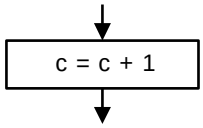
Capítulo 4
Procesos con estructuras de repetición

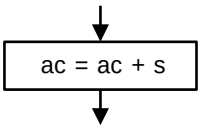
Estructura de contador



Esta es una operación que incrementa en una unidad el valor almacenado en la variable c, cada vez que el flujo del diagrama pasa por esta estructura.

Su secuencia de cálculo es igual a las vistas anteriormente, es decir, primero se resuelve la operación a la derecha del signo igual y después se transfiere el resultado obtenido a la variable que se muestra a la izquierda del signo igual.
En el caso presentado el incremento del conteo es igual a uno, pero en otras estructuras podría ser necesario un conteo descendente por ejemplo con –1, o bien un conteo con un valor constante h diferente de 1.

Estructura de acumulador

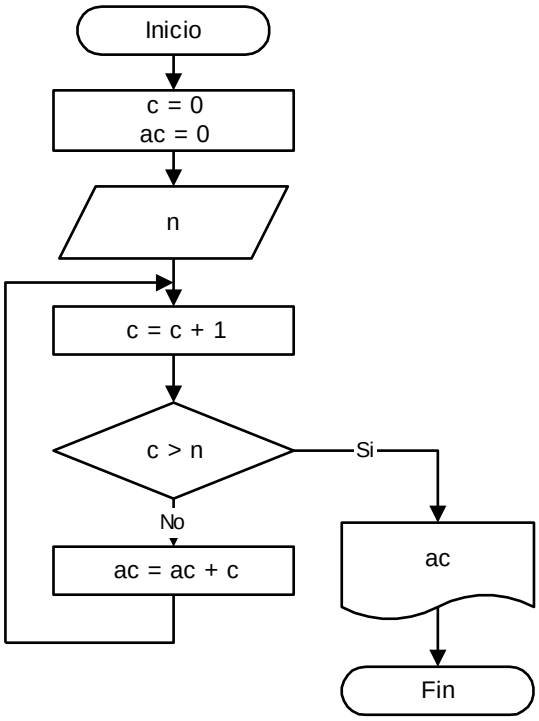


En esta expresión se efectúa una acumulación de un valor s en la variable ac, cada vez que el flujo del programa encuentra esta estructura.

El valor que toma s en cada acumulación es esencialmente variable, lo que diferencia este caso de la estructura de contador vista anteriormente.
Cada valor de s irá incrementando el valor de ac, resolviendo en primer lugar la operación a la derecha del signo igual y transfiriendo luego el resultado a la variable ac.

Ejercicio 4.1:

Efectuar el diagrama de flujo de un programa que acumule la suma de los primeros n números naturales e imprima su resultado.



Prueba de escritorio:

n	c	ac
4	0	0
	1	1
	2	3
	3	6
	4	10
	5	

Salida por pantalla

10

Ejercicio 4.2:

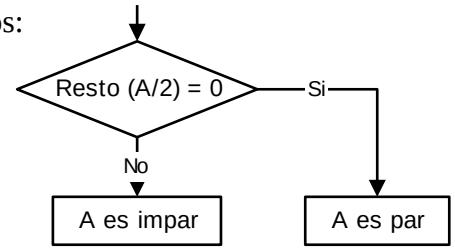
Efectuar el diagrama de flujo de un programa que sume los números impares comprendidos en el intervalo (a,b) e imprima el resultado.

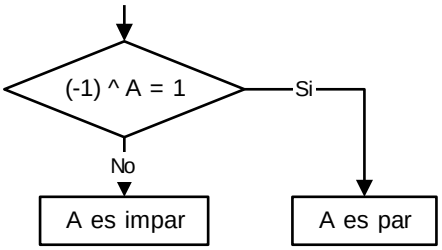
Para determinar si un número es par o impar tenemos dos métodos que se pueden utilizar en los diagramas de flujo. Los métodos consisten en:

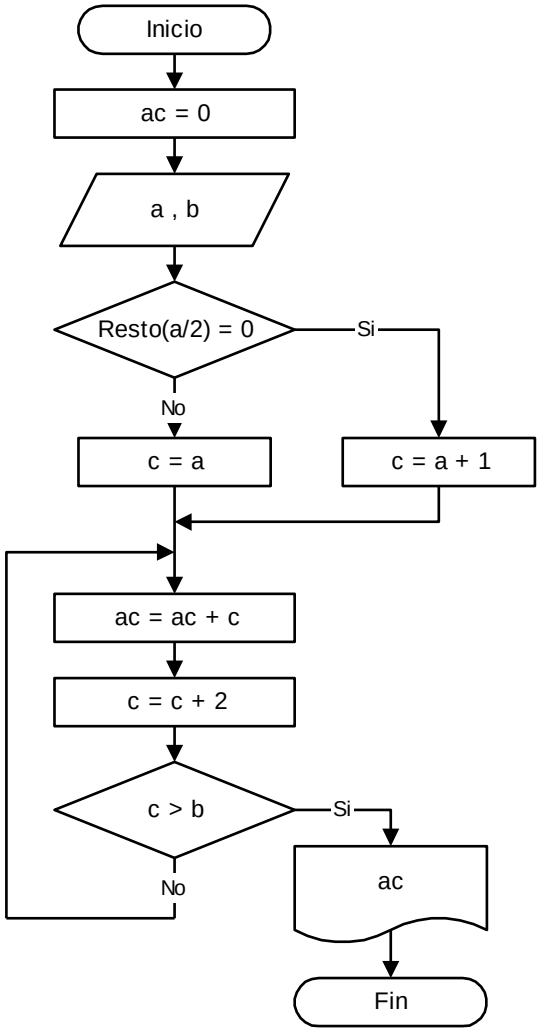
1.- Comparar el resto de la división por 2 : Si es cero = el número es par
Si no es cero = el número es impar

2.- Comparar el resultado de elevar –1 a la potencia
usando como exponente el número a averiguar: Si es 1 = el número es par
Si es –1 = el número es impar

Ejemplos:







Prueba de escritorio:

a	b	c	ac
2	8		0
		3	3
		5	8
		7	15
		9	

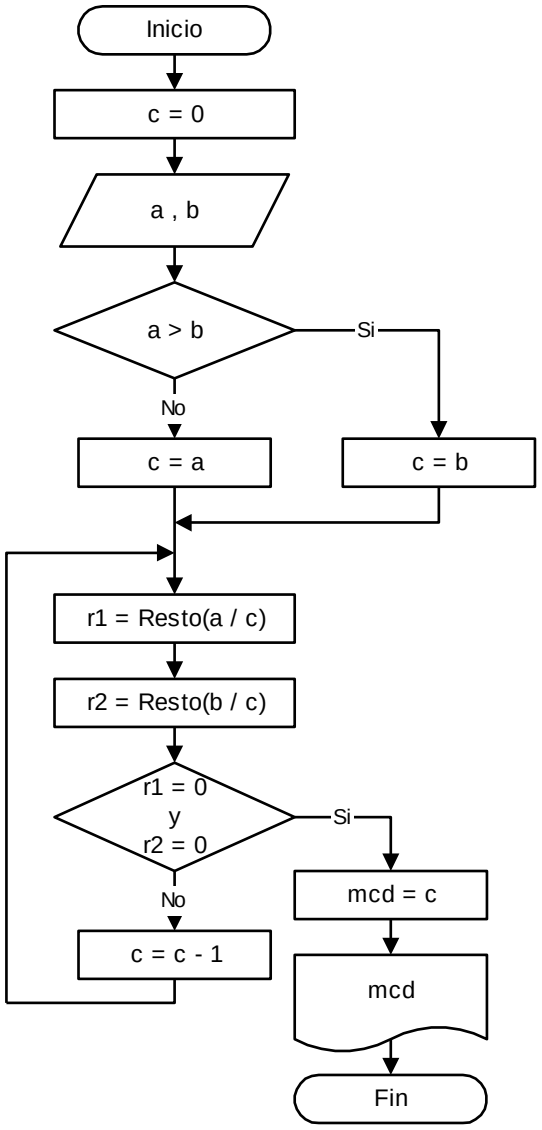
Salida por pantalla

15

Ejercicio 4.3:

Efectuar el diagrama de flujo de un programa que calcule el máximo común divisor mcd de dos números naturales a y b, e imprima el resultado.

Obtendremos el mcd mediante el siguiente procedimiento:
- Determinar el menor valor entre a y b.
- Verificar si ese valor ya es el mcd, es decir, si es divisible el mayor valor por el menor valor.
- Si no lo es, se utiliza un contador c decreciente.
- Se verifica para cada valor de c si los números a y b son divisibles por el contador c.
- Si no son divisibles, se decrementa el valor de c.
- En el caso en que los dos números a y b sean divisibles por el contador c, estamos en presencia del mcd que es igual a dicho valor de c.
- Se imprime el resultado.



Prueba de escritorio:

c	a	b	r1	r2	mcd
0	4	10			
4			0	2	
3			1	1	
2			0	0	2

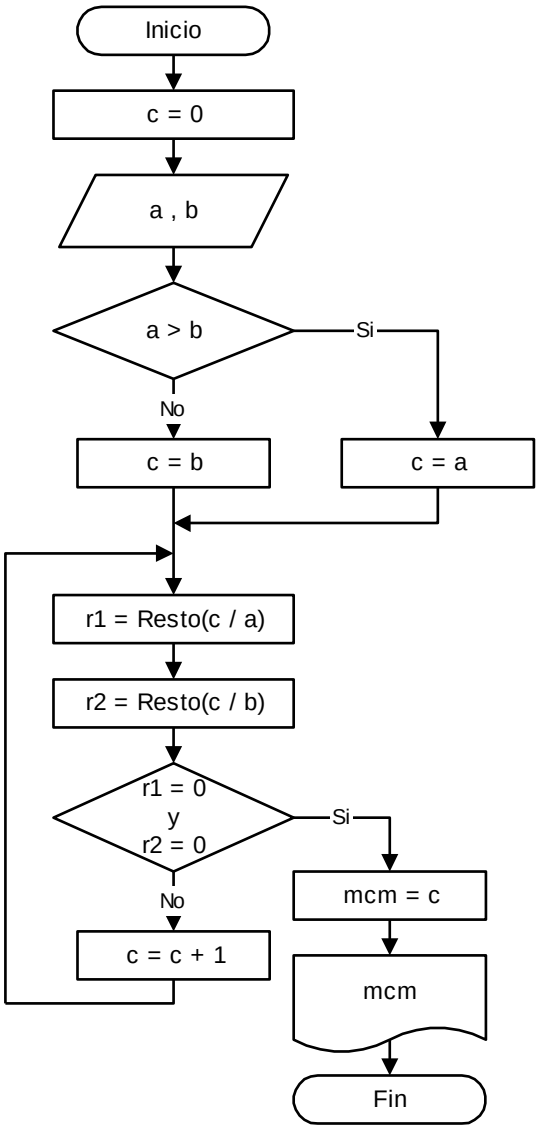
Salida por pantalla

2

Ejercicio 4.4:

Efectuar el diagrama de flujo de un programa que calcule el mínimo común múltiplo mcm de dos números naturales a y b, e imprima el resultado.

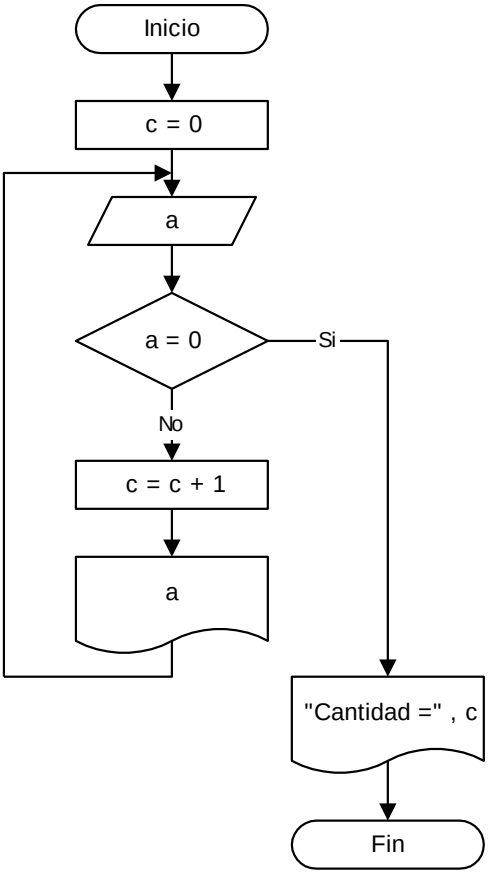
En este caso el procedimiento es similar, pero tomando el mayor de los dos números, que será el límite inferior del contador c creciente. Se verifica en cada caso si el contador c es divisible por los dos números a y b, y en caso afirmativo estaremos en presencia del mcm.
El contador será ascendente y su límite superior será a.b.



Nota: Es importante aclarar que las dos expresiones vistas de contador y acumulador no serían válidas desde el punto de vista matemático, ya que plantean una igualdad que no se puede conseguir, pero en computación, donde el signo igual significa transferencia, son totalmente válidas e inclusive muy utilizadas por su utilidad en conteos, acumulación de valores, aumento del valor de una variable x con incremento h, etc.

Ejercicio 4.5:

Efectuar el diagrama de flujo de un programa que lea por teclado e imprima una lista de números enteros distintos de cero. El proceso debe terminar con un valor igual a cero que no se debe imprimir. Imprimir además el número de valores leídos.



Prueba de escritorio:

a	c
	0
4	1
7	2
5	3
9	4
2	5
0	

Salida por pantalla

```
4
7
5
9
2
Cantidad = 5
```

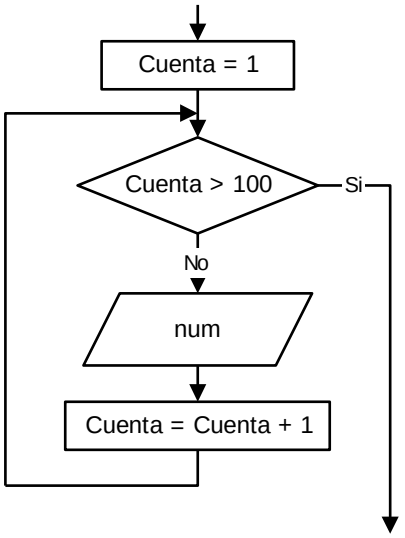
Control de las estructuras de repetición

Bucles controlados por contador

Un **bucle controlado por contador** es un bucle que se ejecuta un número especificado de veces. Es necesario para el diseño correcto del bucle un mecanismo que cuente el número de veces que se ejecuta el cuerpo del bucle. El mecanismo que se utiliza es una *variable de control del bucle* que actúa como un *contador*.
Un bucle controlado por contador consta de tres partes, además del cuerpo y de la condición de salida:

- Inicialización de la variable de control del bucle.
- Comprobación del valor de la variable de control del bucle o de la condición de salida.
- Incremento del valor de la variable de control del bucle.

El siguiente segmento de diagrama de flujo muestra un bucle que cuenta 100 números leídos desde el teclado, con las aclaraciones de los pasos importantes en el bucle.



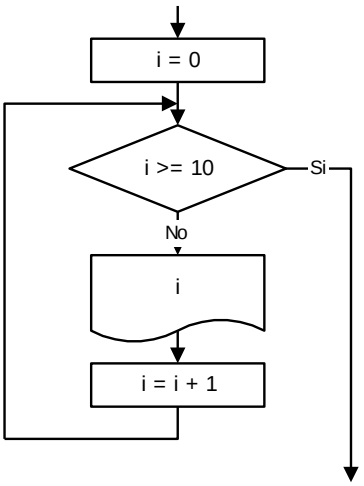
- Inicialización de cuenta, variable de control de bucle

- Condición de salida

- Incremento de la variable de control

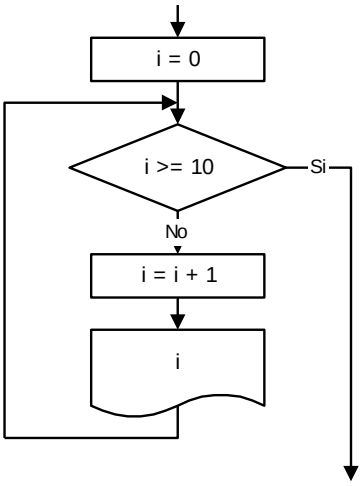
El cuerpo del bucle se repite mientras la condición sea falsa. Inicialmente el contador (variable *Cuenta*) es igual a 1, y por lo tanto la condición es falsa: el cuerpo del bucle se ejecutará. A continuación la variable *Cuenta* se incrementa a 2 y la condición sigue siendo falsa. El cuerpo del bucle se repite 100 veces y en ese momento la variable *Cuenta* toma el valor 101, por lo tanto la condición se vuelve verdadera y el bucle termina saliendo por la rama del Si.
Las variables de control del bucle son siempre contadores. Normalmente los contadores se inicializan en 0 o 1, La decisión de inicializar el contador a 0 o 1 depende del diseño del programador o del problema en cuestión. En el uso de contadores se deben considerar al menos tres factores: 1) el valor inicial, 2) el valor final, y 3) el operador relacional utilizado para comprobar la terminación del bucle.
La ubicación física de la sentencia que modifica el valor de la variable de control del bucle depende de su diseño, y su posición normalmente influye en el número de iteraciones que realiza el bucle.

Comparación de diferentes tipos de bucles controlados por contador



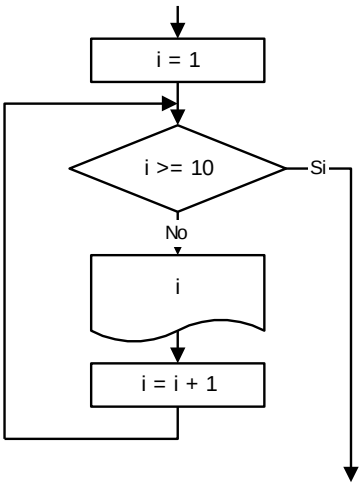

```
0
1
2
3
4
5
6
7
8
9
```

La salida por pantalla es el valor del contador del bucle *i*. El bucle se ejecuta 10 veces, hasta que *i*=10, y la condición se hace verdadera.




```
1
2
3
4
5
6
7
8
9
10
```

El cuerpo del bucle se ejecuta 10 veces pero su salida es diferente al caso anterior. Cuando i=10 el bucle se ejecuta, se efectúa la impresión, y al llegar a la condición de salida y ser verdadera, el bucle finaliza.




```
1
2
3
4
5
6
7
8
9
```

El bucle sólo se ejecuta 9 veces, como se puede apreciar en la salida por pantalla. Las únicas diferencias radican en el valor inicial de la variable *i* y la ubicación del contador *i=i+1*.

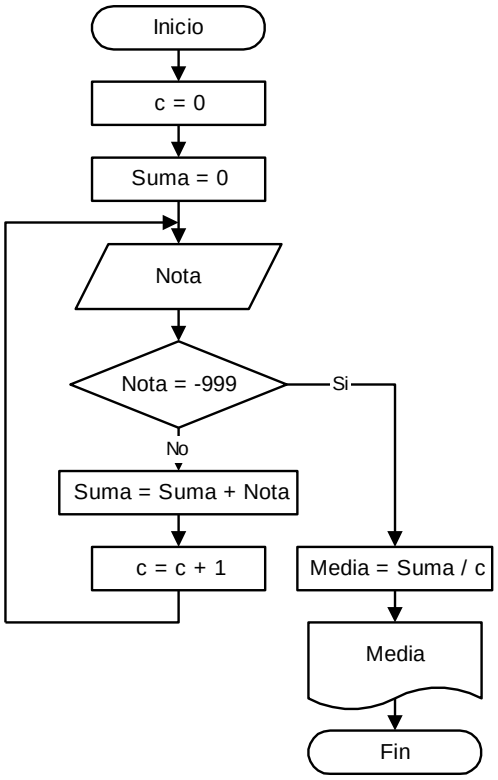
Los ejemplos anteriores han mostrado principalmente el mecanismo para modificar el flujo de control de los programas. Sin embargo, los bucles sirven fundamentalmente para realizar tareas a fin de que puedan ser útiles. La complejidad de la tarea a realizar dependerá del problema y del diseño eficiente del bucle, es decir que el gráfico de impresión por pantalla de los ejemplos dados anteriormente será reemplazado por un conjunto de instrucciones de acuerdo a cada caso en particular.

Bucles controlados por centinela

Los bucles se utilizan con frecuencia para leer y procesar listas de datos sin conocer el número de ellos por anticipado. Cada vez que una iteración se ejecuta, se realiza una nueva lectura de datos. El problema consiste en determinar cuándo se termina el proceso de lectura y se sale del bucle. Una solución al problema es el uso de un valor especial de datos denominado *centinela*.
Un **centinela** es un valor especial utilizado para indicar el final de una lista de datos y que *no* se procesa como un dato válido. El valor centinela es la acción o suceso que controla el bucle.
El bucle se ejecuta mientras no se lea el valor centinela. En el momento que se lee el valor centinela y éste se reconoce en la condición inicial, se sale del bucle.
Uno de los problemas que se plantean en el diseño del tipo de bucles controlado por centinela es precisamente la elección adecuada del valor centinela. Muchas veces el problema fija cuál es el centinela. Por ejemplo, si el problema no permite números de valores iguales a cero, entonces el cero será el valor centinela. Otro ejemplo puede ser el caso de procesar valores enteros que representen números de identificación de alumnos o trabajadores de una empresa; como en este caso los números deben ser mayores o iguales a cero, cualquier número negativo o cero puede ser tomado como centinela. Un ejemplo más es el proceso de edades de personas, que puede suponerse razonablemente que estarán entre 0 y 150 (siendo inclusive 150 un valor ya muy alto), por ello cualquier número negativo o un número positivo superior, como ser 999, etc, puede ser el centinela.
Los requisitos que se pueden fijar para ser considerados los valores como centinela son:

- El valor centinela debe ser único, es decir, debe ser fácilmente diferenciable del resto.
- El centinela debe ser del mismo tipo que los valores reales de los datos, de modo que pueda leerse con la misma sentencia de entrada.
- El valor realmente utilizado para terminar la entrada de datos debe ser el mismo valor esperado por el programa.

Ejemplo de bucle controlado por centinela



Se deben leer las calificaciones de los diferentes estudiantes de una clase, cuyas notas están en el rango de 0 a 100. Determinar y visualizar la nota promedio de la clase.

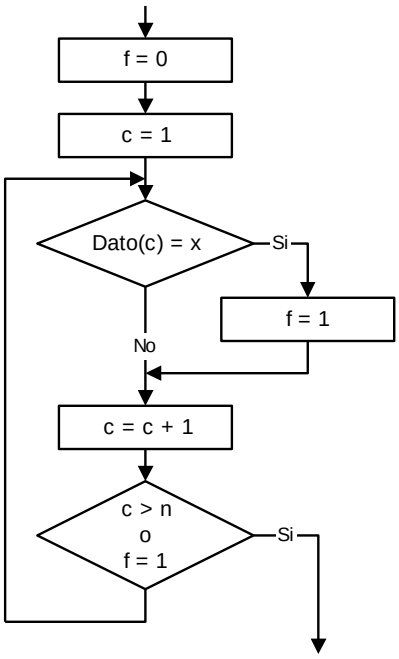
La condición estipulada de salida será el ingreso de un valor igual a –999, es decir, el centinela del problema es el número –999. Además de ello, al no conocer de antemano el número de notas a ingresar se coloca un contador *c*, y con el ingreso de cada nota válida se activa el conteo *c=c+1* y la acumulación de las notas en una variable *Suma*. Cuando se ingresa el valor centinela se sale del bucle y se procede a calcular el promedio, dividiendo el valor acumulado en *Suma* y la cantidad de notas almacenada en la variable *c*. Por último se produce la visualización del valor de la variable *Media*.

Bucles controlados por bandera o interruptor

Una **bandera**, **indicador** o **interruptor** (flag) es una variable que se utiliza para conservar el estado (verdadero o falso) de una condición. Se denomina bandera o interruptor debido a que puede ser asociado a un interruptor (encendido / apagado) o a una bandera (arriba / abajo). Se pueden utilizar variables lógicas cuando el lenguaje las provea, o bien se pueden simular con variables enteras a las que se le asignan valores 0 (falso) o 1/-1 (verdadero).
Con frecuencia se desea repetir una serie de acciones mientras que una determinada condición sea verdadera. Entonces el bucle se controla por el valor verdadero de la condición. Se puede utilizar una bandera de programa para almacenar este valor verdadero.

Ejemplo de bucle controlado por bandera

Supongamos que se necesita un proceso para buscar un valor x en una lista de n datos, que están almacenados en un vector Dato( ). La búsqueda debe terminar si se encuentra el valor x o bien si la lista de datos se agota.



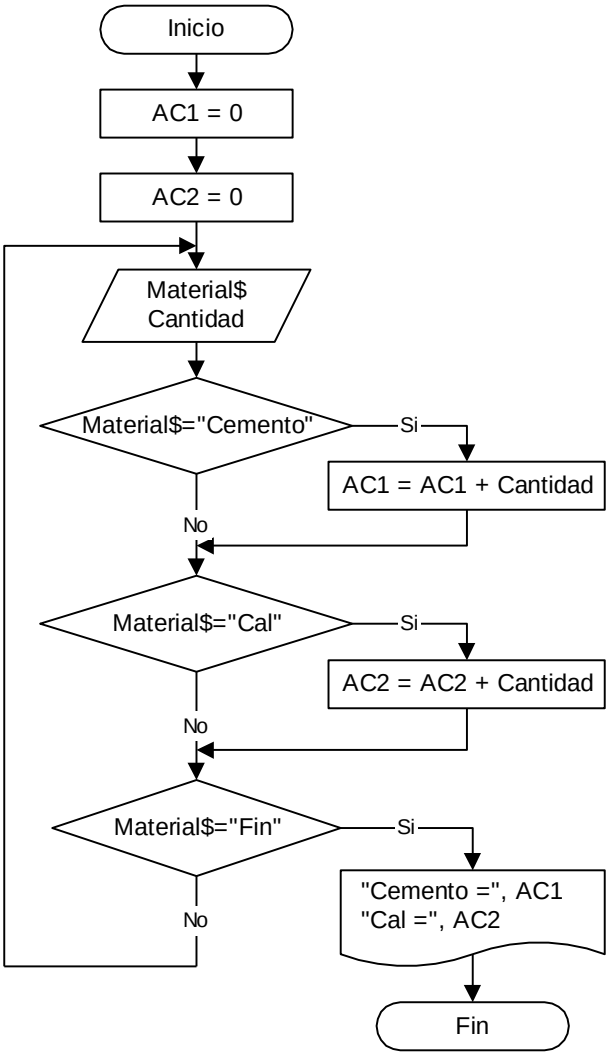
- La variable bandera f comienza el proceso con un 0.

- Si se encuentra el valor x dentro del vector Dato( ) la variable bandera f cambia su valor a 1.

- La condición de salida del bucle será la comparación de la cantidad de datos leídos y además la verificación del valor bandera almacenado en la variable f.

Ejercicio 4.6:

Efectuar el diagrama de flujo de un programa que lea por teclado un material de construcción y la cantidad de bolsas compradas de cada uno, acumule las distintas cantidades por separado y termine con la cadena alfanumérica “Fin”. Imprimir los resultados.



Prueba de escritorio:

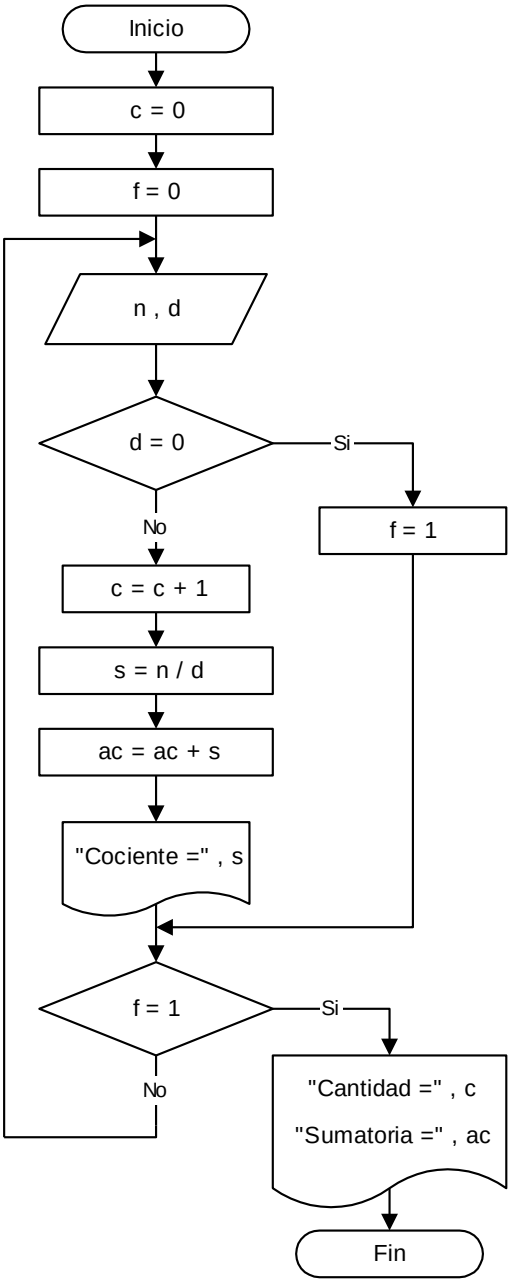
Material$	Cantidad	AC1	AC2	.
“Cemento”	10	10		
“Cal”	25		25	
“Cal”	10		35	
“Cal”	30		65	
“Cemento”	15	25		
“Cemento”	10	35		
“Fin”				

Salida por pantalla

```
Cemento = 35
Cal = 65
```

Ejercicio 4.7:

Efectuar el diagrama de flujo de un programa que lea por teclado un numerador n y un denominador d, haciendo el cociente para cada caso y almacenando los resultados en un acumulador ac, manteniendo una variable bandera f = 0 mientras se pueda hacer la división, hasta la situación en que el usuario ingrese un denominador d = 0, lo cual provocaría un error matemático originado por una división por cero. En tal caso, controlando el valor ingresado en la variable d, se debe modificar el valor de la bandera a f = 1, lo cual además dará fin al ciclo repetitivo. Al salir del ciclo repetitivo se debe imprimir el valor del acumulador ac y la cantidad c de divisiones válidas.



Ejercicios Adicionales Capítulo 4:

1. Realizar el diagrama de flujo que efectúe la suma de los primeros números hasta el 200 inclusive. Imprimir la suma.

2. Realizar el diagrama de flujo que efectúe la suma de los números pares hasta 200 inclusive. Imprimir la suma.

3. Realizar el diagrama de flujo que lea por teclado e imprima una serie de números distintos de cero. El proceso debe terminar con un valor cero que no se debe imprimir. Imprimir además el número de valores leídos.

4. Realizar el diagrama de flujo para calcular la velocidad (m/seg) de cada corredor de una carrera de 1500 mts. Las entradas consistirán en parejas de números (minutos, segundos) que dan el tiempo del corredor, para cada corredor. El proceso debe imprimir el tiempo en minutos y segundos, así como la velocidad media de cada corredor. El fin del proceso se producirá cuando la pareja de valores de tiempo sea (0,0).

5. Realizar el diagrama de flujo que determine el menor valor y el mayor valor de una lista de 25 números leídos desde el teclado.

6. Realizar el diagrama de flujo de un procedimiento que genere 10 números aleatorios y los imprima.

7. Realizar el diagrama de flujo de un procedimiento que genere números al azar entre 0 y 10, y cuya salida se produzca con el número 0.

8. Realizar el diagrama de flujo de un procedimiento que genere 10 números aleatorios entre 1 y 6 inclusive, simulando 10 tiradas de un dado.