

**UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTA REGIONAL ROSARIO**

FUNDAMENTOS DE INFORMÁTICA

Profesor: Dra. Sonia Benz.

Auxiliares: Ing. Evangelina Delfratte - Ing. Patricia Mores

**Introducción a la programación.
Estructuras algorítmicas.**

Elaboración: Ing. Patricia Mores - Ing. Evangelina Delfratte

TABLA DE CONTENIDOS

INTRODUCCIÓN.	3
ALGORITMOS.	3
CARACTERÍSTICAS DE LOS ALGORITMOS.	4
REPRESENTACIÓN DE ALGORITMOS.	5
ALGORITMOS. ESTRUCTURAS BÁSICAS.	7
OPERADORES LÓGICOS.	11
USO DE ARREGLOS MATRICIALES PARA ALMACENAMIENTO DE DATOS.	11
ALGORITMOS CUALITATIVOS. EJEMPLOS.	13
ESTRUCTURAS DE SECUENCIACIÓN.	13
ESTRUCTURAS DE DECISIÓN.	14
ESTRUCTURAS DE REPETICIÓN.	15
ESTRUCTURAS ANIDADAS.	16
ALGORITMOS CUANTITATIVOS. EJEMPLOS.	17
ESTRUCTURAS SECUENCIALES.	17
ESTRUCTURAS DE SELECCIÓN.	19
ESTRUCTURAS DE REPETICIÓN.	20

INTRODUCCIÓN.

El siguiente apunte tiene por objeto orientar al alumno en el aprendizaje de los conceptos generales de la programación mediante ejemplos concretos. Se realizará un avance progresivo en los conceptos utilizando como herramienta el programa Matlab.

ALGORITMOS.

En matemáticas, ciencias de la computación y disciplinas relacionadas, un algoritmo es un conjunto ordenado y finito de pasos o instrucciones que permite realizar una actividad mediante pasos sucesivos sin generar dudas a quien deba realizar dicha actividad, conduciendo a la solución de un problema determinado. De esta manera, dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución.

En la vida cotidiana, frecuentemente se emplean algoritmos para resolver problemas. Por ejemplo: las instrucciones para el montaje de un equipo, las instrucciones de uso de equipos electrónicos, las técnicas de laboratorio para valorar ácidos, etc. Cualquier proceso que implique un análisis de la situación y una posible solución puede dar lugar a un algoritmo (no necesariamente desde el punto de vista de las ciencias). De esta manera, nos encontramos con algoritmos del tipo cualitativos y/ o cuantitativos. Veamos algunos ejemplos:

Algoritmo cualitativo

Problema 1: insertar tarjeta SIM en un celular.

Datos: conjunto de piezas iniciales (celular, tarjeta SIM)

- Inicio
- Abra la tapa de la ranura de la tarjeta SIM.



- Inserte la tarjeta SIM en la ranura. Asegúrese que el área de contacto de la tarjeta esté orientada hacia arriba y que la esquina biselada esté orientada hacia el dispositivo.

Algoritmo cuantitativo

Problema 2: calcular el área de un triángulo.

Datos: base y altura, unidades de medición.

- Inicio
- Ingresar la base
- Ingresar la altura
- Calcular el área
- Mostrar resultado
- Fin



- Presione la tarjeta.
- Cierre la tapa de la ranura de la tarjeta.



- Fin

En el problema 1 (cualitativo), el algoritmo es preciso solo si se presentan las imágenes orientativas, ya que de lo contrario, el usuario podría mal interpretar el procedimiento y se obtendría un resultado final erróneo. De la misma manera, el problema 2 (cuantitativo) necesita tener la información correspondiente a la base y la altura del triángulo, pero también es necesaria la consistencia dimensional para presentar resultados coherentes. Por lo tanto, se podrían agregar pasos adicionales al procedimiento para evitar errores posteriores. Por ejemplo,

1. Inicio
2. Ingresar la base y unidad de medición.
3. Ingresar la altura y unidad de medición.
4. Verificar que las unidades sean consistentes. Si no lo son, hacer la conversión correspondiente.
5. Calcular el área
6. Mostrar resultado.
7. Fin

Es posible concluir, que un algoritmo puede ser tan detallado como se desee. Además, puede haber diferentes caminos (algoritmos) para alcanzar el mismo objetivo. En estos casos, siempre es conveniente seleccionar el algoritmo más corto, que utilice la menor cantidad de recursos posibles (tiempo, elementos, etc.) para alcanzar el objetivo.

Características de los algoritmos.

Un algoritmo se caracteriza por ser preciso, finito y definido.



REPRESENTACIÓN DE ALGORITMOS.

Cualquier algoritmo puede representarse utilizando diferentes tipos de técnicas, tales como:

- Pseudocódigo
- Diagramas de flujo
- Diagramas de Nassi-Shneiderman
- Lenguaje natural (español, inglés, etc.)
- Fórmulas matemáticas.

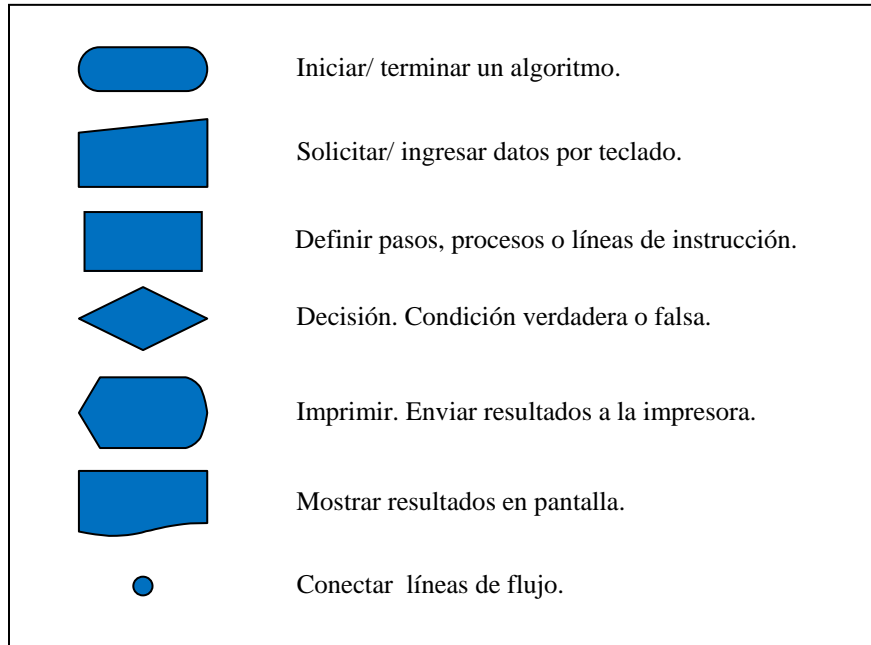
El lenguaje natural, empleado en los ejemplos anteriores, en muchas ocasiones puede no ser preciso y dar lugar a ambigüedades. En este curso, trataremos la representación mediante diagramas de flujo y pseudocódigo.

Diagramas de flujo.

Los diagramas de flujo son descripciones gráficas de algoritmos, que permiten al programador centrarse en los aspectos lógicos de la solución.

Se construyen utilizando ciertos símbolos de uso especial como ser rectángulos, rombos, óvalos, y pequeños círculos. Estos símbolos están conectados entre sí por flechas, conocidas como líneas de flujo para indicar la secuencia de instrucciones.

En la siguiente figura se presenta la simbología básica.



Para generar un diagrama de flujo se deben seguir las siguientes reglas:

- Los diagramas de flujo deben escribirse de arriba hacia abajo, y/o de izquierda a derecha.
- Los símbolos se unen con líneas, las cuales tienen en la punta una flecha que indica la dirección que fluye la información procesos, se deben de utilizar solamente líneas de flujo horizontal o verticales (nunca diagonales).
- Se debe evitar el cruce de líneas, para lo cual si se quisiera separar el flujo del diagrama a un sitio distinto, se debe realizar utilizando los conectores correspondientes. Se debe tener en cuenta que solo se van a utilizar conectores cuando sea estrictamente necesario.
- No deben quedar líneas de flujo sin conectar.
- Todo texto escrito dentro de un símbolo debe ser legible, preciso, evitando el uso de muchas palabras.
- Todos los símbolos pueden tener más de una línea de entrada, a excepción del símbolo final.
- Solo los símbolos de decisión pueden y deben tener más de una línea de flujo de salida.

Pseudocódigo.

Es la técnica que permite expresar la solución de un problema mediante un algoritmo escrito con las palabras normales de un idioma y utilizando palabras imperativas como: inicie, lea, imprima, calcule, finalice, etc. No hay un léxico obligado para el pseudocódigo, pero con el uso cotidiano se han establecido algunos estándares.

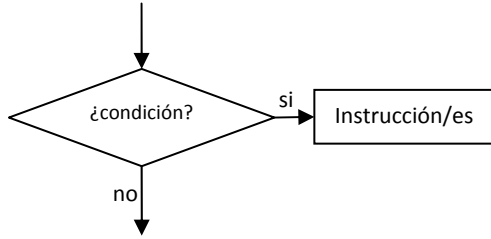
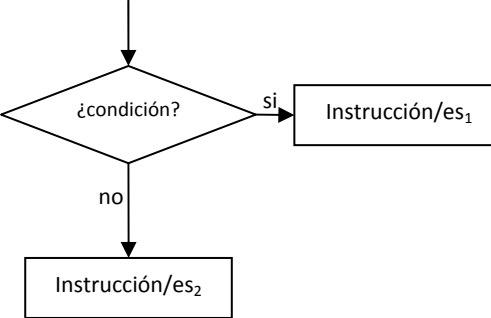
Ventajas del pseudocódigo sobre los diagramas de flujo

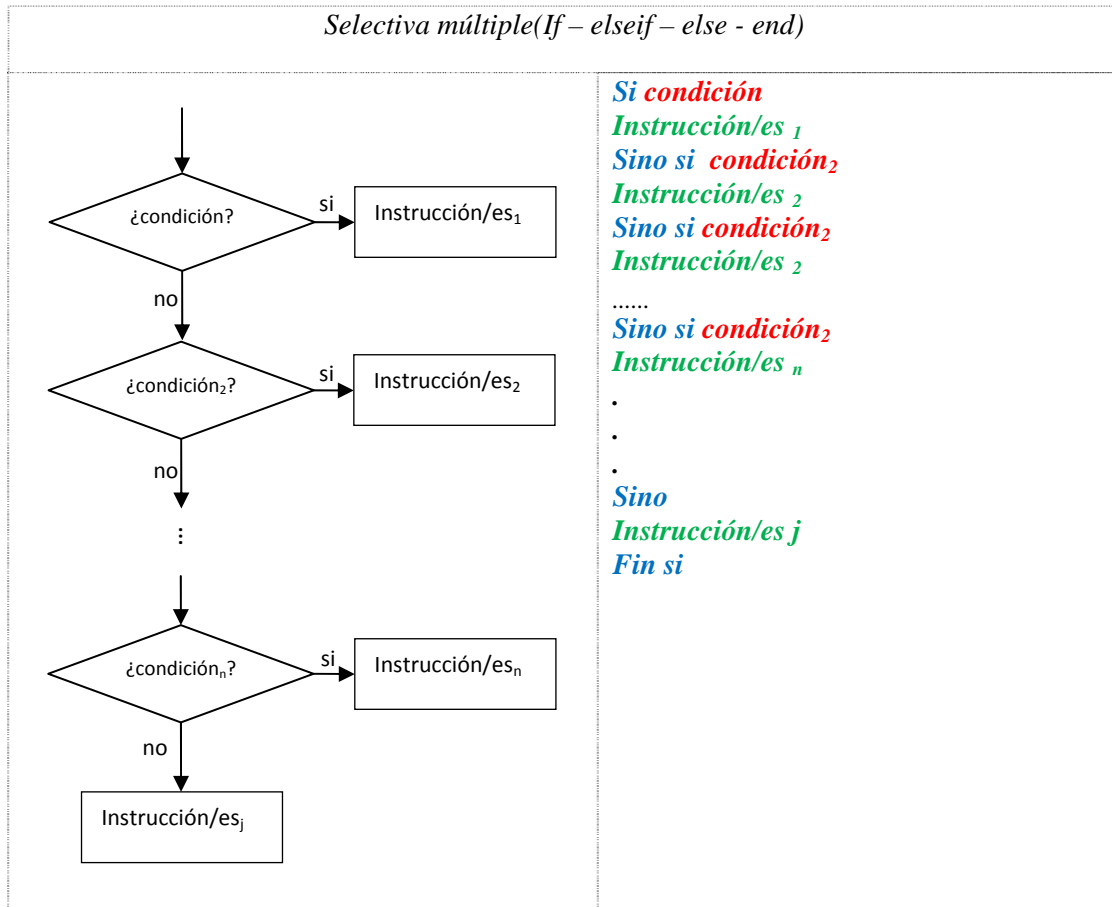
- Ocupan menos espacio en el desarrollo del problema.
- Permiten representar de forma fácil operaciones repetitivas complejas.
- Es más sencilla la tarea de pasar de pseudocódigo a un lenguaje de programación formal.

Algoritmos. Estructuras básicas.

Cualquier algoritmo con un solo punto de entrada y un solo punto de salida puede seguirse con tres tipos de estructuras de control:

- **Secuencial.** Es aquella que ejecuta las acciones sucesivamente, unas a continuación de otras, sin posibilidad de omitir ninguna.
- **Selectiva o condicional:** Es aquella en la que únicamente se realizan una o varias acciones dependiendo del valor de una condición determinada. Este tipo de estructura, también llamada de decisión, se clasifica en simple, doble o múltiple.

Diagrama de flujo.	Pseudocódigo
<i>Selectiva Simple (If - end)</i>	
	<p><i>Si condición</i> <i>Instrucción/es</i> <i>Fin si</i></p>
<i>Selectiva Doble (If- else - end)</i>	
	<p><i>Si condición</i> <i>Instrucción/es 1</i> <i>Sino</i> <i>Instrucción/es 2</i> <i>Fin si</i></p>



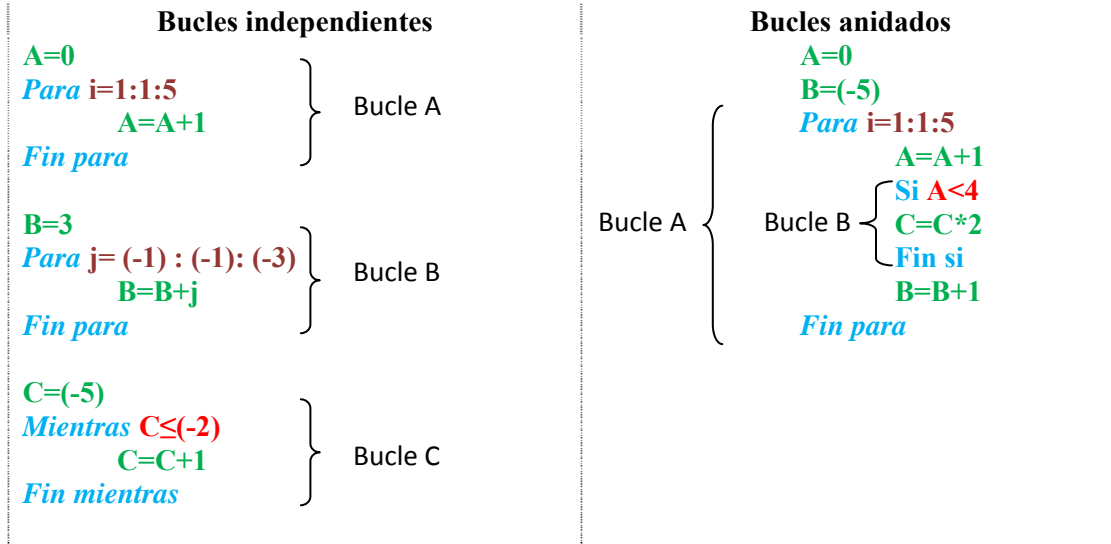
- **Repetitiva o iterativa:** son aquellas en las que las acciones se ejecutan un determinado número de veces y dependen de un valor predefinido o del cumplimiento de una determinada condición lógica. En la siguiente tabla, se presentan tres tipos de estructuras repetitivas:

- Repetir mientras.
- Repetir hasta
- Desde-hasta

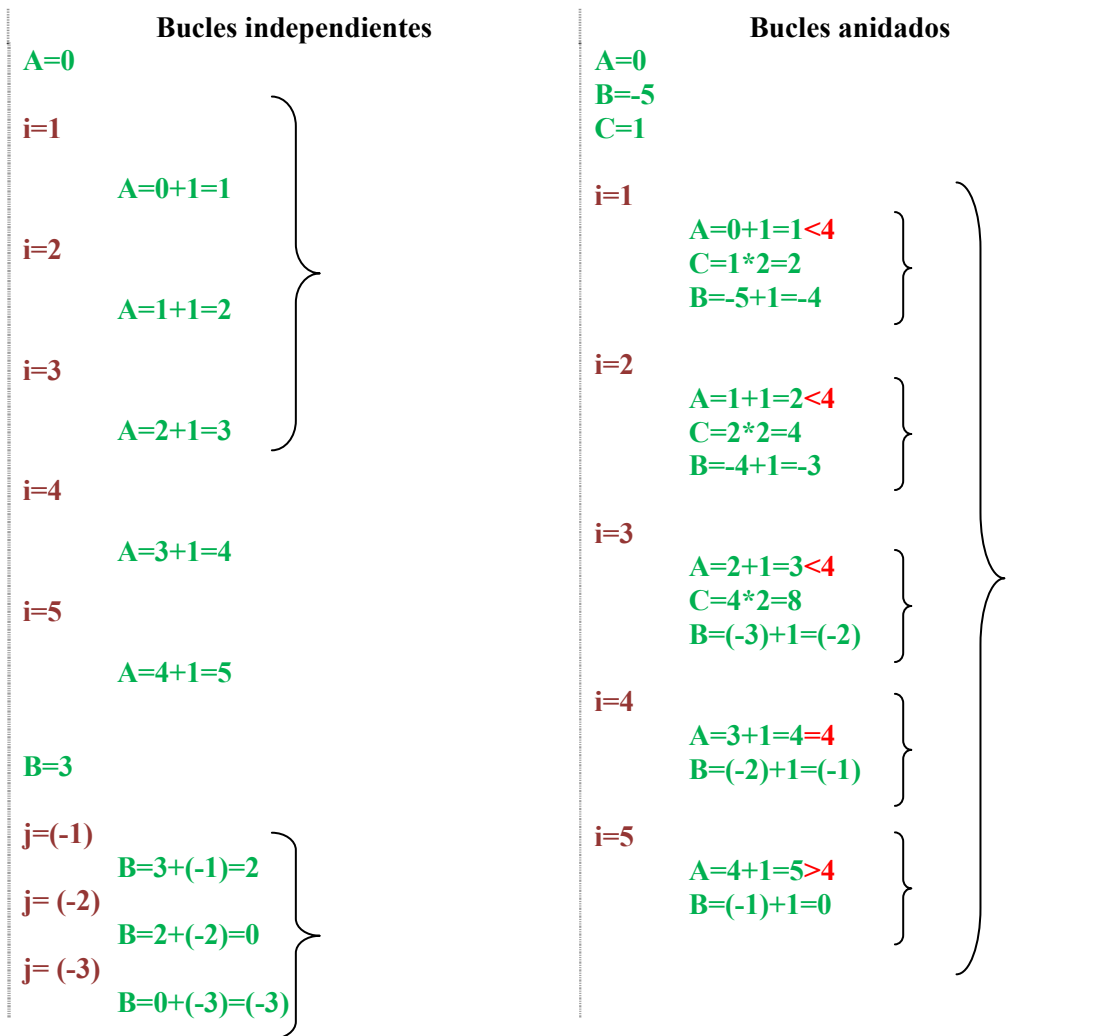
Diagrama de flujo	Pseudocódigo
<p>Bucle repetir mientras (<i>while-end</i>). La acción se repite mientras la condición sea verdadera. Cuando toma el valor “Falso” sale del bucle. Se utiliza cuando queremos repetir la ejecución de un paso un número indefinido de veces. La acción se ejecutará de 0 a n veces.</p>	

	<p><i>Mientras condición</i> <i>Instrucción/es</i> <i>Fin mientras</i></p>
<p>Bucle repetir hasta (do while-end). La acción se repite hasta que la condición se haga falsa, saliendo del bucle. La acción se ejecutará al menos una vez.</p>	
	<p><i>Repetir</i> <i>Instrucciones</i> <i>Hasta que condición</i></p>
<p>Bucle Desde - Hasta (for-end). Permite ejecutar un bloque de instrucciones un número determinado y conocido de veces. En este tipos de estructuras hay que definir el valor inicial, el paso y el valor final (i=valor inicial:paso:valor final). Siempre que no se indique el paso se asume que es igual a uno (1).</p>	
	<p><i>Para i=1 hasta i=n (i=1:1:n)</i> <i>Instrucciones</i> <i>Fin para</i></p>

- **Estructuras anidadas:** en un algoritmo, es posible encontrar estructuras independientes o dependientes (anidadas). Por ejemplo, en el algoritmo de la izquierda los bucles A, B y C son independientes entre sí, mientras que en el algoritmo de la derecha los bucles A y B son dependientes.



Veamos cómo serían los resultados paso a paso para cada caso:



```

C=(-5)
C=(-5)+1=(-4)≤(-2)
C=(-4)+1=(-3)≤(-2)
C=(-3)+1=(-2)≤(-2)
C=(-2)+1=(-1)>(-2)
    
```

Operadores lógicos.

En las estructuras de decisión se utilizan operadores lógicos. Este tipo de operadores se construyen con operadores relacionales y permiten comparar dos variables entre sí o una variable con un valor fijo. Solo admiten dos tipos de respuestas: Verdadero (1) o Falso (0).

- Igual
- Diferente
- Menor o igual
- Mayor o igual
- Menor
- Mayor
- Y
- O

Uso de arreglos matriciales para almacenamiento de datos.

En muchas ocasiones, se hace uso de arreglos matriciales para favorecer la organización de datos y resultados. Las posiciones de cada elemento en la estructura se identifican mediante índices. La posición de un elemento en un vector se identifica mediante un solo índice, mientras que en la matriz se identifica mediante dos índices.

$$u = [a_1 \quad a_2 \quad a_3 \quad a_4]$$

$$Q = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$u(3)$ es el elemento que se encuentra en la tercera fila del vector u.

$Q(3,2)$ es el elemento que se encuentra en la tercera fila, segunda columna de la matriz Q.

En el siguiente ejemplo, se desarrolla un algoritmo que permite calcular y almacenar el duplo de todos los números comprendidos entre 1 y un valor N ingresado por el usuario.

Opción 1	Opción 2
Ingrese N Para i=1:N Duplo(i)=2*i Fin para Mostrar Duplo	Ingrese N Para i=1:N Duplo=2*i Mostrar Duplo Fin para
N=4	
i=1 Duplo(1)=2 i=2 Duplo(2)=4 i=3 Duplo(3)=6 i=4 Duplo(4)=8 Duplo=[2 4 6 8]	i=1 Duplo=2 i=2 Duplo=4 i=3 Duplo=4 i=4 Duplo=4

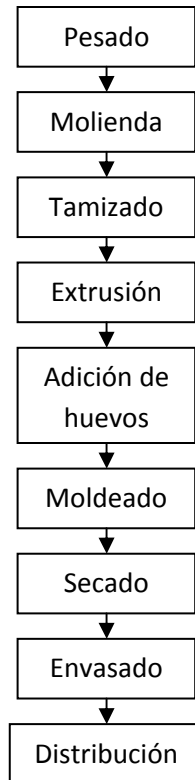
En la opción 1, en la que se utiliza un arreglo vectorial para almacenar valores, el resultado de multiplicar $2*i$ se almacena en la posición i del vector Duplo. De esta manera, al salir del bucle se tiene un vector que ha almacenado todos los valores calculados permitiendo mostrar el resultado al salir del bucle.

En la opción 2, el resultado de multiplicar $2*i$ se almacena en la variable Duplo, que es un escalar. Cada vez que se vuelve a entrar al bucle, el valor de la variable Duplo se modifica, haciendo necesario mostrar el resultado cada vez que se realiza un cálculo.

Algoritmos Cualitativos. Ejemplos.

Estructuras de secuenciación.

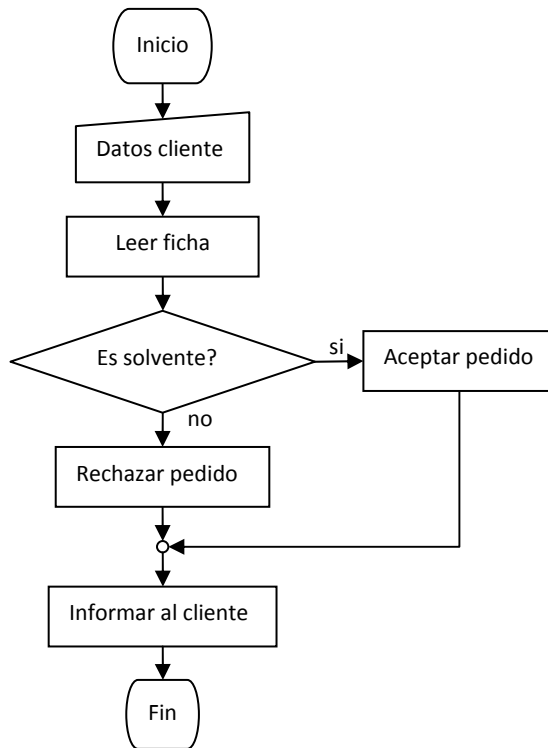
1. Desarrolle un diagrama de flujo que represente el proceso de obtención de fideos envasados a partir del trigo seleccionado que ingresa al primer equipo del proceso.



Estructuras de decisión.

2. Desarrolle un algoritmo que acepte o rechace el pedido realizado por un cliente de acuerdo a su solvencia. Considere que los datos de todos los clientes están cargados en formularios, de manera tal que usted ingresa los datos del cliente y tiene toda la información disponible.

Diagrama de flujo

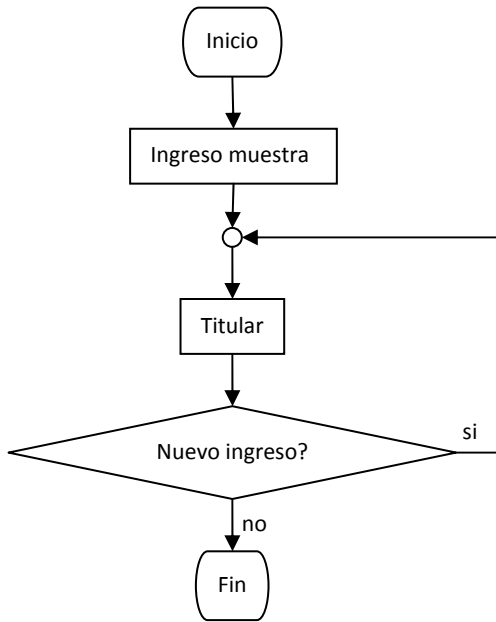


Pseudocódigo

Ingresar datos cliente
Leer ficha cliente
Si “cliente es solvente”
 Aceptar pedido
Sino
 Rechazar pedido
Fin si
Informar al cliente

Estructuras de repetición.

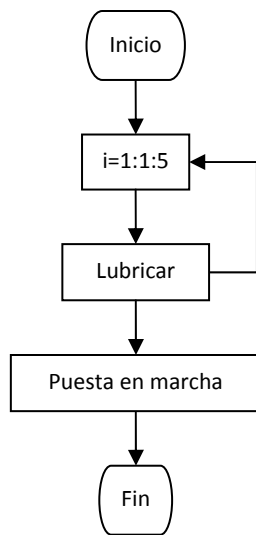
- Un laboratorista necesita titular todas las muestras que ingresan del departamento de control de calidad. Desarrollar un algoritmo para la situación presentada considerando que desconoce la cantidad de muestras que ingresarán al laboratorio.



*Mientras "ingrese muestra"
titular
Fin mientras*

La acción se repetirá hasta que la condición adopte un estado falso, de manera tal que si no ingresan más muestras el laboratorista terminará la tarea asignada.

- Un operario, encargado del sistema de bombeo, debe lubricar las 5 bombas antes de su puesta en marcha. Desarrolle un algoritmo que represente tal situación.

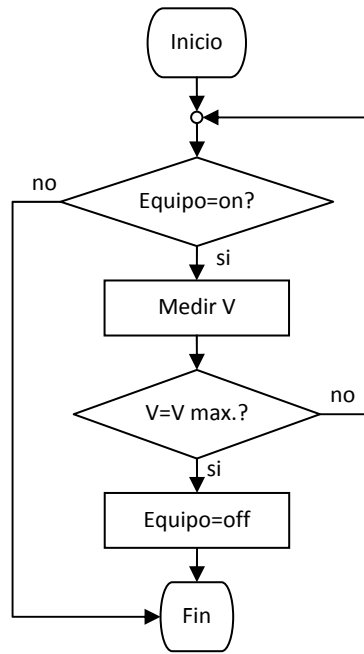


*Para i=1:1:5
lubricar
Fin para
Puesta en marcha*

Estructuras anidadas.

5. Debido a la necesidad de proteger un equipo frente a sobretensiones, se ha instalado un interruptor automático que lee el voltaje y apaga el equipo si el voltaje alcanza un límite de seguridad. Desarrolle un algoritmo en el que se represente la secuencia de control considerando que se trata de un interruptor de reposición manual.

Diagrama de flujo



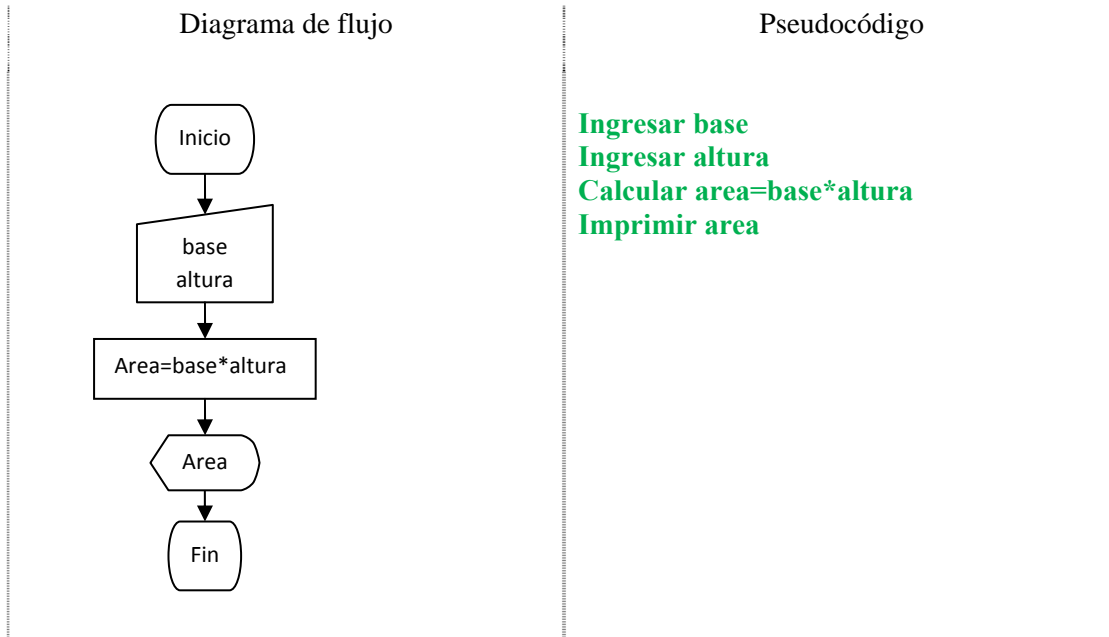
Pseudocódigo

```
Mientras "equipo=on"  
  Medir tensión  
  Si "tensión=tensión máx."  
    equipo=off  
  Fin si  
Fin mientras
```

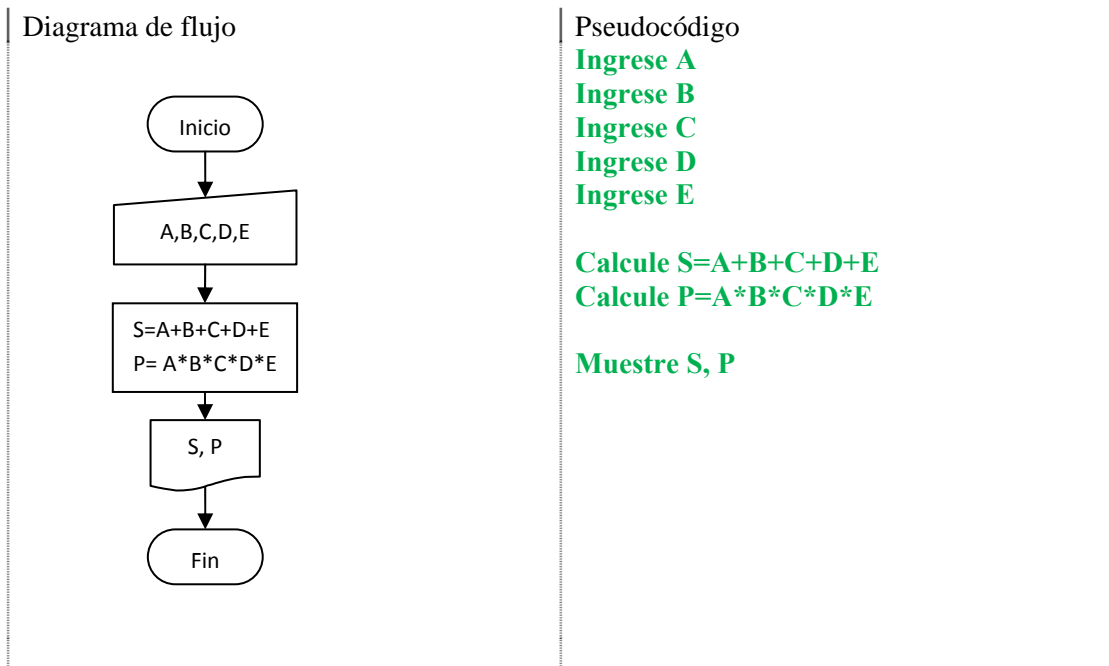

Algoritmos Cuantitativos. Ejemplos.

Estructuras secuenciales.

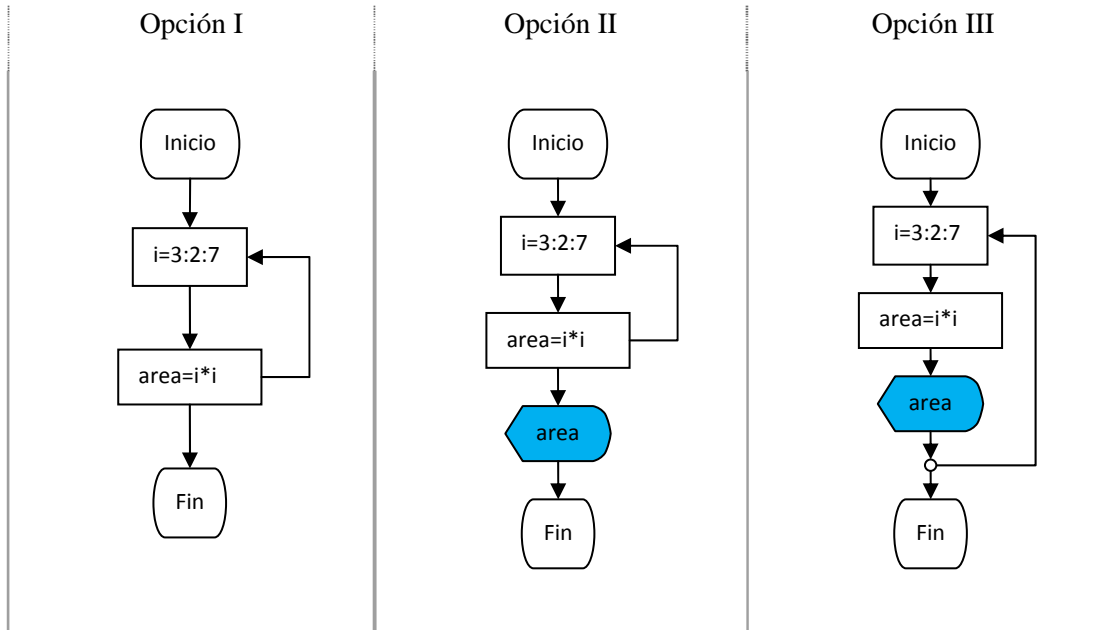
6. Desarrollar un algoritmo que permita calcular el área de un rectángulo a partir de los datos ingresados por el usuario (base y altura).



7. Desarrollar un algoritmo que pida al usuario que ingrese cinco números, se calcule la suma y el producto de dichos valores y se muestren los resultados en pantalla.



8. Los siguientes diagramas de flujo, representan un algoritmo que calcula el área de tres cuadrados de lados 3, 5 y 7 cm. Determine cuál de las tres estructuras es la correcta, si se pretende imprimir el área de los tres cuadrados considerados.



La expresión $i=3:2:7$ implica que la variable i adoptará valores mayores o iguales a tres y menores o iguales a 7 con paso de dos unidades, es decir $i=3, i=5, i=7$.

Para explicar que se hace en cada caso, se muestran cuales son los cálculos paso a paso.

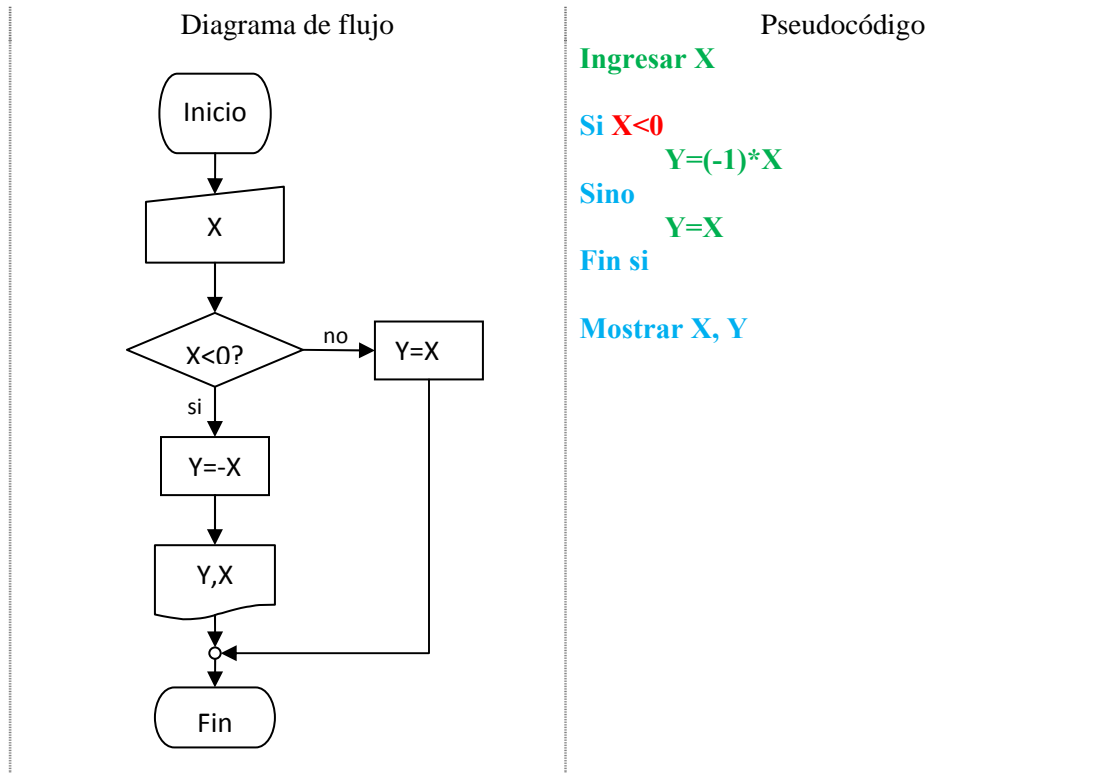
Opción I	Opción II	Opción III
Inicio algoritmo	Inicio algoritmo	Inicio algoritmo
Inicio bucle	Inicio bucle	Inicio bucle
$i=3$	$i=3$	$i=3$
$area=3*3=9$	$area=3*3=9$	$area=3*3=9$
$i=5$	$i=5$	imprime area
$area=5*5=25$	$area=5*5=25$	$i=5$
$i=7$	$i=7$	$area=5*5=25$
$area=7*7=49$	$area=7*7=49$	imprime area
fin bucle	fin bucle	$i=7$
Fin algoritmo	imprime area	$area=7*7=49$
	Fin algoritmo	imprime area
		fin bucle
		Fin algoritmo
Incorrecto	Incorrecto	Correcto

En todos los casos, se calcula correctamente el área. La opción I es incorrecta porque no muestra el resultado obtenido. La opción II es incorrecta porque solo muestra el resultado almacenado para el último valor de lado asignado. La opción III es la correcta ya que muestra el resultado

inmediatamente después de cada cálculo, antes de volver a entrar al bucle y sobre-escribir el resultado anterior. En este ejemplo se hace evidente que la estructura del algoritmo (diagrama o pseudocódigo) es fundamental.

Estructuras de selección.

- Desarrollar un algoritmo que cambie el signo de un número ingresado por el usuario solo en caso que sea negativo.



- Desarrolle paso a paso las acciones que se realizarán en los ejemplos desarrollados para los siguientes valores de X ingresados por el usuario: 5, 1, 14, -3.

Caso 1	Caso 2	Caso 2
<p>Ingrese X</p> <p>Si $X > 2$ $Y = X + 5$</p> <p>Fin si</p>	<p>Ingrese X</p> <p>Si $X > 2$ $Y = X + 5$</p> <p>Sino $Y = X * 2$</p> <p>Fin si</p>	<p>Ingrese X</p> <p>Si $0 > X > 2$ $Y = X + 5$</p> <p>Sino $7 < X < 12$ $Y = X * 2$</p> <p>Sino $X < 0$ $Y = X * 3 + 5$ $Z = X / 1000$</p> <p>Fin si</p>

X=5		
Y=5+5=10	Y=5+5=10	No se cumple la condición por lo que no se calcula nada.
X= 1		
No se cumple la condición por lo que no se calcula nada.	Y=1*2=2	Y=1+5=6
X= 14		
Y=5+5=10	Y=14+5=19	No se cumple la condición por lo que no se calcula nada.
X= -5		
No se cumple la condición por lo que no se calcula nada.	Y=(-5)*2=(-10)	Y=(-5)*3+5=(-10) Z=-0.005

Estructuras de repetición.

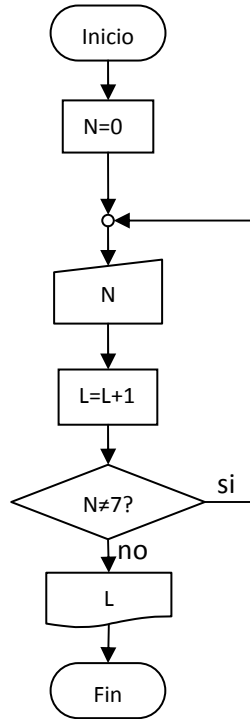
11. Desarrollar un algoritmo que pida al usuario que ingrese un número mientras el número sea distinto de 7.

<p style="text-align: center;">Diagrama de Flujo</p> <pre> graph TD Inicio([Inicio]) --> N0[N=0] N0 --> Conn(()) Conn --> N[/N/] N --> Cond{N≠7?} Cond -- si --> Conn Cond -- no --> Fin([Fin]) </pre>	<p style="text-align: center;">Pseudocódigo</p> <pre> Defina N=0 Mientras N≠7 Ingrese N Fin mientras </pre>
--	--

Intente explicar cuál es el objetivo de definir N (N=0) antes de establecer la condición.

El siguiente paso consiste en modificar el algoritmo de manera tal que le permita calcular y mostrar la cantidad de veces que se ha ingresado un número.

Diagrama de flujo



Pseudocódigo

```

Defina N=0
Defina L=0
Mientras N≠7
  Ingrese N
  Calcule L=L+1
Fin mientras
Muestre L
  
```

12. Se desea diseñar un algoritmo que a partir de un número (N) ingresado por el usuario, muestre números sucesivos menores que N. Asuma que siempre se ingresa un número entero positivo. Decida cuál de las siguientes opciones es la correcta.

Opción 1	Opción 2	Opción 3	Opción 4	Opción 5
Ingrese N i=1 Mientras i<N Mostrar i i=i+1 fin mientras	Ingrese N i=1 Mientras i<N i=i+1 Mostrar i fin mientras	Ingrese N i=0 Mientras i<N i=i+1 Mostrar i fin mientras	Ingrese N Para i=1:(N-1) Mostrar i Fin para	Ingrese N Para i=1:N Mostrar i Fin para
N=5				
i=1 2 3 4 Correcta	i=2 3 4 5 Incorrecta	i= 1 2 3 4 5 Incorrecta	i= 1 2 3 4 Correcta	I=1 2 3 4 5 Incorrecta

13. Se desea diseñar un algoritmo que a partir de un número (N) ingresado por el usuario, muestre números sucesivos en orden decreciente hasta 1. Asuma que siempre se ingresa un número entero positivo.

Opción 1	Opción 2
Ingrese N Para i=N:-1:1 Mostrar i fin para	Ingrese N i=N Mientras i≤N Y i≥1 Mostrar i i=i-1 fin mientras
N=5	
i= 5 4 3 2 1	i=5 4 3 2 1