceso, son aquellas que intervienen en algún proceso dentro del algoritmo, como por ejemplo la variable <superficie> que se utiliza para almacenar la superficie de un rectángulo.

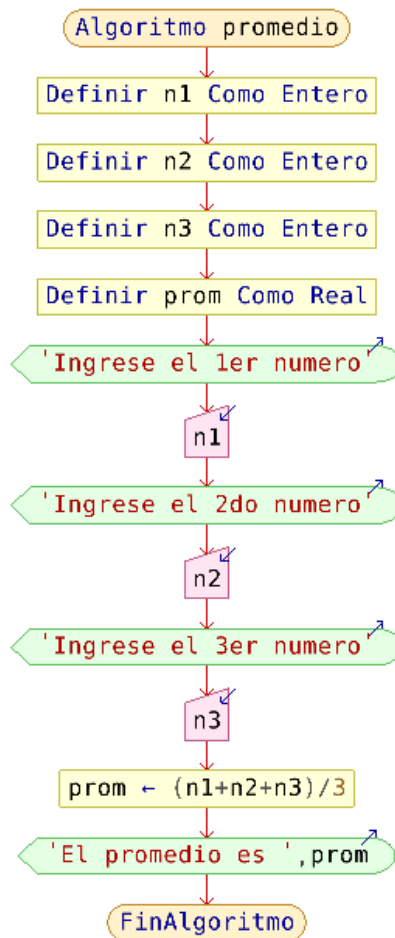
Las variables de proceso son aquellas que en cualquier momento del programa cambian su valor, es decir su valor cambia durante la ejecución del proceso, es por eso que una variable que solo se lee y se utiliza en un condicional no es considerada variable de proceso.

Es importante aclarar que una variable puede disponer de varios tipos. Por ejemplo, la variable <superficie>, es el resultado de un cálculo, el cual es mostrado por pantalla. En este caso la variable <superficie> se definiría como tipo proceso y tipo salida.

Documentación de variables

NOMBRE	DESCRIPCION	TIPO DE DATO	TIPO DE VARIABLE		
			E	P	S
n1	Número 1	Entero	x		
n2	Número 2	Entero	x		
n3	Número 3	Entero	x		
Prom	Promedio	Real		x	x

Ingresar 3 números, calcular el promedio y mostrarlo por pantalla

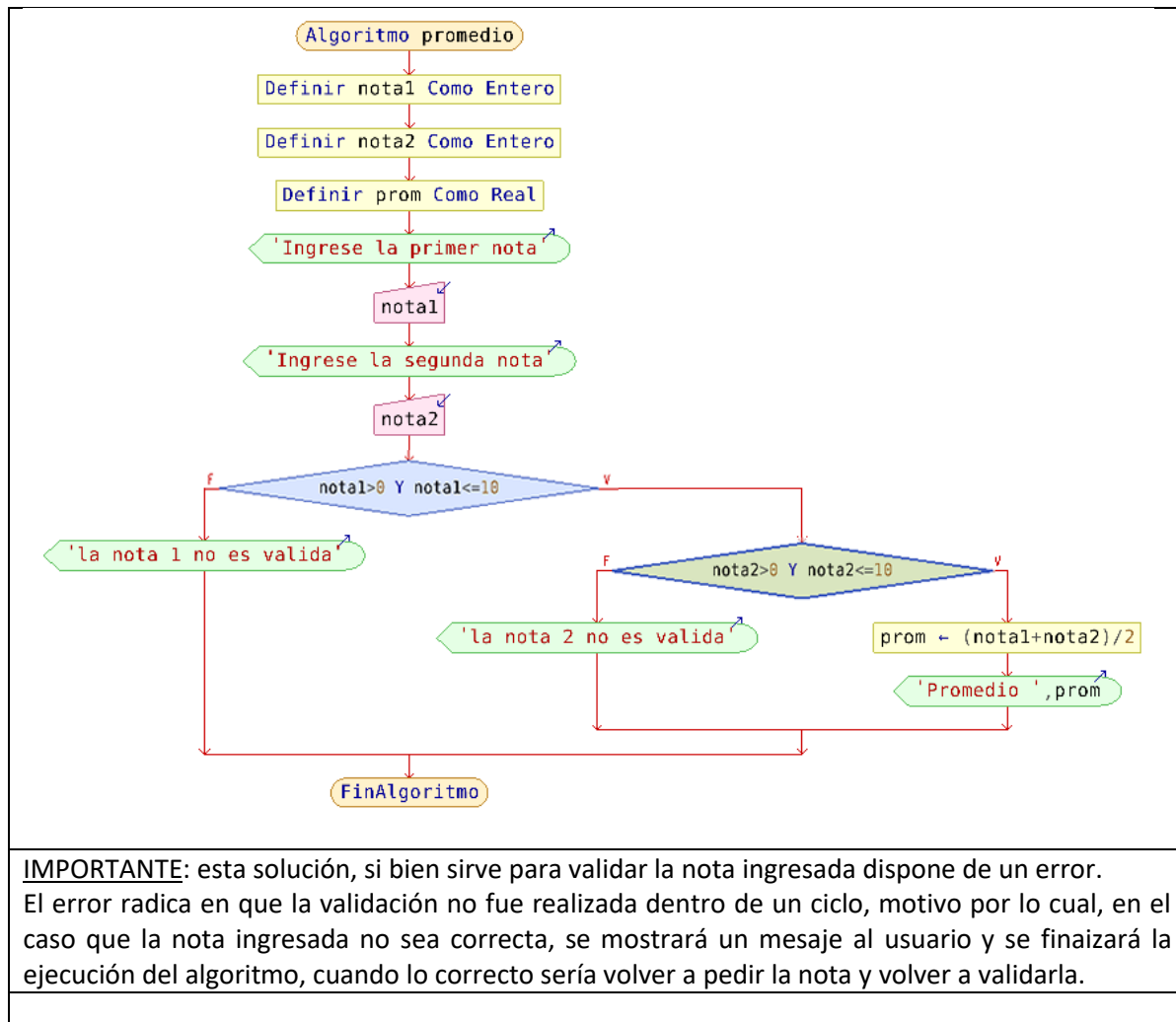


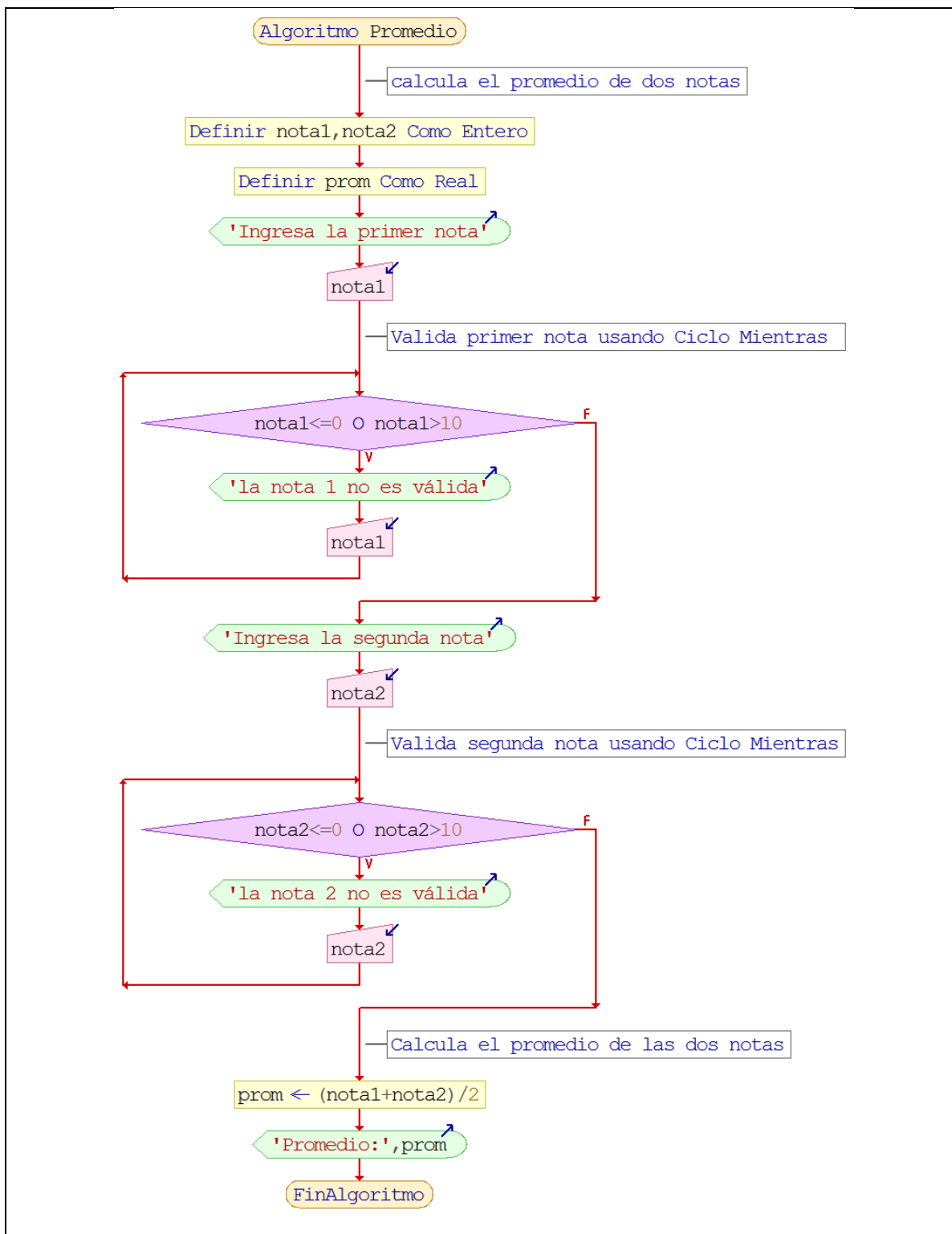
Validaciones

Las validaciones se aplican a los valores ingresados por teclado y sirven para verificar que los mismos se correspondan con los datos solicitados, para asegurarnos que el sistema almacenará y procesará correctamente la información.

Por ejemplo, si tenemos un sistema en donde se cargan las notas de los alumnos, deberíamos validar que las mismas sean mayor a cero y menor o igual que 10. Si no realizamos la validación, el sistema permitiría cargar notas no válidas, como por ejemplo 15 ó -1, lo cual ocasionaría que los cálculos del sistema generen errores.

Ingresar las notas de un alumno y mostrar su promedio
Resolución sin aplicar validaciones
<pre> graph TD Start([Algoritmo promedio]) --> Def1[Definir nota1 Como Entero] Def1 --> Def2[Definir nota2 Como Entero] Def2 --> Def3[Definir prom Como Real] Def3 --> In1[/Ingrese la primer nota/] In1 --> Out1[nota1] Out1 --> In2[/Ingrese la segunda nota/] In2 --> Out2[nota2] Out2 --> Calc[prom ← (nota1+nota2)/2] Calc --> Out3[/Promedio ',prom/] Out3 --> End([FinAlgoritmo]) </pre>
Resolución aplicando validaciones





Prueba de escritorio

La prueba de escritorio es una herramienta que se utiliza para verificar el funcionamiento paso a paso de un determinado algoritmo, para lo cual se realizan simulaciones del comportamiento con el objetivo de determinar la validez del mismo. Permite llevar el registro y seguimiento de los valores que puede asumir cada variable del algoritmo.

La prueba de escritorio consiste en generar una tabla, en donde cada columna de la misma representa variables del algoritmo y condiciones a ser evaluadas.

La prueba de escritorio, permite detectar errores, omisiones o mejorar el algoritmo.

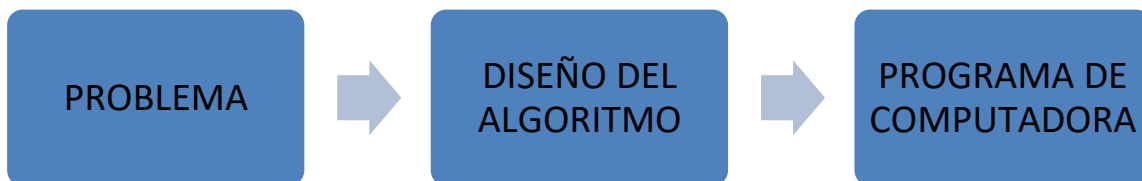
Con la prueba de escritorio es factible detectar errores de lógica (decisiones, cálculos, etc.), de entrada de datos o de salida de datos.

Por ejemplo:

- No ingresar un dato de nacimiento que sea mayor que la fecha actual.
- No ingresar un número negativo donde debe ir uno positivo o sin decimales.
- No ingresar un valor numérico donde sólo debe ir texto.
- No ingresar un valor fuera del rango establecido.

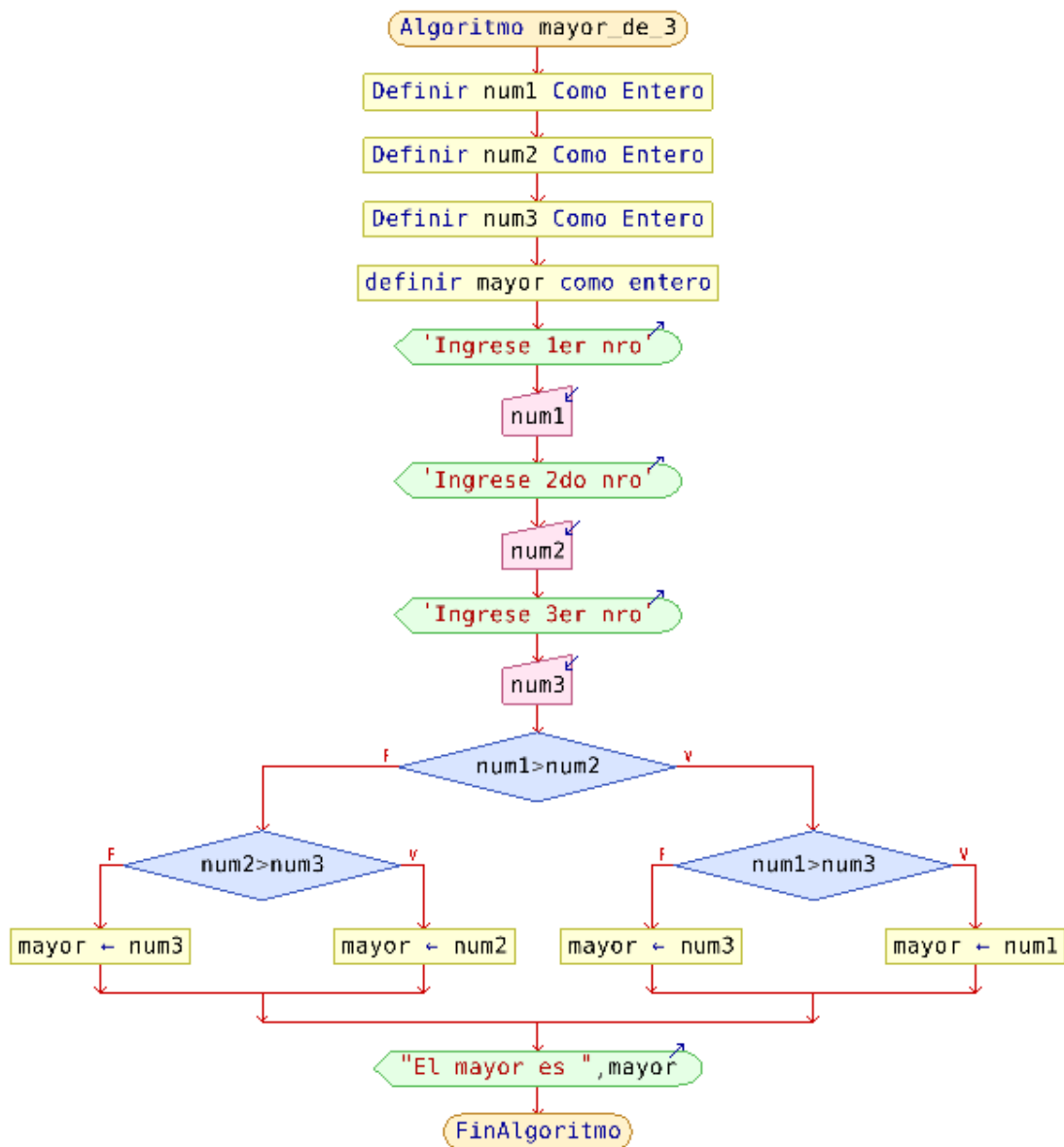
¿Cuándo es conveniente hacer la prueba de escritorio?

Teniendo en cuenta que la etapa del diseño del algoritmo está compuesta por la construcción del diagrama de flujo y el pseudocódigo, es conveniente realizar la prueba de escritorio una vez finalizado el diagrama de flujo, debido a que es importante detectar y corregir errores lo antes posible, para no arrastrarlos en el pseudocódigo.



Para que la prueba de escritorio sea efectiva, la misma debe contener todas las posibilidades a ser evaluadas y reflejar el recorrido exhaustivo del algoritmo diseñado, evitando cálculos y recorridos inexistentes.

Ingresar 3 números por teclado y determinar el número mayor.



Prueba de escritorio:

num1	num2	num3	mayor	num 1 > num2	num1 > num3	num2 > num3	Salida
							Ingrese 1er nro
8							
							Ingrese 2do nro
	6						
							Ingrese 3er nro
		4		V			
					V		
			8				El mayor es: 8
							Ingrese 1er nro
5							
							Ingrese 2do nro
	1						
							Ingrese 3er nro
		7		V			
					F		
			7				El mayor es: 7
							Ingrese 1er nro
2							
							Ingrese 2do nro
	9						
							Ingrese 3er nro
		3		F			
					V		
			9				El mayor es: 9
							Ingrese 1er nro
1							
							Ingrese 2do nro
	5						
							Ingrese 3er nro
		8					
				F			
					F	8	El mayor es: 8

Errores frecuentes

En Algoritmos representados con Diagramas de Flujo

La siguiente tabla presenta un listado de los errores frecuentes detectados en las evaluaciones parciales, entregadas por estudiantes de cohortes anteriores a la realización de este documento,

en línea con las características fundamentales que debe cumplir todo algoritmo, enunciadas en la definición de Algoritmo en las primeras páginas.

Características fundamentales	Errores frecuentes
Descripción de los datos que serán manipulados en el algoritmo	<ul style="list-style-type: none"> - Omite la documentación de variables. - Usa nombre de variables que no refieren a su contenido. - Olvida alguna variable. - Omite la descripción de la variable, es decir, escribir ¿para qué la usa? - No discrimina variables de entrada, proceso y salida. - Confunde el tipo de variable de entrada, de proceso o de salida.
Descripción de las acciones principales que deben ser ejecutadas para la manipulación de los datos	<p>Omite incluir comentarios que señalen las partes principales del Algoritmo. Ejemplos:</p> <pre>//Validación de datos de entrada con ciclo Mientras. //Ciclo de proceso principal. //Comparación de datos.</pre>
Un algoritmo debe ser preciso e indicar el orden de realización de cada paso	<ul style="list-style-type: none"> - Desconoce o no encuentra la forma de resolver el problema planteado. Ejemplos: detección de pares e impares, números primos, generación de series numéricas. - Olvida validar datos de entrada, es decir, puede procesar datos incorrectos. - Omite mensajes que indiquen al usuario que dato debe ingresar. - Inicio erróneo de variables que usará cómo contador o acumulador. - Usa “Selección múltiple” cuando debe usar una “estructura selectiva anidada”, desconoce que la “estructura selectiva múltiple” se usa para comparar diferentes valores que puede tomar UNA sola variable. - Confunde los símbolos <: menor y >: mayor - Confunde el uso de operadores lógicos (Y, O, NO), se sugiere usar la tabla de verdad, para chequear si el resultado es lo que se pide en el enunciado del problema. - Procesa una sola vez, desconoce cómo usar ciclos. - Repite un cálculo en cada una de las salidas de una estructura selectiva compuesta y/o anidada, lo esperable es que ubique la acción una sola vez al finalizar la estructura. - Incluye acciones no requeridas, por ejemplos: cálculos, comparaciones innecesarias. - Usa variable contador para controlar un ciclo PARA, en lugar de la misma variable incluida en el ciclo PARA. - Ubica en forma errónea, dentro del Ciclo PARA, los

	<p>parámetros: inicio, fin, incremento de la variable que controla el ciclo.</p> <ul style="list-style-type: none"> - Desconoce el funcionamiento de la variable dentro del ciclo PARA y agrega otras variables para resolver el problema planteado. Ejemplo en problemas para generar serie de datos. - Usa ciclo MIENTRAS en lugar del ciclo PARA. - Aplica fórmulas complejas, en lugar de usar las estructuras disponibles.
Un algoritmo debe estar definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.	<ul style="list-style-type: none"> - Números aleatorios (random) que sirven de condición de corte en un bucle. - Números aleatorios (random) que sirven de condiciones de evaluación en una estructura de control selectiva.
Un algoritmo debe ser finito. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos	<p>Ciclo o bucle infinito, olvida ingresar el dato de entrada antes de cerrar el ciclo de proceso principal. En el siguiente apartado se trata de forma detallada este error.</p> <p>Ejemplos:</p> <ul style="list-style-type: none"> - Omite el ingreso de la variable que controla el ciclo MIENTRAS, antes de ingresar al Ciclo MIENTRAS. Ejemplo: inicializa una variable o no realiza el ingreso. - Omite el ingreso de la variable que controla el ciclo MIENTRAS antes del cierre del ciclo.

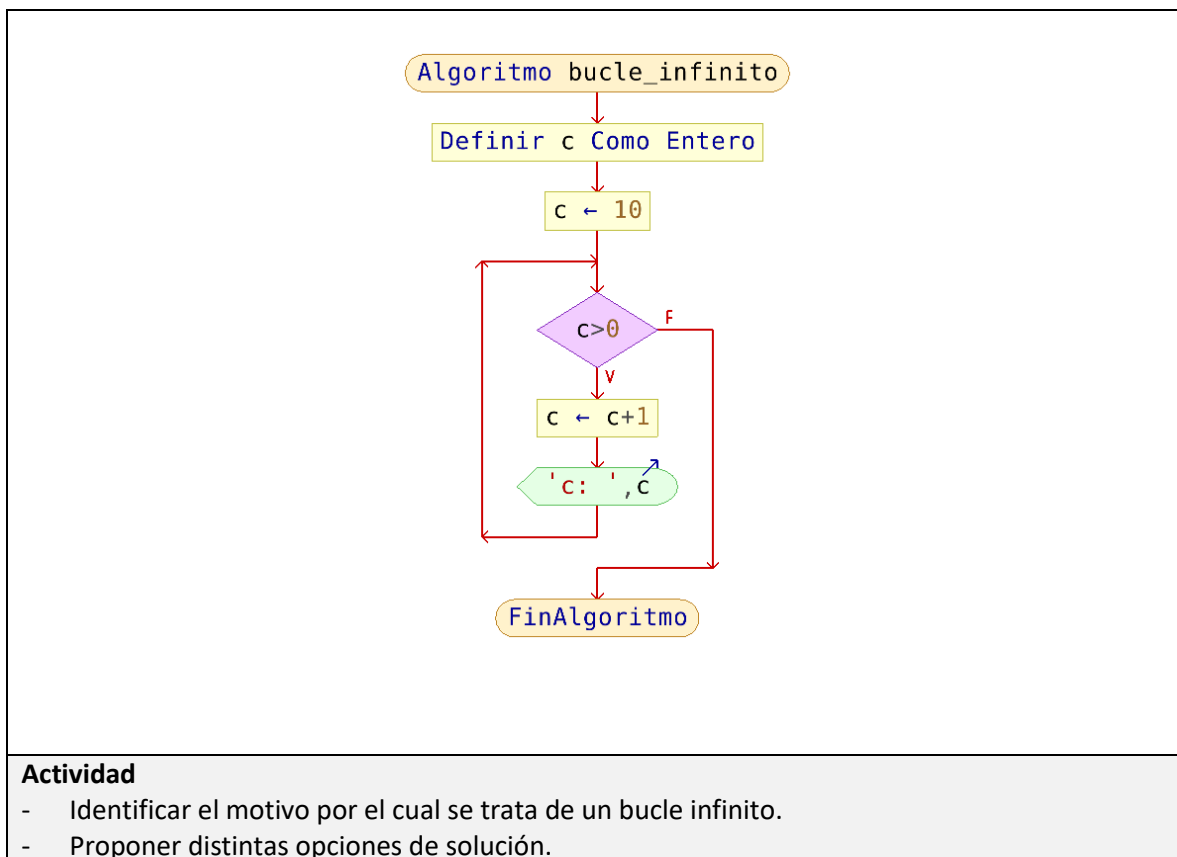
En Pruebas de Escritorio

- No aplica la metodología propuesta y desarrollada en clase, aplica un estilo propio
- Omite alguna variable
- Omite alguna comparación de variables existente en estructuras de selección
- Olvida la salida
- Olvida chequear el algoritmo completo, equivoca o retacea la elección de datos que aseguren el recorrido de todos los caminos existentes en el diagrama de flujo
- Imagina o predice resultados en lugar de seguir las acciones tal como las escribió en el Diagrama de Flujo.

Ciclos o Bucles infinitos

En la práctica podemos encontrarnos con bucles que no exigen una finalización y otros que no la incluyen en su diseño. Por ejemplo, un sistema para cargar las notas de los alumnos puede repetir de forma indeterminada un bucle que permita al usuario añadir nuevas notas sin ninguna condición de finalización. El programa y el bucle se ejecutan siempre, o al menos Repetir...Hasta que la computadora se apague. En otras ocasiones, un bucle no se termina porque nunca se cumple la condición de salida. Un bucle de este tipo se denomina bucle infinito o sin fin. Los bucles

infinitos no intencionados, causados por errores de programación, pueden provocar bloqueos en el programa.



Pseudocódigo

El pseudocódigo es un lenguaje de especificación de algoritmos (no de programación) basado en un sistema notacional, con estructuras sintácticas y semánticas, similares a los lenguajes procedurales y que no puede ser ejecutado directamente en una computadora.

El pseudocódigo utiliza palabras reservadas para su representación.

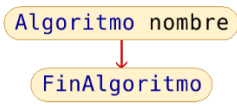
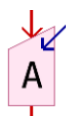
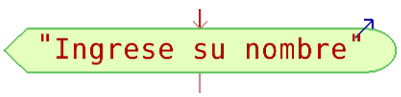
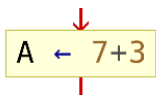
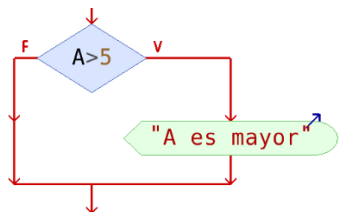
El pseudocódigo es una forma de escribir los pasos que va a realizar un programa de la forma más cercana al lenguaje de programación que vamos a utilizar posteriormente.

Características que debe tener un pseudocódigo:

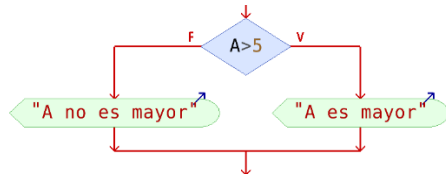
- Se puede ejecutar en un ordenador
- Es una forma de representación sencilla de utilizar y de manipular.
- Facilita el paso del programa al lenguaje de programación.
- Es independiente del lenguaje de programación que se vaya a utilizar.
- Es un método que facilita la programación y solución al algoritmo del programa.

Sentencias

La simbología utilizada a continuación se corresponde con el software PSeInt.

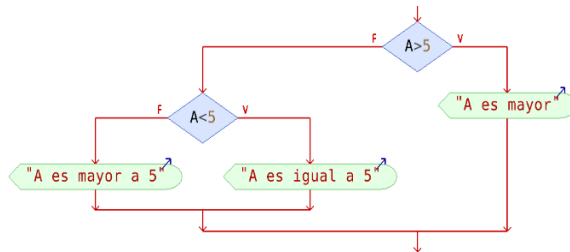
SIMBOLO	PSEUDOCODIGO
Terminal: dispone de una doble representación: - Inicio: representa el comienzo del algoritmo. - Fin: representa el final del algoritmo.	
	Algoritmo nombre ... FinAlgoritmo
Lectura: representa el ingreso de una variable por parte del usuario.	
	Leer A
Escritura: representa la salida de un texto o una variable por pantalla. <u>Importante:</u> si el texto que se va a mostrar tiene palabras reservadas PSeInt generará un error en tiempo de ejecución. Se recomienda colocar todo el texto entre comillas dobles o simples.	
	Escribir "Ingrese su nombre"
Asignación: se utiliza para darle un valor a una variable.	
	$A \leftarrow 7+3$
Estructura selectiva simple: Se evalúa la condición y si se cumple se realizan las acciones correspondientes.	
	Si $A > 5$ Entonces Escribir "A es mayor" FinSi

Estructura selectiva doble: Se evalúa la condición, si se cumple se realizan determinadas acciones y si no se cumple se realizan otras acciones diferentes.



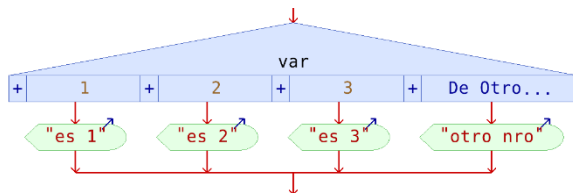
Si $A > 5$ Entonces
 Escribir "A es mayor"
 SiNo
 Escribir "A no es mayor"
 FinSi

Estructura selectiva compuesta: Se utiliza cuando se necesita evaluar más de una condición con un intervalo de valores.



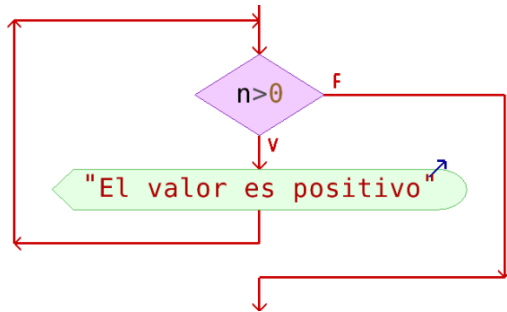
Si $A > 5$ Entonces
 Escribir 'A es mayor'
 SiNo
 Si $A < 5$ Entonces
 Escribir "A es igual a 5"
 SiNo
 Escribir "A es mayor a 5"
 FinSi
 FinSi

Estructura selectiva múltiple: Se utiliza cuando una variable a evaluar puede tomar diferentes valores exactos, no en intervalos.



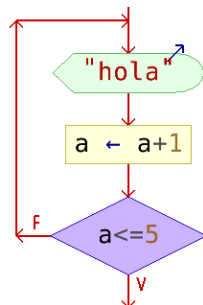
Segunvar Hacer
 1: Escribir "es 1"
 2: Escribir "es 2"
 3: Escribir "es 3"
 De Otro Modo: Escribir "otro nro"
 FinSegun

Ciclo Mientras: este ciclo se utiliza para repetir determinadas acciones mientras la condición que se está evaluando sea verdadera. Finaliza cuando la condición pasa a ser falsa.



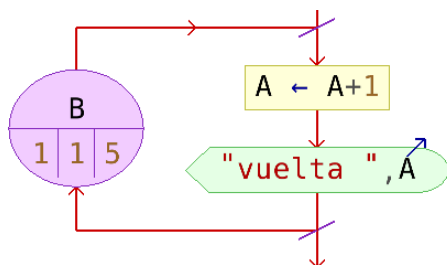
Mientras $n > 0$ Hacer
 Escribir "El valor es positivo"
 FinMientras

Ciclo Repetir...Hasta que: este ciclo asegura que por lo menos se ejecutará una vez, y posteriormente se evaluará la condición para ver si se repite la ejecución. Finaliza cuando la condición es Verdadera.



Repetir
 Escribir "hola"
 $a \leftarrow a + 1$
 Repetir...Hasta que $A \leq 5$

Ciclo Para: se utiliza este ciclo cuando se sabe de antemano la cantidad de repeticiones que se van a realizar. En este caso se va a repetir 5 veces.



Para $B \leftarrow 1$ Hasta 5 Hacer
 $A \leftarrow A + 1$
 Escribir "vuelta ", A
 FinPara

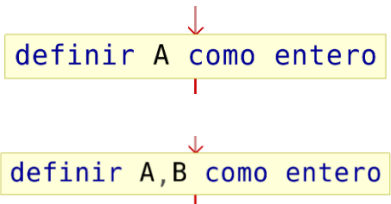
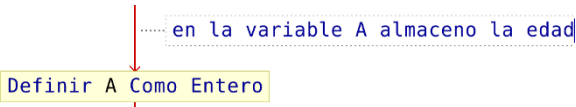
Tipo de dato: Definir el tipo de dato de una variable	
	<p>definir A como entero</p> <p>definir A,B como entero</p>
Comentarios: sirve para agregar texto de explicación o ayuda, que sirve para interpretar el diagrama o pseudocódigo.	
	<p>// en la variable A almaceno la edad</p> <p>Definir A Como Entero</p>

Tabla 2

Comentarios

En la construcción de pseudocódigo, un comentario consiste en incrustar anotaciones legibles al programador en el pseudocódigo. Estas anotaciones son potencialmente significativas para los programadores, pero son ignoradas en la ejecución. Los comentarios son añadidos usualmente con el propósito de hacer el pseudocódigo más fácil de entender y que son muy útiles en el mantenimiento. Para realizar comentarios debemos anteponer una doble barra (//)

Ingresar 2 números por teclado y mostrar por pantalla la suma de los mismos.	
<pre>graph TD A([Algoritmo suma_dos_nros]) --> B[Definir n1, n2 Como Entero] B --> C[Definir suma Como Entero] C --> D[/Ingrese el primer nro/] D --> E[n1] E --> F[/Ingrese el segundo nro/] F --> G[n2] G --> H[en suma almaceno n1+n2] H --> I[suma <- n1+n2] I --> J[/La suma es: ', suma/] J --> K([FinAlgoritmo])</pre>	<pre>Algoritmo suma_dos_nros definir n1,n2 como entero definir suma como entero Escribir "Ingrese el primer nro" Leer n1 Escribir "Ingrese el segundo nro" Leer n2 // en suma almaceno n1+n2 suma <- n1+n2 Escribir "La suma es: ",suma FinAlgoritmo</pre>

Errores frecuentes

En Algoritmos representados en Pseudocódigo

- Mala interpretación de enunciados
- Error en tipo de datos
- Confusión en tipo de variable
- Confunden asignación con ingreso
- Problema en la descripción de las variables
- No inicializan variables que usarán como contador o acumulador
- Inicializan variables que están dentro de la acción de entrada de datos, es decir, valores que va a ingresar el usuario
- Confusión con los signos > (mayor) y < (menor) en las comparaciones dentro de la estructura selectiva
- Problemas con el funcionamiento del PARA

- No cierra los ciclos
- Uso de escritura propia, Ejemplo: = o \leftarrow , Escribir o Mostrar, leer o ingresar
- Confusión entre el uso de estructuras de repetición
- Uso de terminología de Scratch
- No muestran mensajes para que el usuario sepa que dato debe ingresar
- No documentan, es decir omiten comentarios dentro del algoritmo para indicar que hace en cada parte
- Uso de estructura selectiva anidada, en lugar de estructura selectiva múltiple
- Cuando tiene que finalizar ingresando 0 (cero) se confunden y hacen un SI
- Cierran el SI y el SI NO
- No validan
- Confusión entre contador y acumulador

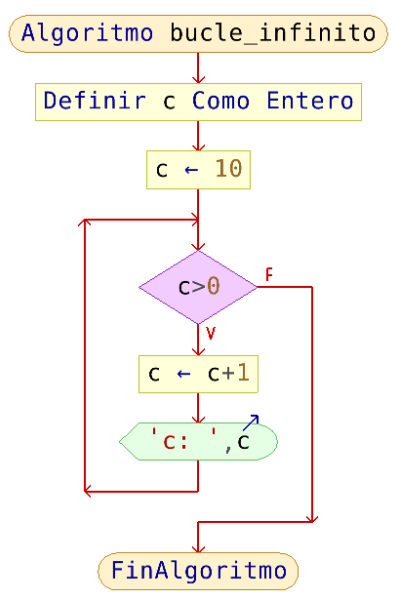
En la realización de la Prueba de Escritorio

- No usar metodología propuesta y desarrollada en clase
- No se prueba con todos los valores necesarios
- Omiten las columnas donde se prueban las condiciones que se encuentran dentro de las estructuras selectivas o de repetición
- Omiten probar con valores cero o con valores negativos
- Se da por cierto los valores
- No hacen P.E. si encuentran una variable "rara"
- No se verifican las validaciones de los datos de entrada

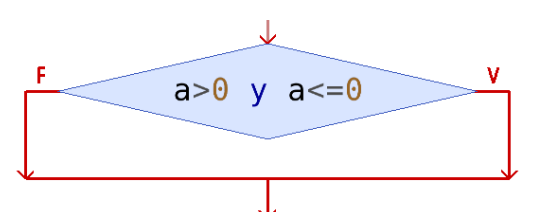
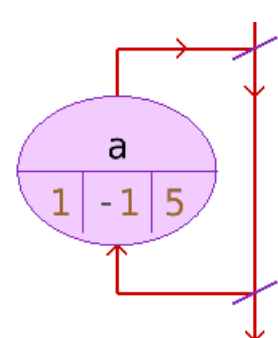
En ejercicios de Detección de errores

- Se deducen sin hacer la Prueba de Escritorio.
- Cuando se pide realizar el algoritmo correcto a partir del algoritmo ofrecido como solución, modifican la estructura total del algoritmo original.
- Cuando se solicita pasar el Diagrama de Flujo a Pseudocódigo y se pide redactar el enunciado del problema que resuelve: se observa mala, inadecuada y confusa redacción del enunciado

Ciclos o Bucles infinitos

 <pre> graph TD Start([Algoritmo bucle_infinito]) --> Def[Definir c Como Entero] Def --> Init[c ← 10] Init --> Cond{c > 0} Cond -- V --> Inc[c ← c + 1] Inc --> Out[/c: ', c/] Out --> Cond Cond -- F --> End([FinAlgoritmo]) </pre>	<p>Algoritmo bucle_infinito Definir c Como Entero c <- 10 Mientras c>0 Hacer c <- c+1 Escribir "c: ",c FinMientras FinAlgoritmo</p>
---	---

Reglas bien definidas en su lógica

	<p>Independientemente del número ingresado, el resultado de la evaluación de la condición lógica, siempre daría falso, ya que un número no puede ser mayor a cero y menor/igual a cero.</p> <p>Si $a > 0$ y $a \leq 0$ Entonces</p>
	<p>No es posible que la variable a, empiece a recorrer el ciclo valiendo 1 hasta llegar a el valor 5, si en cada paso decrementa su valor en una unidad.</p> <p>Para a<-1 Hasta 5 Con Paso -1 Hacer FinPara</p>

Sin ambigüedades

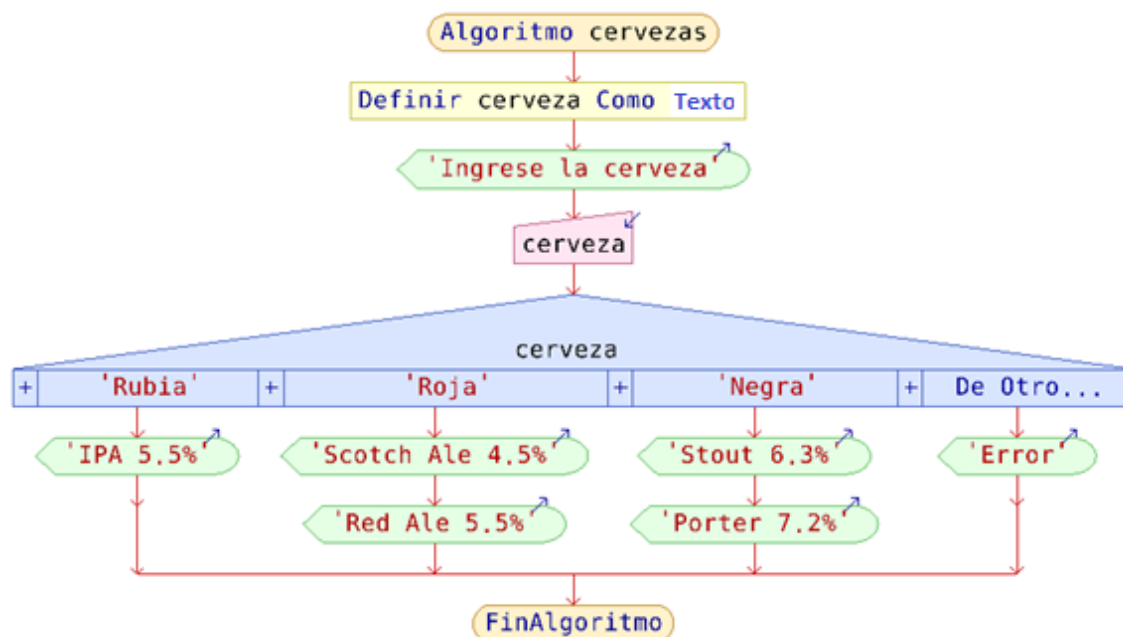
No está contemplado abordar el tema en esta materia, pero debe ser considerado en las materias de Programación.

Ejercicios resueltos

Ejercicio 1

Desarrollar el algoritmo para una APP que permita mostrar la carta de cervezas disponibles en una Cervecería, en donde al ingresar se debe seleccionar el tipo (rubia, roja ó negra) de cerveza y mostrar las opciones disponibles.

NOMBRE VARIABLE	DESCRIPCION	TIPO DE DATO	TIPO DE VARIABLE		
			E	P	S
cerveza	Tipo de cerveza	Texto	X	X	



cerveza	Según cerveza				Salida
	Rubia	Roja	Negra	De otro Modo	
					Ingrese la cerveza
Rubia					
	V	F	F	F	IPA 5.5%
					Ingrese la cerveza
Roja					
	F	V	F	F	Scotch Ale 4.5%
					Porter 7.2%
					Ingrese la cerveza
Negra					
	F	F	V	F	Stout 6.3%
					Porter 7.2%"
					Ingrese la cerveza
Blanca					
	F	F	F	V	Error

Algoritmo cervezas

```

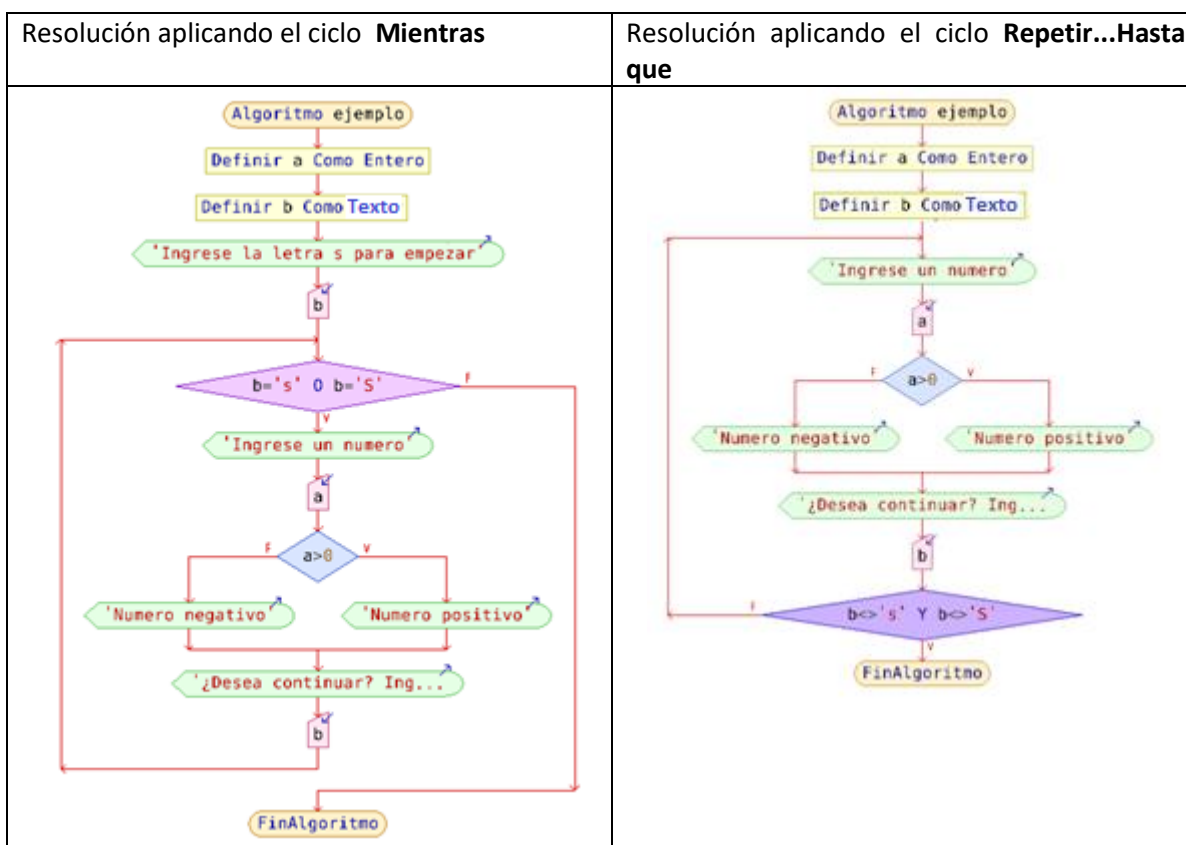
    Definir cerveza Como Texto
    Escribir "Ingrese la cerveza"
    Leer cerveza
    Segun cerveza Hacer
        "Rubia":
            Escribir "IPA 5.5%"
        "Roja":
            Escribir "Scotch Ale 4.5%"
            Escribir "Red Ale 5.5%"
        "Negra":
            Escribir "Stout 6.3%"
            Escribir "Porter 7.2%"
        De Otro Modo:
            Escribir "Error"
    FinSegun
FinAlgoritmo

```

Ejercicio 2

Ingresa una serie de números y, para cada uno de ellos, mostrar por pantalla si el mismo es positivo o negativo. Para continuar con la carga del siguiente número se debe ingresar la letra s

NOMBRE VARIABLE	DESCRIPCION	TIPO DE DATO	TIPO DE VARIABLE		
			E	P	S
B	Letra s para continuar	Texto	X	x	
A	Número	Entero	X	x	



Prueba de escritorio aplicando el ciclo **Mientras**

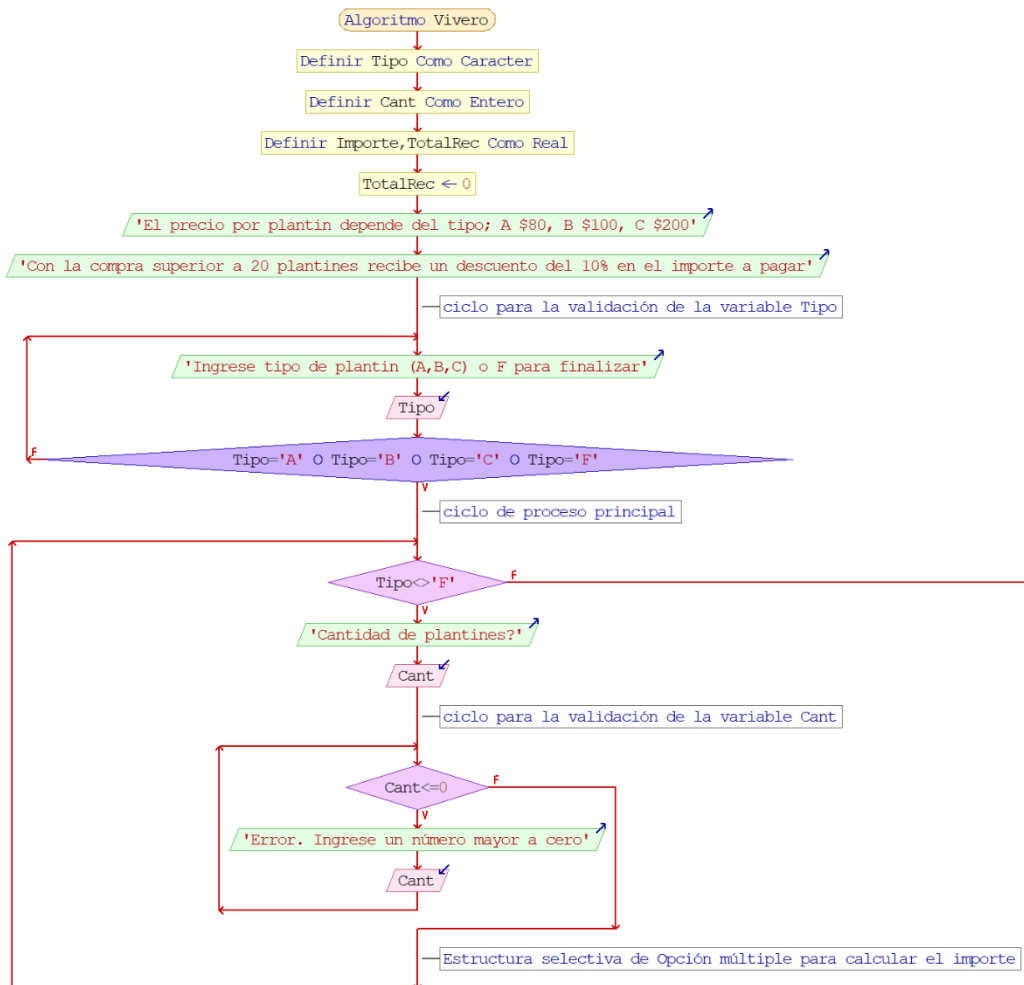
b	a	b=s O b=S	a>0	Salida
				Ingrese la letra s para empezar
S		V		
				Ingrese un numero
	5			
			V	Numero positivo
				¿Desea continuar? Ing..
S		V		
				Ingrese un numero
	-1			
			F	Numero negativo
				¿Desea continuar? Ing..
N		F		

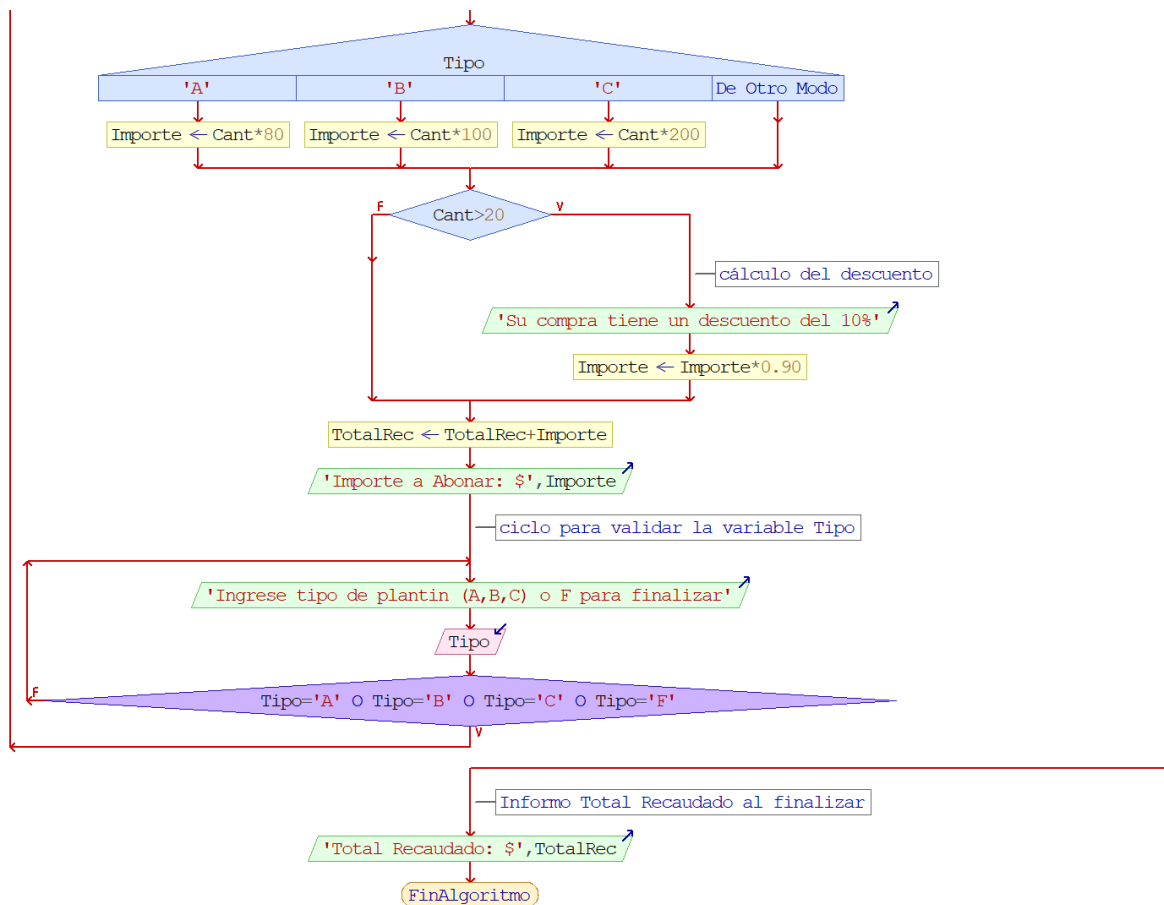
Prueba de escritorio aplicando el ciclo **Repetir...Hasta que**

b	a	b=s O b=S	a>0	Salida
				Ingrese un numero
	5			
			V	Numero positivo
				¿Desea continuar? Ing..
S		V		
				Ingrese un numero
	-1			
			F	Numero negativo
				¿Desea continuar? Ing..
N		F		

Ejercicio 3

Un vivero vende tres tipos de plantines (A, B y C), cuyo costo es de \$80, \$100 y \$200 respectivamente. Por cada compra superior a 20 plantines, el cliente obtiene un descuento del 10%. Dado el tipo de plantín que compra el cliente (sólo compra de un tipo) y la cantidad, mostrar a cada cliente el importe que tiene que pagar; y al finalizar el proceso mostrar el total recaudado. El proceso finaliza cuando ingresa F en tipo de plantín.





Nombre	Descripción	Tipo de dato	Tipo de Variable		
			E	P	S
Tipo	Tipo de plantín (A,B,C)	Caracter	X		
Cant	Cantidad de plantines compra	Entero	X		
Importe	Importe a abonar	Real		X	X
TotalRec	Total recaudado	Real		X	x

Algoritmo Vivero

Definir Tipo Como Caracter
Definir Cant Como Entero
Definir Importe, TotalRec Como Real
TotalRec <- 0
Escribir 'El precio por plantin depende del tipo; A \$80, B \$100, C \$200'
Escribir 'Con la compra superior a 20 plantines recibe un descuento del 10% en el importe a pagar'

```
// ciclo para la validación de la variable Tipo
Repetir
    Escribir 'Ingrese tipo de plantin (A,B,C) o F para finalizar'
    Leer Tipo
Hasta Que Tipo='A' O Tipo='B' O Tipo='C' O Tipo='F'
// ciclo de proceso principal
Mientras Tipo<>'F' Hacer
    Escribir 'Cantidad de plantines?'
    Leer Cant
    // ciclo para la validación de la variable Cant
    Mientras Cant<=0 Hacer
        Escribir 'Error. Ingrese un número mayor a cero'
        Leer Cant
    FinMientras
    // Estructura selectiva de Opción múltiple para calcular el importe
    Segun Tipo Hacer
        'A':
            Importe <- Cant*80
        'B':
            Importe <- Cant*100
        'C':
            Importe <- Cant*200
    FinSegun
    Si Cant>20 Entonces
        // cálculo del descuento
        Escribir 'Su compra tiene un descuento del 10%'
        Importe <- Importe*0.90
    FinSi
    TotalRec <- TotalRec+Importe
    Escribir 'Importe a Abonar: $',Importe
    // ciclo para validar la variable Tipo
    Repetir
        Escribir 'Ingrese tipo de plantin (A,B,C) o F para finalizar'
        Leer Tipo
    Hasta Que Tipo='A' O Tipo='B' O Tipo='C' O Tipo='F'
    FinMientras
    // Informo Total Recaudado al finalizar
    Escribir 'Total Recaudado: $',TotalRec
FinAlgoritmo
```

Ejercicio 4

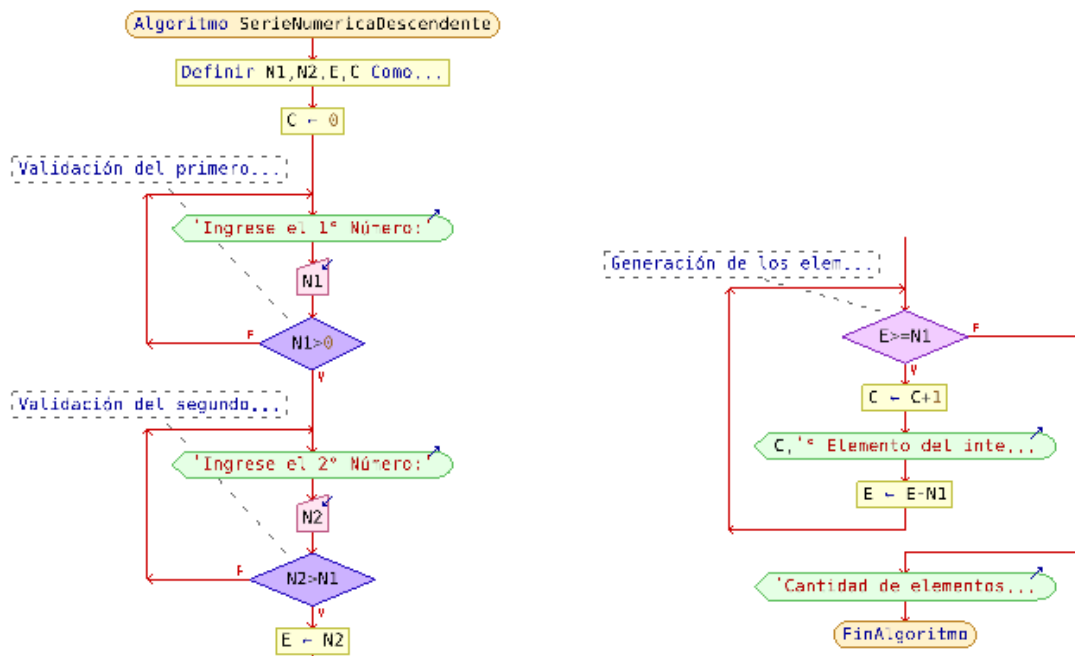
Ingresar dos números enteros positivos, el primero (N1) menor al segundo (N2). Generar la serie numérica descendente del intervalo [N2, N1]. Según se ilustra en el siguiente ejemplo:

Para el intervalo [3,10], los elementos son: 10, 7, 4

Para el intervalo [6,15], los elementos son: 15, 9

Se pide mostrar cada uno de los elementos de la serie. Al finalizar mostrar la cantidad de elementos de la serie.

Para resolver este algoritmo se necesita encontrar el patrón que relaciona los elementos de la serie propuesta



Nombre de Variable	Descripción	Tipo de dato	Tipo de Variable		
			E	P	S
N1	Primer número ingresado	Entero	X		
N2	Segundo número ingresado	Entero	X		
E	Elemento de la serie	Entero		X	X
C	Cantidad de elementos de la serie	Entero		X	X

Algoritmo SerieNumericaDescendente

Definir N1, N2, E, C Como Entero

C <- 0

Repetir // Validación del primero número ingresado

 Escribir 'Ingrese el 1° Número:'

 Leer N1

Hasta Que N1>0

Repetir // Validación del segundo número ingresado

 Escribir 'Ingrese el 2° Número:'

 Leer N2

Hasta Que N2>N1

E <- N2

Mientras E >= N1 Hacer // Generación de los elementos de la serie hasta el tope N

 C <- C+1

 Escribir C, 'º Elemento del intervalo [', N2, ', ', N1, ']:', E

 E <- E - N1

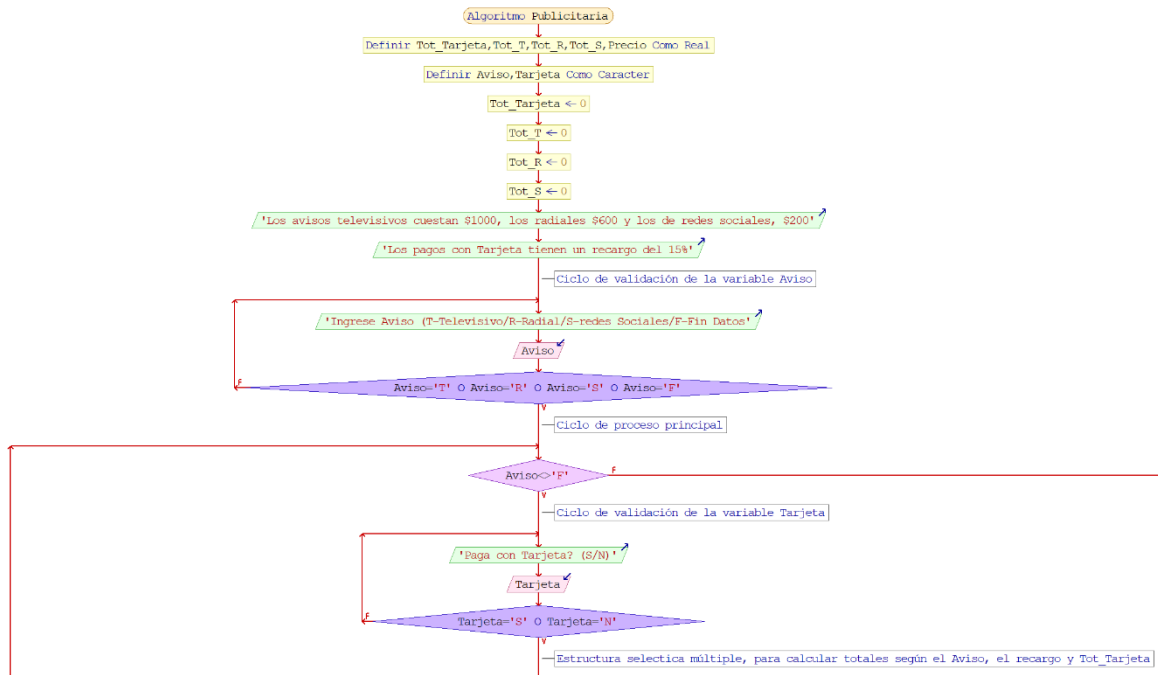
FinMientras

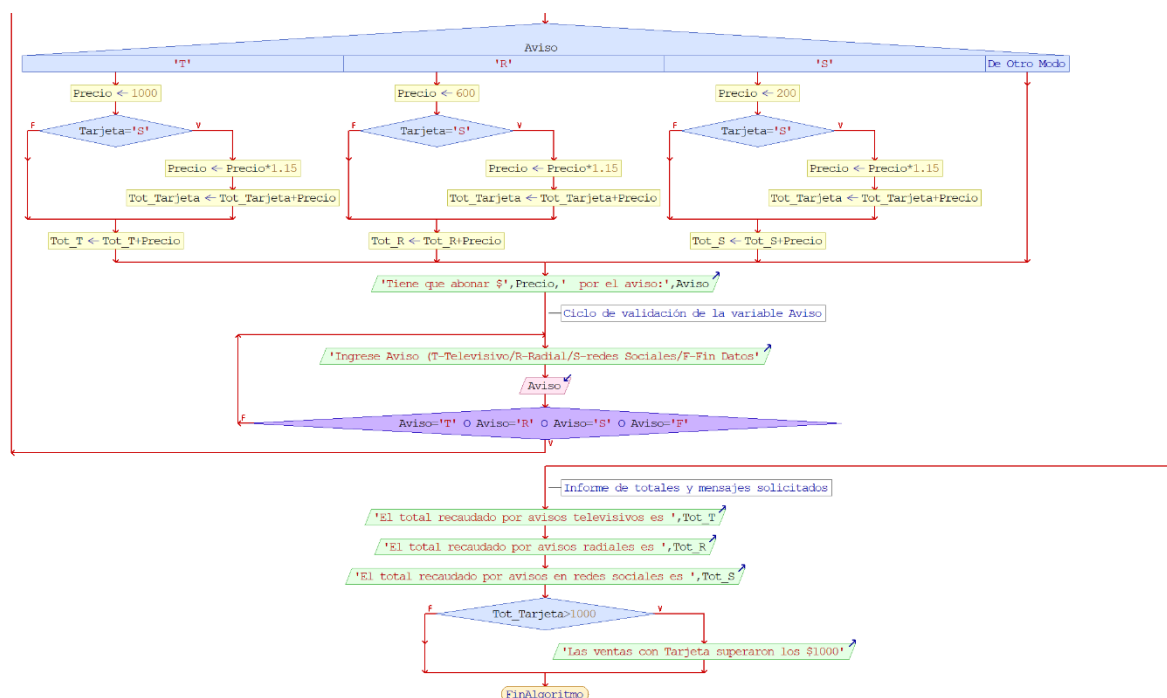
Escribir 'Cantidad de elementos de la serie:', C

FinAlgoritmo

Ejercicio 5

Una compañía publicitaria ofrece avisos en tres modalidades: televisivos (T), radiales (R) y por redes sociales (S). Los avisos televisivos cuestan \$1000, los radiales \$600 y los de redes sociales, \$200. Si el pago es con tarjeta, tiene un 15% de recargo. Se pide mostrar el importe total vendido en cada modalidad (televisión, radio o redes sociales) y si el importe que se pagó en Tarjeta superó los \$1000





Nombre de Variable	Descripción	Tipo de dato	Tipo de Variable		
			E	P	S
Aviso	Tipo de aviso publicitario (Televisión/Radio/Redes Sociales)	Carácter	X		
Tarjeta	Respuesta a la pregunta: ¿Paga con Tarjeta? S/N	Carácter	X		
Precio	Precio del aviso, considerando interés del 15% pago con tarjeta o no	Real		X	
Tot_Tarjeta	Total recaudado por avisos pagados con Tarjeta	Real		X	
Tot_T	Total recaudado por avisos televisivos	Real		X	X
Tot_R	Total recaudado por avisos radiales	Real		X	X
Tot_S	Total recaudado por avisos en redes sociales	Real		X	X

Algoritmo Publicitaria

```
Definir Tot_Tarjeta,Tot_T,Tot_R,Tot_S,Precio Como Real
Definir Aviso,Tarjeta Como Caracter
Tot_Tarjeta <- 0
Tot_T <- 0
Tot_R <- 0
Tot_S <- 0
Escribir 'Los avisos televisivos cuestan $1000, los radiales $600 y los de redes sociales,
$200'
Escribir 'Los pagos con Tarjeta tienen un recargo del 15%'
// Ciclo de validación de la variable Aviso
Repetir
    Escribir 'Ingrese Aviso (T-Televisivo/R-Radial/S-redes Sociales/F-Fin Datos'
    Leer Aviso
Hasta Que Aviso='T' O Aviso='R' O Aviso='S' O Aviso='F'
// Ciclo de proceso principal
Mientras Aviso<>'F' Hacer
    // Ciclo de validación de la variable Tarjeta
    Repetir
        Escribir 'Paga con Tarjeta? (S/N)'
        Leer Tarjeta
    Hasta Que Tarjeta='S' O Tarjeta='N'
    // Estructura selectica múltiple, para calcular totales según el Aviso, el recargo y
Tot_Tarjeta
    Segun Aviso Hacer
        'T':
            Precio <- 1000
            Si Tarjeta='S' Entonces
                Precio <- Precio*1.15
                Tot_Tarjeta <- Tot_Tarjeta+Precio
            FinSi
            Tot_T <- Tot_T+Precio
        'R':
            Precio <- 600
            Si Tarjeta='S' Entonces
                Precio <- Precio*1.15
                Tot_Tarjeta <- Tot_Tarjeta+Precio
            FinSi
            Tot_R <- Tot_R+Precio
        'S':
            Precio <- 200
            Si Tarjeta='S' Entonces
                Precio <- Precio*1.15
                Tot_Tarjeta <- Tot_Tarjeta+Precio
            FinSi
```

```

                                Tot_S <- Tot_S+Precio
    FinSegun
    Escribir 'Tiene que abonar $',Precio,' por el aviso:',Aviso
    // Ciclo de validación de la variable Aviso
    Repetir
        Escribir 'Ingrese Aviso (T-Televisivo/R-Radial/S-redes Sociales/F-Fin Datos'
        Leer Aviso
    Hasta Que Aviso='T' O Aviso='R' O Aviso='S' O Aviso='F'
    FinMientras
    // Informe de totales y mensajes solicitados
    Escribir 'El total recaudado por avisos televisivos es ',Tot_T
    Escribir 'El total recaudado por avisos radiales es ',Tot_R
    Escribir 'El total recaudado por avisos en redes sociales es ',Tot_S
    Si Tot_Tarjeta>1000 Entonces
        Escribir 'Las ventas con Tarjeta superaron los $1000'
    FinSi
FinAlgoritmo
```

Para resolver:

Diseñar un algoritmo que permita detectar si una persona tiene COVID-19, para lo cual se deben realizar las siguientes preguntas:

- ¿Tenes fiebre? Si la respuesta es sí, se debe ingresar la temperatura.
- ¿Tenes tos?
- ¿Dolor de garganta?
- ¿Dificultad para respirar?
- ¿Visitaste otro país en los últimos 14 días?

Cada pregunta se responde con n (no) ó s (si).

Se pide:

- En el caso que la temperatura sea mayor a 38, y existan 3 respuestas con “s”, se debe solicitar a la persona que acuda a un centro de atención.
- En el caso que existan 3 respuestas con “s” y no se tenga fiebre, se debe solicitar a la persona que permanezca en su casa.
- En el caso que existan 2 respuestas con “s”, se debe solicitar realizar el test en 24 hs.
- Realizar todas las validaciones necesarias.

Notas

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.

Opciones del lenguaje:

El sistema PSeINT permite trabajar con distintos perfiles de usuarios, en donde se configuran los parámetros a ser utilizados y el esquema de trabajo.

El equipo docente de la materia Fundamentos de la Informática definió el perfil “FI-UNMdP” el cual se adapta con el material y la metodología de la materia. Por este motivo es importante que al utilizar el sistema PSeINT se configure el perfil en cuestión, el cual está disponible en el campus de la materia para ser descargado.

Para adjuntar el perfil en el sistema PSeINT, se debe seguir los siguientes pasos:

- En el menú del sistema ingresar a la opción “Configurar”, “Opciones del Lenguaje (perfiles)...”.
- Posteriormente realizar clic en el botón “Cargar”.
- Seleccionar el archivo “FI-UNMdP”.
- Hacer click en “Abrir” y “Aceptar”.

Link de acceso a ejemplos de algoritmos representados con Pseudocódigo:

<http://pseint.sourceforge.net/index.php?page=ejemplos.php>

Bibliografía

- HERRERA-GOMEZ-PORTILLA (2016). *Diseño y construcción de algoritmos*. Universidad del Norte.
- JOYANES AGUILAR L. (2003). *Fundamentos de programación*. España: Mc Graw Hill.
- VILLEGAS JARAMILLO E., GUERRERO MENDIETA L. (2016). *Análisis y Diseño de Algoritmos. Un enfoque práctico*. Colombia: Universidad Nacional de Colombia
- Sitio oficial PSeINT. *PseInt – Manuales, documentación y ejercicios*
www.pseint.sourceforge.net