



“La computadora nació para
resolver problemas que
antes no existían”.

Bill Gates.

Sello Editorial:
Fundación para la
Educación Superior San Mateo

Bogotá D.C. Colombia

Edward Reyes Corredor

APRENDER A PROGRAMAR

Edward Reyes Corredor

APRENDER A PROGRAMAR



0010101100111101
010001110100010
11010011010101
00100110100110



COLECCIÓN DIDÁCTICA
PROGRAMA DE SISTEMAS Y TELECOMUNICACIONES

APRENDER A PROGRAMAR

| | | |
|--------|--|----|
| 4. | Papel del Procesador en la programación_____ | 26 |
| 5. | Papel de la Memoria RAM en la programación_____ | 28 |
| 5.1 | Ubicación de los datos en Memoria RAM_____ | 29 |
| 6. | Papel de la Memoria ROM durante la ejecución de un programa _____ | 30 |
| 7. | Campo (variable) _____ | 30 |
| 7.1 | Como definir (crear) un campo _____ | 32 |
| 8. | Constantes _____ | 33 |
| 9. | Operaciones de entrada (lectura) _____ | 34 |
| 10. | Operaciones de salida (escriba) _____ | 35 |
| 10.1 | Cosas a tener en cuenta _____ | 36 |
| 11. | Que es un Algoritmo y para que nos sirven _____ | 37 |
| 11.1 | Como dar solución a los Algoritmos _____ | 39 |
| 11.1.1 | Diagramas de Flujo _____ | 39 |
| 11.1.2 | Reglas para trabajar con Diagramas de Flujo _____ | 45 |
| 11.2 | Operaciones Lógicas _____ | 51 |
| 11.3 | Operaciones de repetición _____ | 61 |
| 11.3.1 | Estructura desde _____ | 61 |
| 11.3.2 | Estructura mientras _____ | 65 |
| 11.3.3 | Estructura repetir/hasta _____ | 69 |
| 11.3.4 | Estructura iterar _____ | 71 |
| | Ejercicios _____ | 73 |
| | Codificación _____ | 79 |
| | Programa 1 en Java _____ | 86 |
| | Ejemplos en Pseudocódigo _____ | 90 |
| | Ejercicios propuestos _____ | 97 |
| | Bibliografía _____ | |

EDWARD REYES CORREDOR

APRENDER A PROGRAMAR

FUNDACIÓN PARA LA EDUCACIÓN SUPERIOR
SAN MATEO
2008

Catalogación en la publicación Fundación para la Educación Superior San Mateo

Programación. Integración didáctica entre conceptos básicos adquiridos por ejercicios de lógica y elementos para aprender a programar, lenguajes, campos, algoritmos, diagramas de flujo y estructuras.

98 p.

ISBN 978-958-98600-2-1

Construcción de programas - Lenguajes de programación - Memorias RAM y ROM - Estructuras Iterar y Programación ejercicios en Java.

© FUNDACIÓN PARA LA EDUCACIÓN SUPERIOR SAN MATEO
© Edward Reyes Corredor
Programa de Sistemas y Telecomunicaciones

ISBN 978-958-98600-2-1

Concepto gráfico y propuesta de portada

Gonzalo Garavito Silva.

Diagramación

María Fernanda Garavito Santos.

Impresión

FOCO Ediciones Bogotá - Colombia

Sello Edit. Fundación para la Educación Superior San Mateo (958-98600)

Todos los derechos reservados. Bajo las sanciones establecidas en las leyes, queda rigurosamente prohibida, sin autorización escrita de los titulares del copyright, la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la reprografía y el tratamiento informático.

INDICE

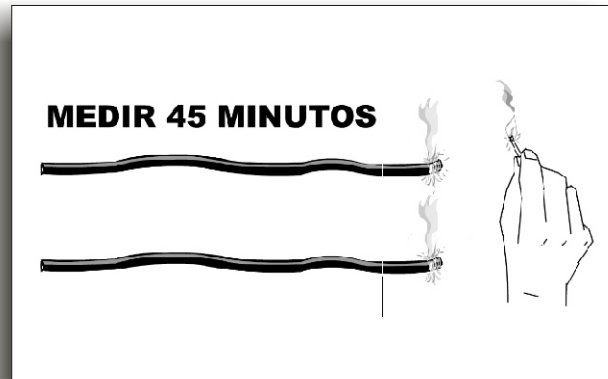
| | |
|--|----|
| Introducción _____ | 9 |
| Juegos para desarrollar la lógica _____ | 11 |
| Juego 1: Cruzar el río _____ | 11 |
| Juego 2: Las dos mechas _____ | 12 |
| Juego 3: La mosca saltarina _____ | 12 |
| Juego 4: Los cacahuets del mono _____ | 13 |
| Juego 5: La botella de vino _____ | 14 |
| Juego 6: Las Puertas _____ | 14 |
| Juego 7: Masacre en prisión _____ | 15 |
| Juego 8: La bombilla _____ | 16 |
| Juego 9: Convento de monjas _____ | 16 |
| Juego 10: Entre números _____ | 17 |
| Juego 11: Expediente X _____ | 18 |
| Juego 12: La linterna _____ | 19 |
| Juego 13: Bolas de billar _____ | 19 |
| Juego 14: Reordenación _____ | 20 |
| Juego 15: El Reloj de pared _____ | 20 |
| Juego 16: ¿64 ó 65 cuadrados? _____ | 20 |
| Juego 17: Viaje galáctico _____ | 21 |
| El Mundo de la Programación _____ | 23 |
| 1. Programa _____ | 23 |
| 2. Pasos a tener en cuenta para la construcción de un programa _____ | 23 |
| 3. Lenguajes de Programación _____ | 25 |

JUEGO 2: LAS DOS MECHAS

Tenemos dos mechas, cada una de las cuales tarda una hora en consumirse completamente. Esto quiere decir que, una vez que se le ha prendido fuego, la mecha se termina exactamente en una hora, y eso es todo lo que sabemos: la mecha no tiene por qué consumirse siempre al mismo ritmo, de forma que puede que media mecha tarde más o menos de media hora en consumirse. Lo único que se sabe es que cada mecha tarda una hora en consumirse completamente. La pregunta es:

¿Cómo podemos medir 45 minutos de tiempo, usando únicamente estas dos mechas?

Por supuesto, se entiende que podemos encenderlas cuando queramos; y desde luego, no puede usarse ningún tipo de reloj, así como ninguna otra forma de medir el tiempo que no sean las mechas. Y la solución debe dar un método exacto de medir los 45 minutos.



JUEGO 3: LA MOSCA SALTARINA

Sobre una de las monedas se ha posado una mosca. Ésta se ha propuesto recorrer las 25 monedas, saltando de una a otra contigua horizontalmente o verticalmente (no en diagonal), y de

INTRODUCCIÓN

Es claro que un buen comienzo en el área de programación es fundamental para que al alumno le guste ésta, de acuerdo a la forma como se manejen los conceptos y a como se apliquen los mismos y teniendo cuenta que es un punto a fortalecer, el objetivo de esta producción académica es que el alumno conozca los conceptos y sepa aplicarlos a la solución de cualquier problema; para una buena solución debe haber un buen análisis un buen planteamiento y una buena aplicación de estructuras, es por eso que el libro busca integrar todo aquello en una forma didáctica y completa, a lo largo de las diferentes unidades se irán desarrollando los conceptos y aplicándolos en ejercicios.

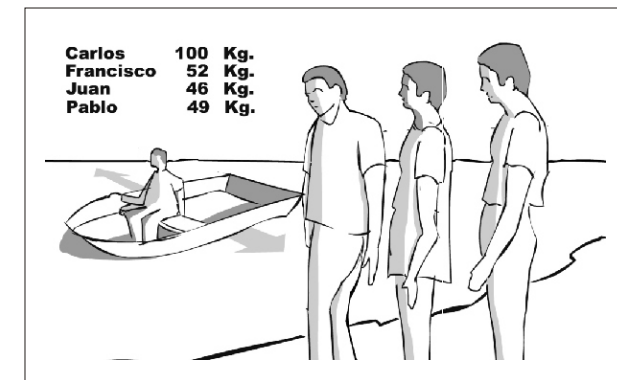
JUEGOS PARA DESARROLLAR LA LÓGICA

JUEGO 1: CRUZAR EL RÍO

Cuatro amigos han de cruzar un lago en una barca de remos. El barquero que les había alquilado la barca les había dicho que ésta sólo podía cargar un máximo de 100 kgs., justo lo que pesaba Carlos. Los otros tres pesaban, sin embargo mucho menos; Francisco pesaba 52 kgs., Juan pesaba 46 kgs.; Pablo pesaba 49 Kgs. Éste, además, no sabía remar. Tras mucho pensar, dieron con una manera de cruzar los cuatro, aunque les supuso varios viajes. ¿Cómo lo hicieron?

Tú deberás conseguirlo en el menor número de viajes posible.

Elabore la solución lógica para el programa de computador que permita dar la solución acertada.



JUEGO 8: LA BOMBILLA

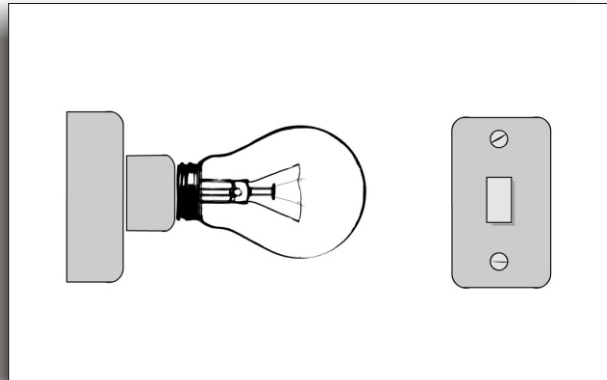
Al final de un largo pasillo hay una habitación dentro de la cual existe una bombilla.

La habitación está cerrada con una puerta de manera que no deja pasar ni un solo rayo de luz.

Inicialmente la bombilla está apagada.

A lo largo del pasillo y a intervalos de 2 metros existentes interruptores, uno sólo de los cuales apaga y enciende la bombilla.

Podemos avanzar por el pasillo y accionar o no los interruptores que queramos, pero con la condición de que no podemos retroceder tras pasar por un interruptor. Tras ello podemos abrir la puerta y entrar en la habitación.



JUEGO 9: CONVENTO DE MONJAS

Había un convento, en Italia, con cierta cantidad de monjas. El convento era cuadrado, tenía dos pisos, y 8 habitaciones por piso. Las monjas estaban distribuidas de tal manera que se respetaban las siguientes características:

- a) En cada lado del convento dormían 11 monjas.
- b) En el segundo piso dormía el doble de monjas que en el primero.

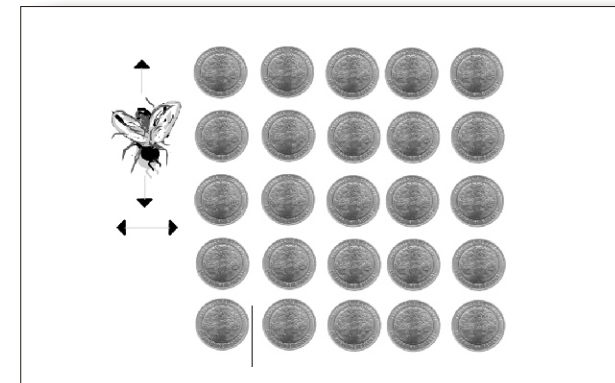
modo que no pase dos veces por la misma moneda.

¿Podrás conseguirlo?

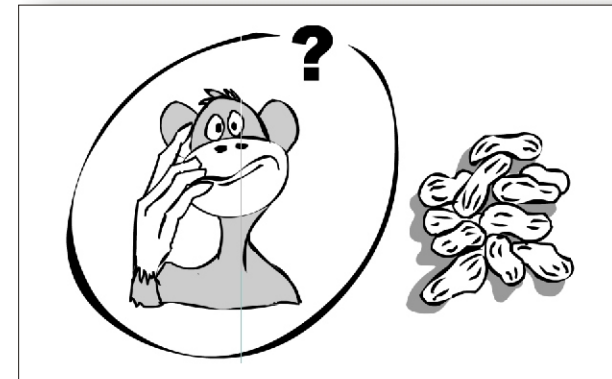
Si es así, describe un recorrido posible. Si no lo es, demuestra por qué. ¿Qué ocurriría si partiera de otra moneda diferente?. ¿Podrá conseguirlo siempre?. ¿No podrá conseguirlo nunca?.

¿Partiendo de unas monedas, sí; pero de otras, no?.

Habrás de determinar las monedas desde las cuales es posible completar el recorrido.



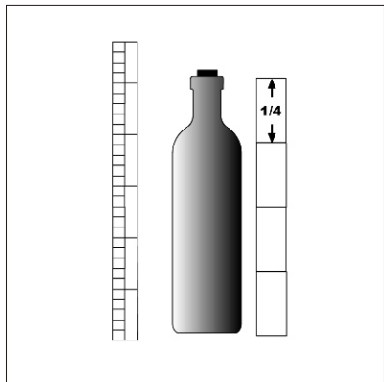
JUEGO 4: LOS CACAHUETES DEL MONO



Un mono tiene una bolsa con bastantes cacahuetses. Cada mañana su dueño le añade 100 cacahuetses exactamente en la bolsa. Luego, durante el día, el mono se come la mitad de los cacahuetses que encuentra en el saco y deja la otra mitad. Una noche, después de varios años comportándose así, el dueño contó el número de cacahuetses que el mono había ahorrado en la bolsa. ¿Cuántos había?

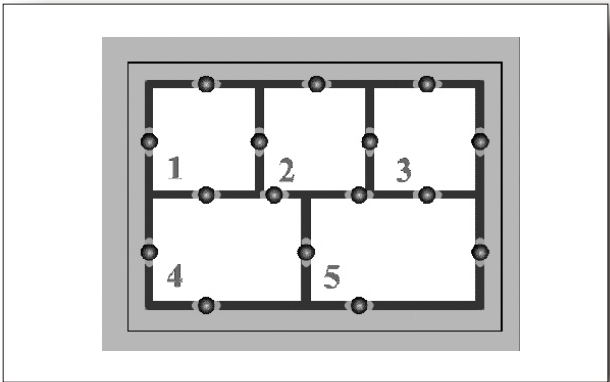
JUEGO 5: LA BOTELLA DE VINO

La parte inferior de las botellas de vino es cilíndrica. La altura de esta zona es 3/4 de la altura total. La parte superior (1/4) tiene forma irregular. Una botella está aproximadamente llena hasta la mitad de su altura. Sin destapar la botella y, ayudándonos únicamente de una regla graduada, ¿cómo podríamos determinar con exactitud el porcentaje del total de la botella ocupado por el líquido?

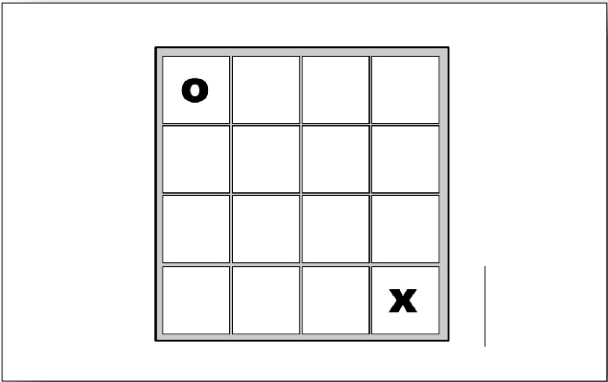


JUEGO 6: PUERTAS

El problema consiste en cruzar todas las puertas de la figura, mediante una línea, sin pasar dos veces por la misma puerta.



JUEGO 7: MASACRE EN PRISIÓN



Un prisionero se vuelve loco en la celda X; rompe un tabique, penetra en la celda vecina y mata al ocupante. Sabiendo que: Hay un preso en cada celda, Mata a todos los presos en el acto, después de cada crimen, el asesino abandona a la víctima en busca de otra (no la transporta). Nunca vuelve a una celda en la que se halla un cadáver, y... No rompe ningún muro que da al exterior ¿Cuál es su macabro itinerario que le lleva desde la celda X a la celda O?

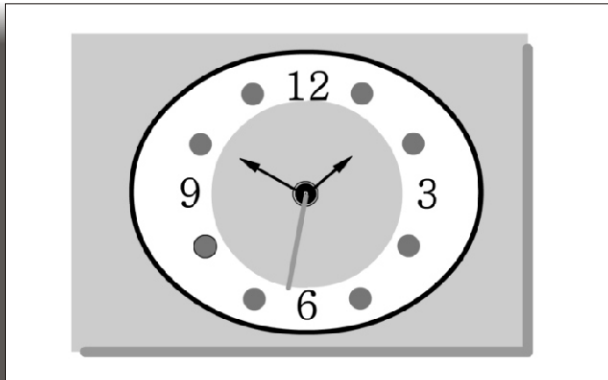
Puedes intentarlo con 10 o 15 bolas.

JUEGO 14:REORDENACIÓN

Se trata de reordenar los números del 1 al 9, de modo que las diferencias en valor absoluto entre las parejas de números que ocupan la misma posición en la lista sean todas diferentes.

JUEGO 15: EL RELOJ DE PARED

Había una vez un hombre que no tenía reloj, ni de pulsera, ni de bolsillo, pero tenía un reloj de pared muy exacto que sólo se paraba cuando se olvidaba de darle cuerda. Cuando esto ocurría, iba a casa de un amigo suyo, pasaba la tarde con él y al volver a casa ponía el reloj en hora. ¿Cómo es posible esto sin saber de antemano el tiempo que tardaba en el camino?



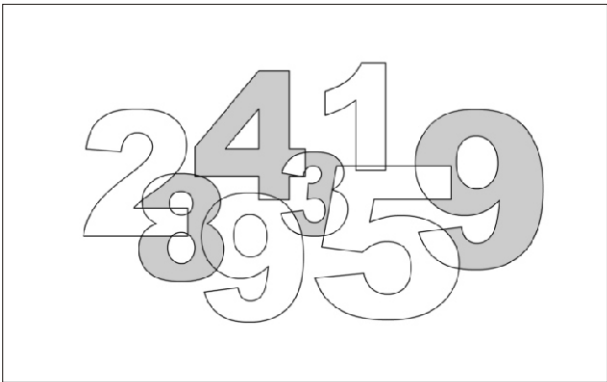
JUEGO 16: ¿64 Ó 65 CUADRADOS?

Observa con detenimiento las dos figuras. ¿Ves algo increíble?. ¿Será posible?. ¿Cómo lo explicas?.

Resulta que un buen día se escaparon 9 monjas, lo cual, de ser descubierto, hubiera causado un gran revuelo. Las restantes monjas entonces idearon una redistribución de las mismas, que mantenía las condiciones anteriores. De esa manera, cuando la superiora pasaba por las noches a revisar los cuartos, se encontraba con que cada lado del convento seguía albergando 11 monjas, y en el piso superior seguía habiendo el doble de monjas que en el primero. Sin embargo faltaban 9 monjas.

¿Cómo estaban distribuidas las monjas, antes y después de la desaparición de las 9?

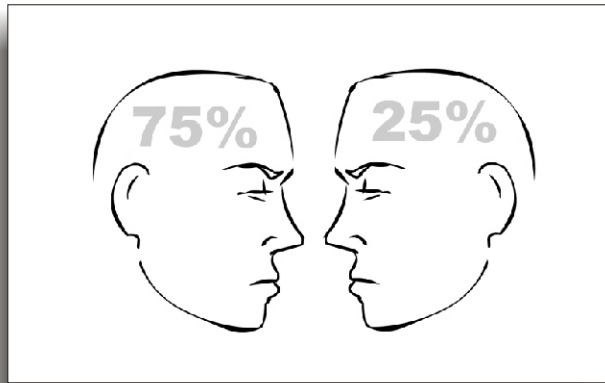
JUEGO 10 : ENTRE NÚMEROS



Partimos de varias parejas de números consecutivos. Es decir, dos "unos", dos "doces", dos "treces", dos "cuatros"....Se trata de disponerlos en fila, de modo que entre los dos "unos" haya un número; entre los dos "doces", dos números; entre los dos treces, tres números... ¿Sencillo, verdad?. No lo creas. Pronto crecerán las dificultades.

El problema consiste en determinar si es posible conseguirlo, cualquiera que sea el número de parejas del que se parte, y, en su caso, idear la manera de hacerlo.

JUEGO 11: EXPEDIENTE X



No es éste, propiamente, un problema de ingenio, sino de probabilidad. No vienen mal ciertos conocimientos de probabilidad para resolverlo, pero, como en muchos problemas de este tipo, la lógica y el ingenio pueden ser suficientes. Si está aquí, es por su "ingenioso" y difícilmente descifrable enunciado:

- 1- Scully piensa que Mulder está errado el 75% de las veces.
- 2- Mulder piensa que los extraterrestres están involucrados el 60% de las veces.
- 3- En los casos en donde Mulder piensa que los extraterrestres están involucrados, Scully piensa que Mulder está errado el 95% de las veces.

Se debe determinar o calcular la probabilidad de que Mulder

piense que los extraterrestres están involucrados en los casos en donde Scully piensa que Mulder está errado.

JUEGO 12: LA LINTERNA

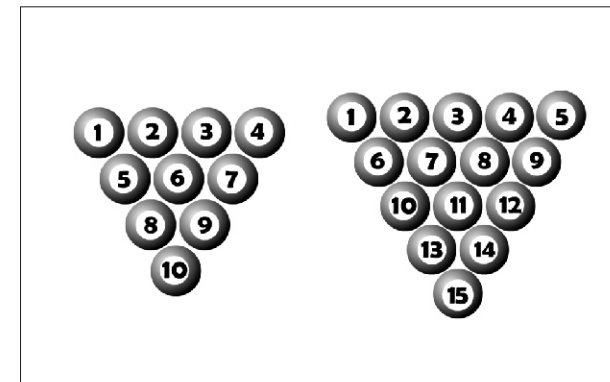
Una noche oscura hay cuatro hombres de este lado del río. Los cuatro deben cruzar al otro lado a través de un puente que como máximo puede sostener a dos hombres al mismo tiempo. Tienen una sola linterna. Esto obliga a que si dos hombres cruzan al mismo tiempo, deban hacerlo juntos, a la velocidad del más lento. También obliga a que alguno de ellos vuelva para alcanzarles la linterna a los que se quedaron.

Cada uno tarda una velocidad diferente en cruzar: Genio, veloz como el pensamiento, tarda 1 minuto. Pablo, rápido como su automóvil, tarda 2 minutos. Gustavo, entumecido por los fríos del Polo Norte, tarda 5 minutos. Ángel, que insiste en llevar doce cajas de cerveza, tarda 10 minutos.

En qué orden deben cruzar los cuatro hombres, para tardar en total exactamente 17 minutos?

JUEGO 13: BOLAS DE BILLAR

Se trata de reenumerar las bolas de modo que cada bola sea la diferencia de las dos que tiene encima.

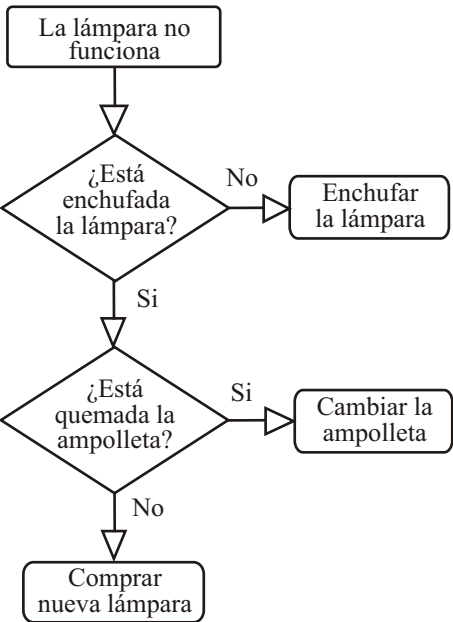


- 1. Construcción de la solución lógica.
- 2. Codificación.
- 3. Compilación.
- 4. Ejecución.

SOLUCIÓN LÓGICA

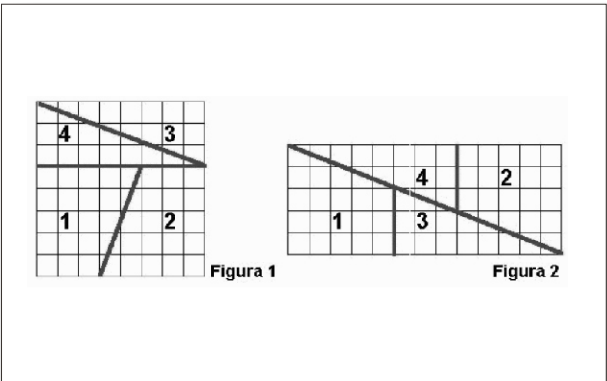
→ Consiste en definir las instrucciones necesarias para resolver un problema (lo primero es tener en cuenta que lo mas importante para dar una solución lógica es saber que es lo que nos piden en un problema, para esto debemos leer hasta entenderlo (análisis).

→ Escribir las instrucciones en un orden lógico.



CODIFICACIÓN

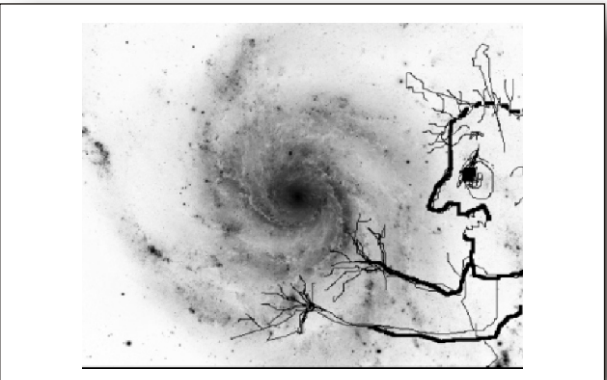
→ Es traducir la solución a un lenguaje de programación (el que el usuario desee).



JUEGO 17: VIAJE GALÁCTICO

Expedición: Planeta L. Biólogo: Profesor K.

Informe: 'El tercer día vimos seres extraños. Aunque tienen veinte dedos en total, como nosotros, tienen una extremidad menos y un dedo más en cada extremidad, lo que les da, por cierto, un aspecto espantoso'.
¿Cuántas extremidades poseen dichos seres?



EL MUNDO DE LA PROGRAMACIÓN

Es importante tener en cuenta que programar no es aprender de memoria es desarrollar, analizar y aplicar una serie de conceptos en la solución de un problema propuesto, por ello es importante no olvidar una serie de conceptos los cuales se verán a continuación.

1. PROGRAMA

- Es un conjunto de instrucciones coherentes.
- Las instrucciones tienen que estar lógicamente distribuidas.
- Las instrucciones deben de estar codificadas en un lenguaje de programación Ej: C, VBASIC, JAVA, DEV C++ , entre otros.

2. PASOS A TENER EN CUENTA PARA LA CONSTRUCCIÓN DE UN PROGRAMA

| Construcción de Solución lógica | Codificación en un lenguaje de programación | Compilación | Ejecución |
|---|---|---|---|
|  |  |  |  |

5. PAPEL DE LA MEMORIA RAM EN LA PROGRAMACIÓN

RAM

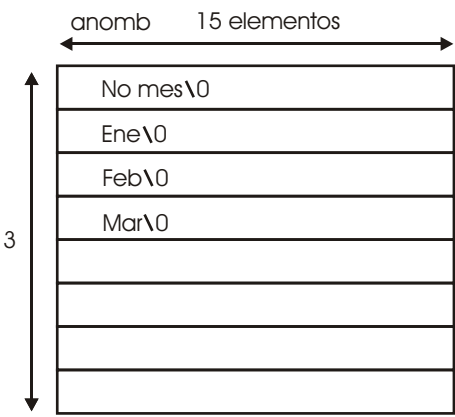


UN MAPA DE LA MEMORIA RAM

| | |
|--|---------|
| Sistema Operativo | |
| Lenguaje de programación Ejemplo C | |
| El código de programa escrito por usted | |
| | Área De |
| | Datos |

Random Acces Memory

- La Memoria RAM es el área de operaciones del procesador, es la mesa sobre la cual se trabaja .
- La Memoria RAM es una memoria de almacenamiento temporal en la cual se almacenan los datos de las variables , los datos que se almacenan en la ejecución de un programa quedan almacenada en campos de la Memoria RAM, al apagarse el computador estos datos se borran.



COMPILACIÓN

- Se cumple una vez digitado el programa.
- Consiste en corregir los errores de sintaxis de un programa.
- Un error de sintaxis es la violación de una norma de escritura.

En lenguajes de programación tales como c++ todas las líneas de programación terminan con punto y coma (;) si al poner a compilar en alguna de las líneas falta un punto y coma el programa no se ejecutará hasta que a nivel de sintaxis el programa este bien.(corrección de errores).

EJECUCIÓN

- En la ejecución el procesador toma las instrucciones las ejecuta una por una con ayuda de otros componentes, como sucede con las operaciones de entrada (lectura) y salida (escritura).
- El proceso no piensa, no razona, no infiere, no deduce, por lo tanto el procesador no se equivoca.
- Si los resultados de un proceso son malos, se equivoco el programador.

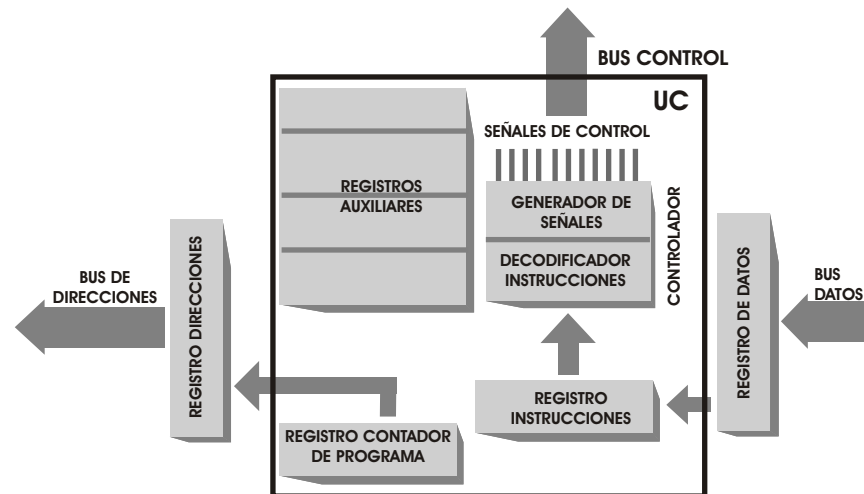
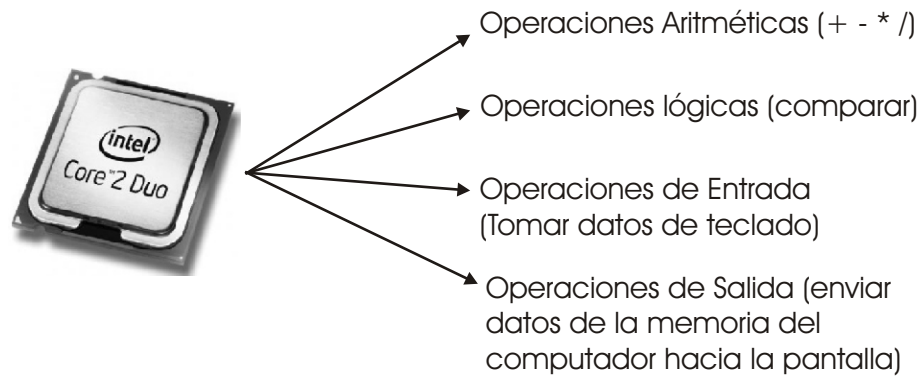
“Una máquina no se puede equivocar ese privilegio es de los humanos. Si las instrucciones del programador son incorrectas..... los resultados serán incorrectos.

3. LENGUAJES DE PROGRAMACIÓN

- Nos sirven para escribir un programa de computador
- Los lenguajes de programación es un programa que sirve para escribir programas.
- Existen varios lenguajes de programación, por ejemplo:

C
DEV C++
VBASIC
DELPHI
COBOL
JAVA

4. PAPEL DEL PROCESADOR EN LA PROGRAMACIÓN



En un programa se pueden presentar los siguientes tipos de operaciones (instrucciones).

Operaciones aritméticas

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

Operaciones Lógicas

Son comparaciones
Los operadores lógicos para hacer comparaciones son
Igual (=)
Menor (<)
Mayor (>)
Diferente (<>)

Operaciones de Entrada

Consisten en capturar (leer) datos desde un periférico, por ejemplo, el teclado.

Los datos capturados van a Memoria RAM.

Operaciones de salida

Consisten en enviar datos desde un área de la Memoria RAM hacia un periférico, por ejemplo, la pantalla.

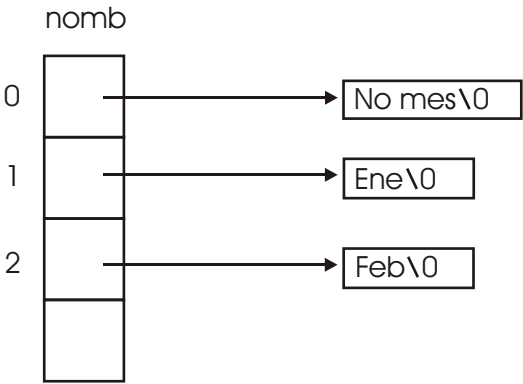
| | |
|---------|--|
| | <p>usualmente trasladando la <i>coma decimal</i> -mediante un exponente- hacia la posición de la primera cifra significativa del valor.</p> <p>De esta forma, con un número dado de dígitos representativos se obtiene mayor precisión que con el punto fijo, debido a que el valor de estos dígitos es siempre significativo sea el que sea el orden de magnitud del número a representar. Debido a esta adaptación, permite representar un rango mucho mayor de números (determinado por los valores límite que puede tomar el exponente).</p> |
| CHAR | Un valor de tipo carácter es cualquier carácter que se encuentre dentro del conjunto ASCII ampliado, el cual está formado por los 128 caracteres del ASCII más los 128 caracteres especiales que presenta. |
| STRING | una cadena de caracteres, palabra o frase (<i>String</i> en inglés) es una secuencia ordenada de longitud arbitraria (aunque finita) de elementos que pertenecen a un cierto alfabeto. En general, una cadena de caracteres es una sucesión de caracteres (letras, números o determinados signos). |
| BOOLEAN | El tipo de dato lógico o <i>booleano</i> es en computación aquel que puede representar valores de lógica binaria, esto es, valores que representen <i>falso</i> o <i>verdadero</i> . |

7.1 COMO DEFINIR (CREAR) UN CAMPO

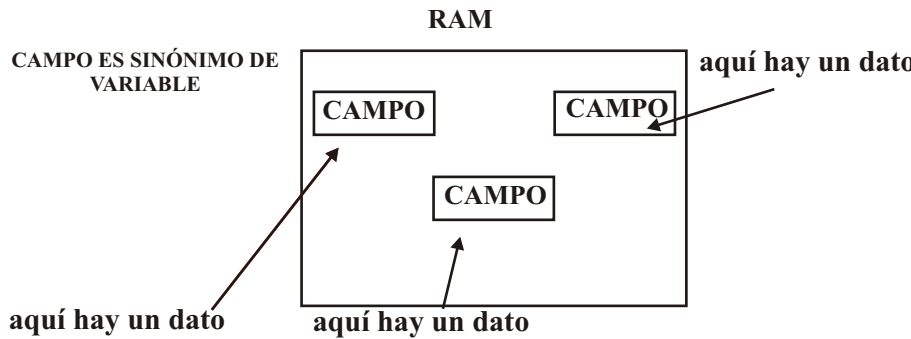
En la mayoría de los lenguajes de programación hay una sección en la cual se definen los campos.

Para definir un campo se debe tener en cuenta:

- **Nombre del campo:** El que el programador quiera.
- **Tipo de campo:** Se indica si es Integer, String, char.....



5.1 UBICACIÓN DE LOS DATOS EN MEMORIA RAM



→ Cada dato se ubica en el área de Memoria RAM que técnicamente los programadores denominan campo o VARIABLE.

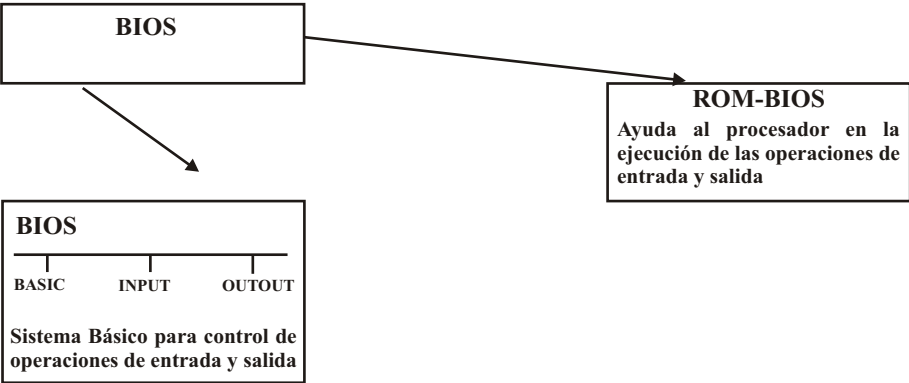
- El programador determina cuantas VARIABLES (campos de memoria) necesita. Se crea uno para cada dato .
- Se le asigna uno a cada variable.
- Establece sus características (el tipo, longitud).

Las operaciones que hace el procesador son entre datos que se encuentran en campos de memoria.

6. PAPEL DE LA MEMORIA ROM DURANTE LA EJECUCIÓN DE UN PROGRAMA

La memoria ROM contiene la Bios (Basic Input Output System) que consiste en una serie de rutinas que se encargan de ejecutar las operaciones de entrada y salida.

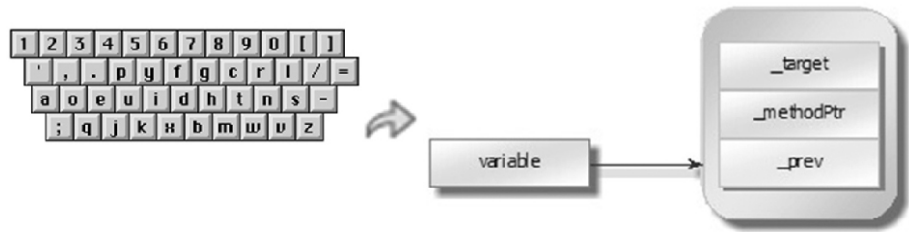
- Cada vez que durante la ejecución del programa aparezca una operación de entrada (lectura) o de salida(escritura) el procesador se detiene y le pide ayuda a la ROM-BIOS para que le ejecute dicha operación.
- Para cada tipo de operación de entrada o de salida hay una rutina específica en la BIOS.
- A las rutinas que compone la ROM-BIOS se les denomina interrupciones del sistema (el procesador interrumpe su actividad mientras la BIOS le resuelve el problema).



7. CAMPO (VARIABLE)

- Es el área de memoria en la cual se almacena un dato
- Es el sinónimo de variable.
- Es el área de memoria en la cual se ubica un dato.

→ El programador determina los campos que necesita para desarrollar la solución de un problema.



Los campos tienen una característica fundamental.

→ EL TIPO

Indica la clase de información que se va a ubicar en le campo.

Tipo de un campo

- Todo campo pertenece a un determinado tipo.
- El tipo indica la clase de información que almacenará el campo.

TIPOS DE CAMPOS

| | |
|--------------|---|
| INTEGER | Un tipo de dato entero en computación es un tipo de dato que puede representar un subconjunto finito de los números enteros. El número mayor que puede representar depende del tamaño del espacio usado por el dato y la posibilidad (o no) de representar números negativos. Los tipos de dato entero disponibles y su tamaño dependen del lenguaje de programación usado así como la arquitectura en cuestión. |
| REAL (FLOAT) | Coma flotante o punto flotante es un método de representación de números reales que se puede adaptar al orden de magnitud del valor a representar, |

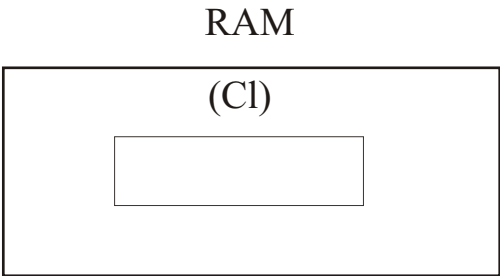
10.1. COSAS A TENER EN CUENTA

Observe la siguiente combinación entre un ESCRIBA y un LEA
Escriba ("digite su clave")
Lea (C1)

→ El Escriba hace que en la pantalla aparezca el mensaje "digite su clave"



→ C1 debe ser el campo de memoria en el cual se recibirá el dato (es este caso la clave), que se va a ingresar desde el teclado.



→ En algunos lenguajes (como cobol) se especifica el tamaño del campo.

Ejemplo:

| VARIABLE | |
|-----------|------------|
| M,B,G,T,Y | Integer |
| L,A | String(10) |
| V | Char |

- M,B,G,T,Y han sido definidos como campos que van a recibir valores enteros únicamente,
- L y A pueden recibir únicamente cadenas de caracteres de una longitud máxima de 10.
- V puede recibir únicamente un carácter.

En el cuerpo del programa (es decir, en las instrucciones) únicamente se puede utilizar los campos indicados en la sección de definición de campos, en caso que no sea especificado en la sección de definición de variables se genera un error conocido como VARIABLE INDEFINIDA .

8. CONSTANTES

Es un dato cuyo valor no puede cambiar durante la ejecución del programa. Recibe un valor en el momento de la compilación y este permanece inalterado durante todo el programa.

```
const
  Min = 0;
  Max = 100;
  Sep = 10;
```

9. OPERACIONES DE ENTRADA (LECTURA)

Consiste en enviar un dato desde un periférico hacia un campo de la Memoria RAM, por ejemplo, un dato se puede enviar desde el teclado hacia un campo de memoria.
En los programas de computador la operación de entrada se especifica el comando LEA.

LEA (campo)

- LEA
→ Indica que se debe recibir un dato desde un periférico de entrada (teclado).
- CAMPO
→ Es el nombre del campo en el cual recibirá el dato tecleado por el usuario del programa.

Cuando un programa se está ejecutando y el procesador llega a una instrucción LEA, se dan los siguientes eventos:

- A. El Procesador deja de trabajar.
- B. Una rutina ubicada en la ROM BIOS se encarga de ejecutar la operación de entrada (lea).
 - B1. Se recibe el dato desde teclado.
 - B2. El dato se ubica en el campo indicado.

Si el usuario del programa no tecléa un dato, el programa queda detenido hasta que el usuario proceda a digitar algo.

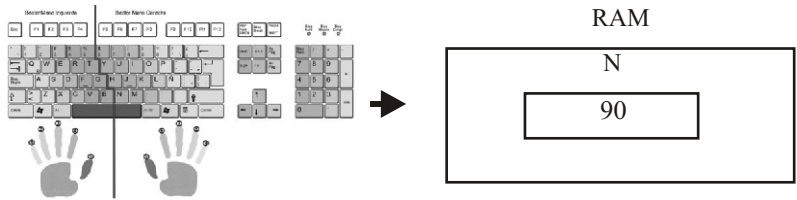
C. Cuando se termina de ejecutar la instrucción LEA (es decir, se tecléa el dato y el campo respectivo lo recibió), el procesador retorna el control del programa y pasa a ejecutar la siguiente instrucción.
Si la siguiente instrucción es una operación de entrada o de

salida, obviamente la ROM-BIOS volverá a tomar las riendas del programa.

Si por ejemplo se tiene la instrucción.

LEA (N)

Y al ejecutarse el usuario digita el valor 90, en N se almacenará dicho valor.

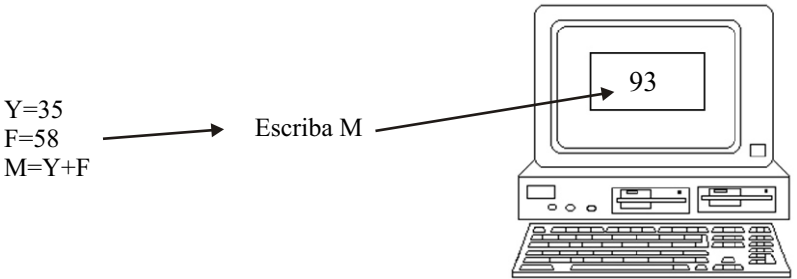


10. OPERACIONES DE SALIDA (ESCRIBA)

Una operación de salida consiste en reflejar un dato en un periférico de salida (pantalla, impresora, unida de disco).

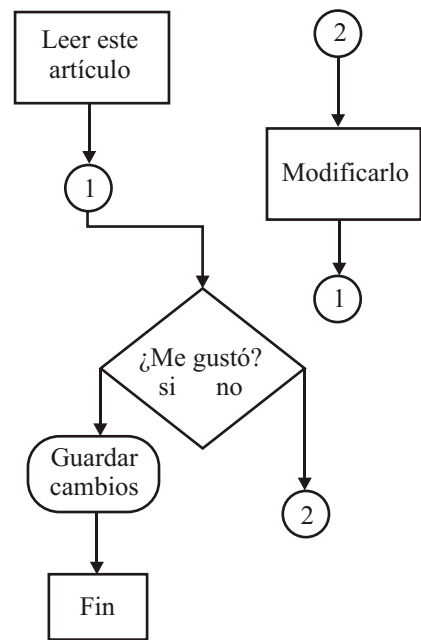
- En este caso, reflejar sería sinónimo de escribir.
- Para este modulo, el periférico de salida será siempre la pantalla.
- Las operaciones de salida también corren a cargo de ROM-BIOS.

ESCRIBA (Campo)



Otra definición del diagrama de flujo es la siguiente:

"Es un esquema para representar gráficamente un algoritmo. Se basan en la utilización de diversos símbolos para representar operaciones específicas. Se les llama diagramas de flujo porque los símbolos utilizados se conectan por medio de flechas para indicar la secuencia de operación. Para hacer comprensibles los diagramas a todas las personas, los símbolos se someten a una normalización; es decir, se hicieron símbolos casi universales, ya que, en un principio cada usuario podría tener sus propios símbolos para representar sus procesos en forma de Diagrama de Flujo. Esto trajo como consecuencia que sólo aquel que conocía sus símbolos, los podía interpretar. La simbología utilizada para la elaboración de diagramas de flujo es variable y debe ajustarse a un patrón definido previamente."



1.1. QUE ES UN ALGORITMO Y PARA QUE NOS SIRVE

Un algoritmo es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea y/o resolver un problema. De un modo más formal, un algoritmo es una secuencia finita de operaciones realizables, no ambiguas, cuya ejecución da una solución de un problema en un tiempo finito.

(Figura: algoritmo en diagrama de flujo en la página 38).

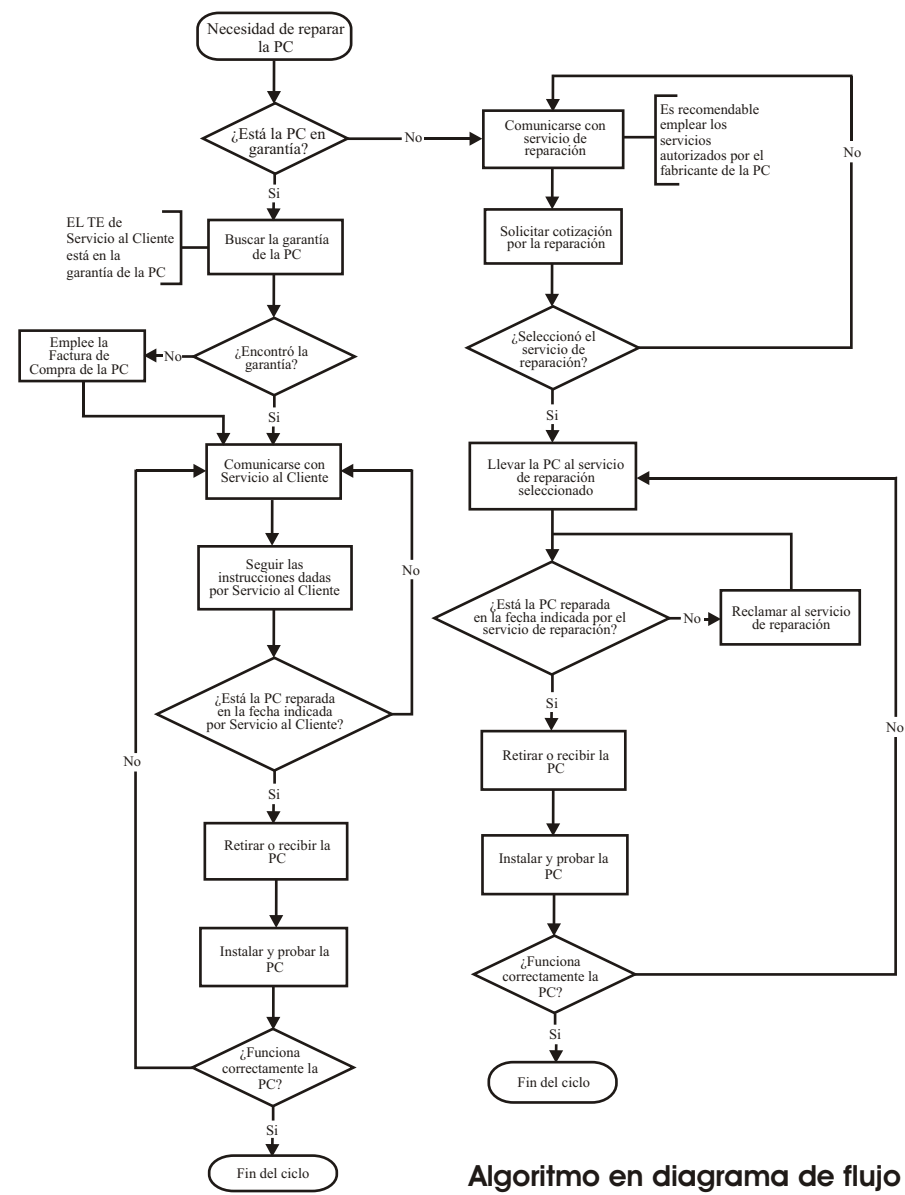
Los diagramas de flujo se usan habitualmente para representar algoritmos.

El término **algoritmo** no está exclusivamente relacionado con la matemática, ciencias de la computación o informática. En realidad, en la vida cotidiana empleamos algoritmos en multitud de ocasiones para resolver diversos problemas. Algunos ejemplos son el uso de una lavadora (se siguen las instrucciones), pero no la preparación de una comida (porque no están perfectamente definidos los pasos).

- Para solucionar computacionalmente un problema debemos:
- Seleccionar un modelo matemático computacional adecuado para el problema (representación del modelo)
 - Concebir con respecto a dicho modelo un algoritmo que de solución al algoritmo (diseño del algoritmo)
 - Programar el algoritmo en algún lenguaje de programación y ejecutar el programa en una computadora (programación del algoritmo)

Estructura Básica:

- 1. inicio
- 2. datos de entrada (operaciones básicas)
- 3. procesamiento de los datos
- 4. datos de salida
- 5. Fin



Algoritmo en diagrama de flujo

Ejemplo:

Se presenta el algoritmo para encontrar el máximo de un conjunto de enteros positivos. Se basa en recorrer una vez cada uno de los elementos, comparándolo con un valor concreto (el máximo entero encontrado hasta ahora). En el caso de que el elemento actual sea mayor que el máximo, se le asigna su valor al máximo.

El algoritmo escrito de una manera más formal, esto es, en pseudocódigo tendría el siguiente aspecto:

ALGORITMO Máximo

ENTRADAS: Un conjunto no vacío de enteros C.

SALIDAS: El *mayor* número en el conjunto C.

máximo ← -∞

PARA CADA elemento **EN** el conjunto C, **HACER**
SI *valor_del_elemento* > *máximo*, **HACER**
máximo ← *valor_del_elemento*
DEVUELVE *máximo*



11.1 COMO DAR SOLUCIÓN A LOS ALGORITMOS

Les podemos dar solución por medio de diagramas de flujo o por medio de pseudocódigos.

11.1.1 . DIAGRAMAS DE FLUJO

Los **diagramas de flujo** representan la forma más tradicional para especificar los detalles algorítmicos de un proceso. Se utilizan principalmente en programación, economía y procesos industriales; estos diagramas utilizan una serie de símbolos con significados especiales. Son la representación gráfica de los pasos de un proceso, que se realiza para entender mejor al mismo. Son modelos tecnológicos utilizados para comprender los rudimentos de la programación lineal.

| Símbolo | Descripción |
|---------|--|
| | Una buena práctica de diagramación utilizar el mismo lado para los positivos siempre que esto sea posible. |

| | |
|---|--|
|  | Desplegado de información. Este símbolo se utiliza para mostrar un resultado, el cual puede representar la solución al problema que se pretende resolver y que fue conseguida a través del resto del diagrama. Dentro de su interior se anotará la variable con el resultado final o el mensaje que represente el resultado del algoritmo. |
|  | Generalmente veremos este símbolo muy cerca del final del proceso y precedido por el símbolo de terminación. Este símbolo siempre deberá tener al menos una conexión de entrada y una de salida. |

Símbolos auxiliares

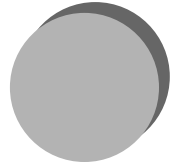
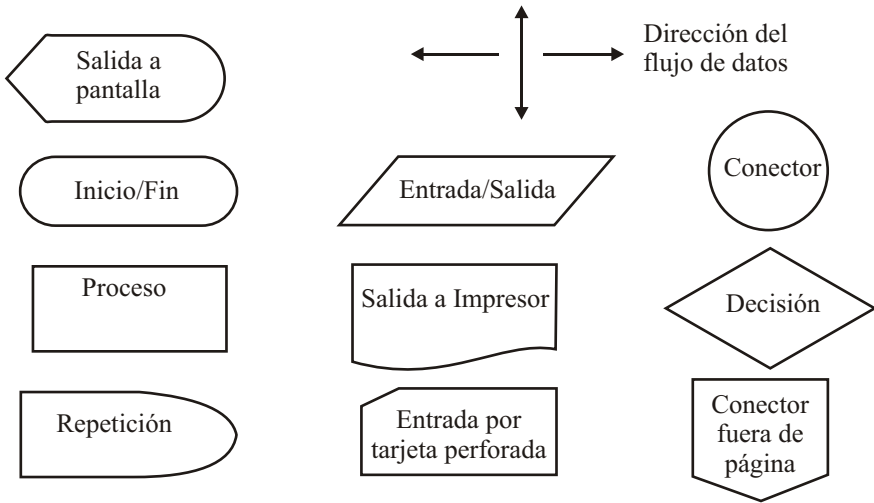
| Símbolo | Descripción |
|---|--|
|  | Conector. Este símbolo se utiliza para indicar un salto dentro del diagrama. Se utiliza con el propósito de facilitar la disposición plana de un diagrama y evitar el cruce excesivo de líneas a través del mismo. Este conector va asociado a un conector “gemelo” y junto con él, representa una puerta de entrada y de salida para el flujo del diagrama, es decir que cuando |

Diagrama de flujo en el cual se emplean los símbolos más comunes.

PRINCIPALES SÍMBOLOS



Estandarizados según ISO 5807

No es indispensable usar un tipo especial de símbolos para crear un diagrama de flujo, pero existen algunos ampliamente utilizados por lo que es adecuado conocerlos y utilizarlos, ampliando así las posibilidades de crear un diagrama más claro y comprensible para crear un proceso lógico y con opciones múltiples adecuadas.

- **Flecha.** Indica el sentido y trayectoria del proceso de información o tarea.
- **Rectángulo.** Se usa para representar un evento o proceso determinado. Éste es controlado dentro del diagrama de flujo en que se encuentra. Es el símbolo más comúnmente utilizado.


• **Rectángulo redondeado.** Se usa para representar un evento que ocurre de forma automática y del cual generalmente se sigue una secuencia determinada.



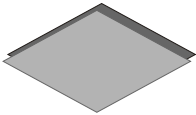
• **Rombo.** Se utiliza para representar una condición. Normalmente el flujo de información entra por arriba y sale por un lado si la condición se cumple o sale por el lado opuesto si la condición no se cumple. Lo anterior hace que a partir de éste el proceso tenga dos caminos posibles.

• **Círculo.** Representa un punto de conexión entre procesos. Se utiliza cuando es necesario dividir un diagrama de flujo en varias partes, por ejemplo por razones de espacio o simplicidad. Una referencia debe darse dentro para distinguirlo de otros. La mayoría de las veces se utilizan números en los mismos.

Existen además una variedad de formas especiales para denotar las entradas, las salidas, los almacenamientos, etcétera.

Descripción de los principales símbolos utilizados en diagramación.

| Símbolo | Descripción |
|---|--|
|  | <p>Inicio / Terminación. Este símbolo se utiliza para señalar el comienzo así como el final de un diagrama. Tradicionalmente se colocan las palabras “INICIO” ó “FIN” dentro de la figura para hacerlo más explícito.</p> <p>Es el único símbolo que solamente tiene una conexión (flecha) ya sea de salida, en el de inicio, o de entrada, para el de fin.</p> |

| Símbolo | Descripción |
|---|---|
|  | <p>Entrada de dato. En este símbolo se indican los valores iniciales que deberá recibir el proceso. Esto se hace asignándoles letras o nombres de variables para cada uno de los valores y anotando estas letras en el interior de la figura.</p> <p>Existen otros símbolos que también representan una entrada de datos pero no consideramos que su utilización, o combinación, aporte mayor utilidad al objetivo intrínseco de ejemplificar una entrada de datos.</p> <p>Este símbolo siempre deberá tener al menos una conexión entrante (generalmente del inicio) y una de salida.</p> |
|  | <p>Proceso de dato. Este símbolo lo utilizaremos para señalar operaciones matemáticas, aritméticas o procesos específicos que se realicen con nuestros datos.</p> <p>La manera de anotar dichos procesos, puede ser mediante una descripción breve de la operación o mediante una asignación de dicha operación hacia una variable como por ejemplo: $R \leftarrow A + B$.</p> <p>Este símbolo siempre deberá tener al menos una conexión de entrada y una de salida.</p> |
|  | <p>Decisión. Este símbolo nos representa una disyuntiva lógica o decisión. En su interior se anota una instrucción o pregunta que pueda ser evaluada como cierta o falsa y que determine el flujo del programa.</p> <p>Este símbolo es el único que puede contener dos salidas y en cada una de las salidas se suele poner un rótulo de “sí/no” o “cierto/falso” indicando con esto cual de ellas se tomará según el resultado de la evaluación de la función.</p> |

Para seguir en forma ordenada, el cambio de valores en las variables, se utiliza una tabla llamada traza.

| | |
|-------|-----------|
| a | 2 |
| ----- | |
| b | |
| a | 2 |
| ----- | |
| b | 8 |
| a | 2 12 |
| ----- | |
| b | 8 |
| a | 2 12 |
| ----- | |
| b | 8 -94 |
| a | 2 12 106 |
| ----- | |
| b | 8 -94 100 |

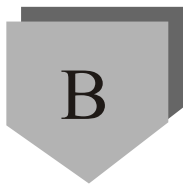
valores finales : a = 106 , b = 100

Por lo tanto si se pide una traza, se debe mostrar la última tabla. Los pasos anteriores son sólo pasos explicativos.

Ejercicio:

Hacer la traza del siguiente algoritmo

```
INICIO
  t <-- 1
  p <-- 4
  t <-- 3 * t
```

| Símbolo | Descripción |
|---|--|
| | una flecha termina en un conector marcado con la letra “A”, se continuará el diagrama a partir de otro conector marcado con la misma letra tal como si se tratara de una línea continua in interrumpida. |
|  | Conector de páginas. Este conector es idéntico en funcionamiento que el anterior, pero su forma pentagonal lo distingue y nos indica que debemos buscar el “gemelo” en una página distinta de la actual. Este conector lleva asociado una especie de salto entre páginas. |

11.1.2. Reglas para trabajar con diagramas de flujo

De acuerdo al estándar ISO, los símbolos e incluso las flechas deben tener ciertas características para permanecer dentro de sus lineamientos y ser considerados sintácticamente correctos. En el caso del círculo de conexión, se debe procurar usarlo sólo cuando se conecta con un proceso contenido dentro de la misma hoja.

Existen también conectores de página, que asemejan a una casita y se utilizan para unir actividades que se encuentran en otra hoja.

En los diagramas de flujo se presuponen los siguientes aspectos:

- ➔ Existe siempre un camino que permite llegar a una solución.
- ➔ Existe un único inicio del proceso.
- ➔ Existe un único punto de fin para el proceso de flujo, salvo del rombo que indica una comparación con dos caminos posibles y además una gran ayuda.

A su vez, es importante que al construir diagramas de flujo, se

observen las siguientes recomendaciones:

- Evitar sumideros infinitos, burbujas que tienen entradas pero no salidas.
- Evitar las burbujas de generación espontánea, que tienen salidas sin tener entradas, porque son sumamente sospechosas y generalmente incorrectas.
- Tener cuidado con los flujos y procesos no etiquetados. Esto suele ser un indicio de falta de esmero, pero puede esconder un error aún más grave: a veces el analista no etiqueta un flujo o un proceso porque simplemente no se le ocurre algún nombre razonable.

Sentencias simples

Asignación de variables

Se le asigna un valor a una variable, a través del símbolo de flecha (\leftarrow) que dice que el valor a la derecha tiene que ser asignado a la variable que está a la izquierda.

Ejemplo:

```
a ← 10
```

Se asigna un valor 10 a la variable a. Al escribir esta sentencia por primera vez en un algoritmo, da por entendido su declaración o creación.

Comúnmente al lado derecho de la flecha, se pueden realizar operaciones aritméticas (+, -, /, *), para que su resultado se le asigne a la variable. Además en estas operaciones se puede incluir la misma variable u otras.

Ejemplo:

```
INICIO
  x ← 5
  y ← 10
  z ← x + y
  z ← y * x + 7
  z ← x/8 + y/9 + (7 + x/8)
  z ← z + 1
  z ← z * z + y * y
  x ← -x + -y + -3
FIN
```

Ejemplo:

Determinar cuánto vale (a,b,c) después de las siguientes sentencias de asignación

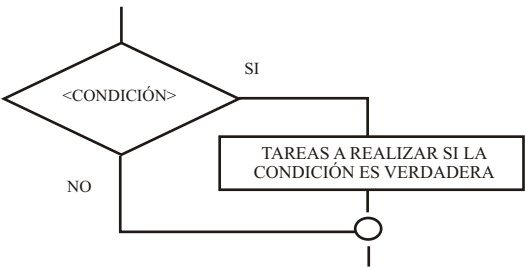
```
INICIO
  a ← 2
  b ← a * 4      /* b ← 2 * 4 = 8 */
  a ← b + 2 * a  /* a ← 8 + 2 * 2 = 12 */
  b ← 2 - a * b  /* b ← 2 - 12 * 8 = 2 - 96 = -94 */
  a ← a - b      /* a ← 12 - -94 = 12 + 94 = 106 */
  b ← -b + 6     /* b ← -(-94) + 6 = 94 + 6 = 100 */
FIN
```


“Toma de Decisión”. Estas tomas de decisión tienen la siguiente forma:

Pseudocódigo:

Si <condición> entonces
Instrucción (es)
Fin - Si

Diagrama de flujo:



Ejercicios:

Supongamos que realizamos un programa para calcular el salario semanal de un empleado que trabaja por horas. La empresa paga una tasa de 1.5 la tasa normal por todas las horas trabajadas superiores a 40.

Leer tasa

Leer HorasTrabajadas

Si HorasTrabajadas > 40 (condición)

entonces (condición se cumple)

paga ← asa * 40 + 1.5 * tasa * (Horas Trabajadas 40)

sino (condición no se cumple)

paga ← tasa * horas

FinSi

Ejercicio:

Algoritmo de la calculadora

INICIO

imprimir("1.- suma");

Imprimir("2.- resta");

p <-- p - 12

t <-- -t + 6

p <-- -p * -2

FIN

Valores finales : t = 3 y p = -16

<BR

Lectura de variables

En muchos algoritmos, el valor de alguna variable debe ser introducido dinámicamente por el usuario a través del teclado o algún otro medio. A esa acción le llamaremos "leer", que consiste en la asignación del valor de entrada, a la variable correspondiente.

Ejemplo:

INICIO

a <-- 12

leer b

a <-- a + b

FIN

Si se tiene que leer más de una variable, se pueden escribir usando el separado coma.

Ej: Leer a, b, c /* lee en ese orden las variables a, b y c */

Declaración de una constante

Si se necesita incluir en el algoritmo alguna constante, se debe seguir la misma metodología de la asignación de variables,

pero con la limitación que el identificador debe estar escrito con letras mayúsculas y que la declaración se haga a continuación del inicio del algoritmo.

Comentarios

Todo comentario de alguna sentencia, debe ir entre los símbolos `/* */`.

Los comentarios son útiles para explicar mejor alguna sentencia que puede estar no clara, para alguien que se enfrenta a las sentencias por primera vez.

Ejemplo:

INICIO

```
PI <-- 3.14 /* constante */
```

```
Leer radio
```

```
área <-- 4 * PI * radio * radio /* área de una esfera */
```

FIN

Método imprimir

Se usará un método llamado "imprimir", para identificar una salida estándar (ej : pantalla o consola).

El método consiste de la palabra "imprimir", seguido de un conjunto de argumentos, delimitado entre paréntesis.

`Imprimir(..argumentos..)`

Los argumentos pueden ser de dos tipos: constantes alfanuméricas o variables. Si se quiere encadenar una constante con una variable, se debe usar el separador '+'.

Ejemplo :

INICIO

```
imprimir("Ingrese radio : ") /* imprime el mensaje
```

```
constante "Ingrese x : " */
```

```
leer radio
```

```
imprimir("El valor del radio ingresado es : " + radio) /* se
```

```
encadena lo constante con lo variable */
```

```
imprimir(r * 2) /* imprime el resultado de la operación
```

```
aritmética */
```

FIN

En el caso que la operación tenga el operador suma, se deben adicionar paréntesis para su correcta interpretación. Debe ser interpretado como un operador aritmético y no de concatenación.

```
Imprimir("la suma de a y b es : " + (a + b) )
```

`Imprimir("a + b + c +++")` /* como el símbolo suma esta dentro de la cadena constante, limitada por comillas "", es interpretado como carácter */

11.2 OPERACIONES LÓGICAS

Las estructuras condicionales comparan una variable contra otro(s) valor(es), para que en base al resultado de esta comparación, se siga un curso de acción dentro del programa. Cabe mencionar que la comparación se puede hacer contra otra variable o contra una constante, según se necesite. Existen tres tipos básicos, las simples, las dobles y las múltiples.

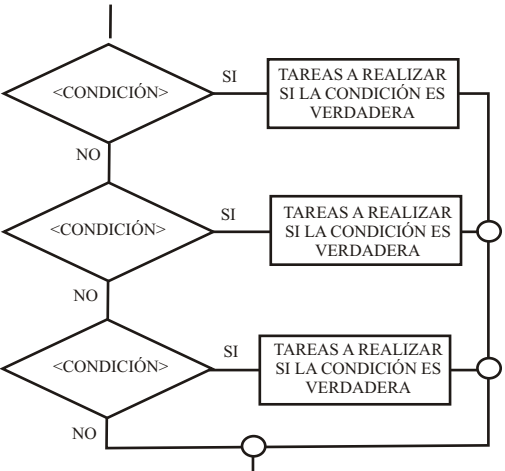
→ **Simples:**

Las estructuras condicionales simples se les conoce como

Pseudocódigo:

Si <condición> entonces
 Instrucción (es)
Si no
 Si <condición> entonces
 Instrucción (es)
 Si no
 .
 . Varias condiciones
 .

Diagrama de flujo:



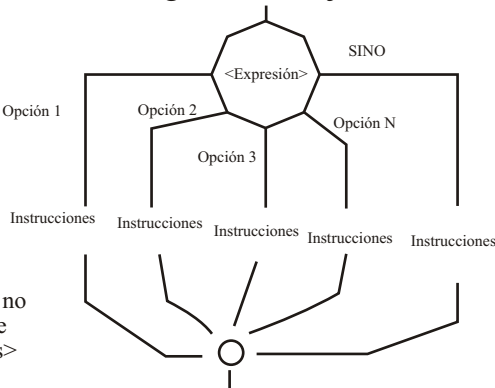
→ Múltiples (En caso de):

Las estructuras de comparación múltiples, es una toma de decisión especializada que permiten evaluar una variable con distintos posibles resultados, ejecutando para cada caso una serie de instrucciones específicas. La forma es la siguiente:

Pseudocódigo:

En-caso-de <expresión> haga
 Caso <opción 1>:
 <instrucciones>
 caso <opción 2>:
 <instrucciones>
 caso <opción 3>:
 <instrucciones>
 ...
 caso <opción N>
 <instrucciones>
SINO <instrucciones a realizar si no se ha cumplido ninguna de las condiciones anteriores>
Fin-Caso

Diagrama de flujo:



```
Imprimir("3.- multiplicación")
imprimir("4.- división")
```

```
imprimir("Ingrese opción : ")
Leer a, b, opción
```

SELECCIÓN (opción)

```
1: imprimir("suma de " + a + "y" + b + " igual a : " + (a + b))

2: imprimir("resta de " + a + "y" + b + " igual a : " + (a - b))

3: imprimir("multiplicación de " + a + "y" + b + " igual a : " + (a * b))

4: SI (b != 0) ENTONCES
    imprimir("división de " + a + "y" + b + " igual a : " + (a / b))
    SINO
        imprimir("división por cero --> infinito")
    FIN SI
```

```
FIN SELECCIÓN
FIN
```

Ejercicio:

```
INICIO
  leer a,b,c
  SI (a == b) ENTONCES
    SI (b == c) ENTONCES
      imprimir("Triángulo Equilátero")
    SINO
      imprimir("Triángulo Isósceles")
  FIN SI
```

```
SINO
  SI (b == c) ENTONCES
    imprimir("Triángulo Isósceles")
  SINO
    imprimir("Triángulo Escaleno")
  FIN SI
FIN SI
FIN
```

Ejercicio:

```
INICIO
  cs <-- 1 /* cantidad de saltos */
  h <-- 0.60 /* altura */

  MIENTRAS( h < 3.0 ) HACER
    cs <-- cs + 1
    h <-- h * 1.3
  FIN MIENTRAS

  imprimir("Cantidad de saltos : " + cs)
  imprimir("Altura máxima alcanzada : " + h)
FIN
```

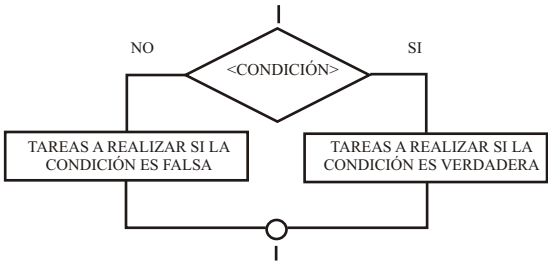
→ Dobles:

Las estructuras condicionales dobles permiten elegir entre dos opciones o alternativas posibles en función del cumplimiento o no de una determinada condición. Se representa de la siguiente forma:

Pseudocódigo:

```
Si <condición> entonces
  Instrucción (es)
Si no
  Instrucción (es)
Fin - Si
```

Diagrama de flujo:



Donde:

- Si: Indica el comando de comparación.
- Condición: Indica la condición a evaluar.
- Entonces: Precede a las acciones a realizar cuando se cumple la condición.
- Instrucción(es): Son las acciones a realizar cuando se cumple o no la condición.
- Si no: Precede a las acciones a realizar cuando no se cumple la condición.

Dependiendo de si la comparación es cierta o falsa, se pueden realizar una o más acciones.

→ Múltiples:

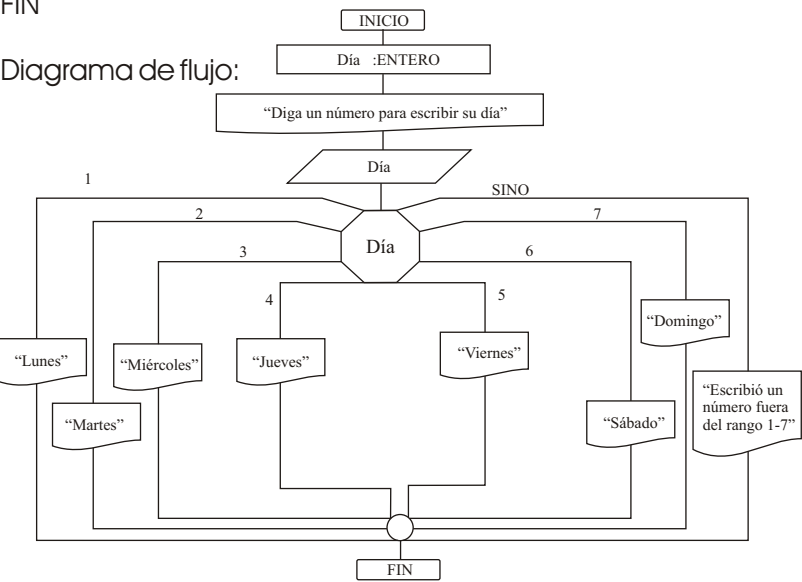
Las estructuras de comparación múltiples, son tomas de decisión especializadas que permiten comparar una variable contra distintos posibles resultados, ejecutando para cada caso una serie de instrucciones específicas. La forma común es la siguiente:

Expresar el algoritmo usando Pseudocódigo y diagrama de flujos.

Pseudocódigo: Pseudocódigo:

INICIO
Día: ENTERO
ESCRIBA "Diga un número para escribir su día"
LEA Día
En-caso-de Día haga
 Caso 1: ESCRIBA "Lunes"
 Caso 2: ESCRIBA "Martes"
 Caso 3: ESCRIBA "Miércoles"
 Caso 4: ESCRIBA "Jueves"
 Caso 5: ESCRIBA "Viernes"
 Caso 6: ESCRIBA "Sábado"
 Caso 7: ESCRIBA "Domingo"
SINO: ESCRIBA "Escribió un número fuera del rango 1-7"
Fin-Caso
FIN

Diagrama de flujo:



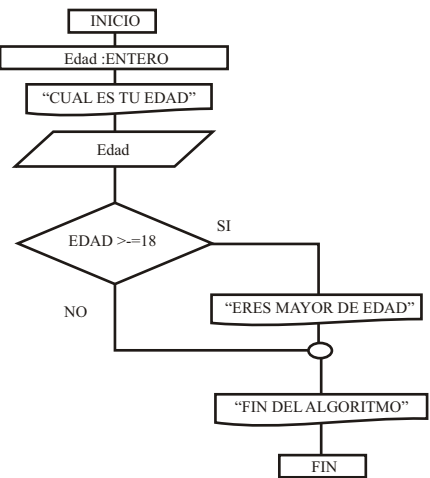
En el siguiente ejemplo se aplique todo lo anterior:

Realizar un algoritmo en donde se pide la edad del usuario; si es mayor de edad debe aparecer un mensaje indicándolo. Expresarlo en Pseudocódigo y Diagrama de flujos.

Pseudocódigo:

INICIO
Edad: Entero
ESCRIBA "¿Cuál es tu edad?"
LEA Edad
Si edad >= 18 entonces
 Escriba "Eres mayor de edad"
Fin-Si
ESCRIBA "Fin del algoritmo"
FIN

Diagrama de flujo:



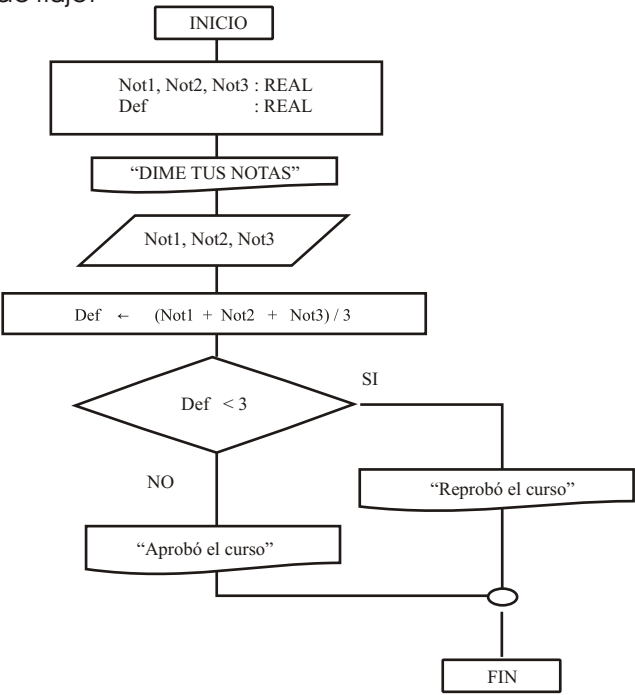
Se pide leer tres notas del alumno, calcular su definitiva en un rango de 0-5 y enviar un mensaje donde diga si el alumno aprobó o reprobó el curso. Expresar el algoritmo usando Pseudocódigo y diagrama de flujos.

Pseudocódigo:

INICIO
Not1, Not2, Not3 :REAL
Def: REAL
LEA Nota1, Nota2, Nota3
Def β (Not1 + Not2 + Not3) /3
Si Def < 3 entonces

Escriba "Reprobó el curso"
Sino
 Escriba "Aprobó el curso"
Fin-Si
FIN

Diagrama de flujo:



Se desea escribir un algoritmo que pida la altura de una persona, si la altura es menor o igual a 150 cm. envíe el mensaje: "Persona de Altura Baja"; si la altura está entre 151 y 170 escriba el mensaje: "Persona de Altura Media" y si la altura es mayor al 171 escriba el mensaje: "Persona Alta" Exprese el

algoritmo usando Pseudocódigo y diagrama de flujos.
Pseudocódigo:

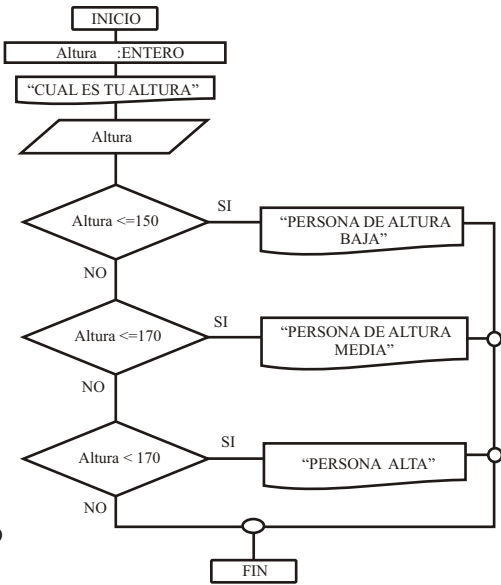
INICIO
 Altura: ENTERO
 ESCRIBA "Cuál es tu altura?"
 LEA Altura
 Si Altura <= 150 entonces
 ESCRIBA "persona de altura baja"
 Sino
 Si Altura <= 170 entonces
 ESCRIBA "persona de altura media"
 Sino
 Si Altura > 170 ENTONCES
 ESCRIBA "persona alta"
 Fin-Si
 Fin-Si
FIN

¡Es importante ser ordenado en el código que se escribe!

Diagrama de flujo:

Dado un numero entre 1 y 7 escriba su correspondiente día de la semana así:

1- Lunes 2- Martes 3- Miércoles
4- Jueves 5- Viernes 6- Sábado
7- Domingo.



Hasta ahora se ha asumido que el incremento de la variable de control es siempre 1, lo cual no es del todo cierto. El incremento de la variable de control es 1 por defecto, esto es, siempre que no se le indique lo contrario.

Sin embargo, existe la posibilidad de establecer este incremento en un número entero cualquiera en función de las necesidades del programa.

Para modificar el incremento y fijarlo en otro número entero que no sea 1, se utiliza una instrucción opcional a la estructura desde. Esta instrucción se halla en la descripción inicial dada metida entre paréntesis, y es la siguiente:

Incremento número 3.

Ejemplo:

Escribir un algoritmo que presente en pantalla los números pares mayores que 0 y menores que 102 en orden creciente, y a continuación en orden decreciente.

Algoritmo Pares

```

Var
  Varcontrol: entero
  Incre: entero
Inicio
  Incre = 2
  desde Varcontrol = 2 hasta 100 incremento = Incre
  hacer
    escribir (Varcontrol)
  Fin-desde
  Incre = -2
  desde Varcontrol = 100 hasta 2 incremento = Incre
  hacer
    escribir (Varcontrol)

```

11.3. OPERACIONES DE REPETICIÓN

Estructuras de repetición

- Una estructura de control que realiza repetición de una serie determinada de sentencias se denomina **bucle**.
- El grupo de sentencias que se repiten dentro de un bucle se denomina **cuerpo del bucle**.
- Cada repetición del cuerpo del bucle se denomina **iteración**.

Las estructuras de control repetitivas que contiene TP son tres:

- Desde (Repeat)
- Mientras (While)
- Repetir/Hasta (For)

11.3.1. ESTRUCTURA DESDE

La variable que sigue a la palabra reservada DESDE, recibe el nombre de variable de control.

```

desde variable = número 1 hasta número 2
hacer

```

```

    acción 1
    acción 2
    :
    acción n

```

```

fin-desde

```

A la variable de control la asignaremos dentro de la propia estructura un valor inicial (número 1).

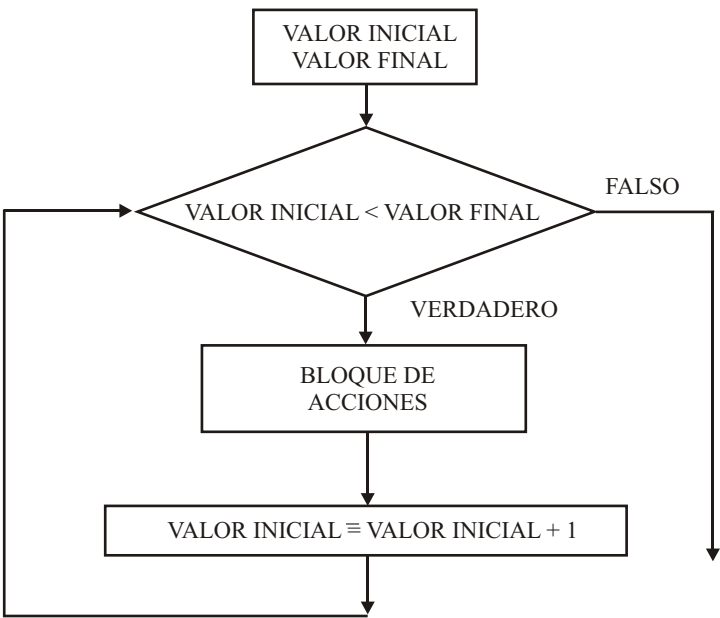
A continuación se ejecuta el cuerpo del bucle, que se halla

delimitado por las palabras reservadas HACER y FIN-DESDE.

Después, automáticamente, sin que el programa se tenga que ocupar de ello, se incrementa en una unidad la variable de control. En este punto se comprueba si el valor que tiene la variable de control es mayor que el numero 2 (este paso también es transparente al usuario).

En caso de que el valor de la variable de control sea superior al valor numero 2, no se vuelve a ejecutar el cuerpo del bucle. En caso contrario, se repite el proceso hasta que la variable de control sea mayor que número 2.

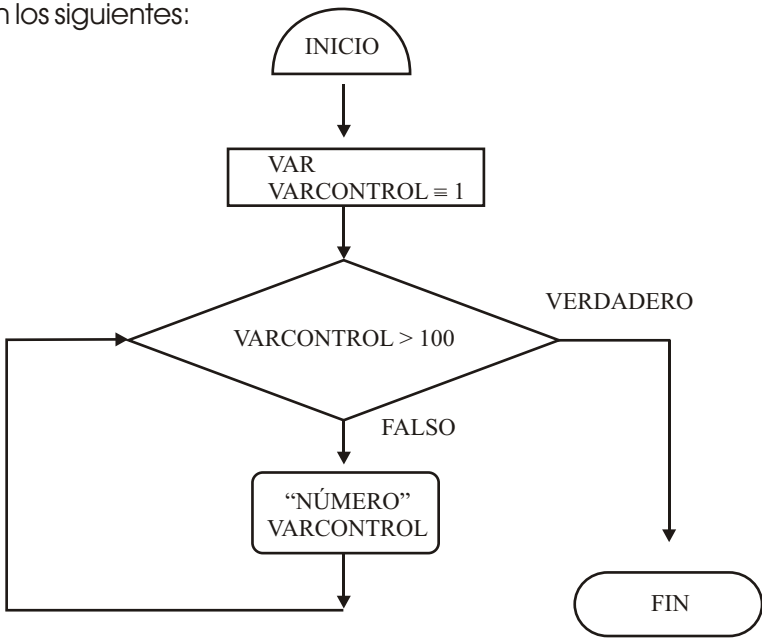
La representación de esta estructura en diagrama de flujo seria.



Ejemplo:
Realizar un algoritmo que muestre los 100 primeros números enteros.
Algoritmos Enteros

Var
 Varcontrol: entero
Inicio
 Desde varcontrol = 1 hasta 100
 hacer
 Escribir ("número", varcontrol)
 Fin-desde
Fin

Los algoritmos de flujo correspondientes al algoritmo "Enteros" son los siguientes:




```

    leer (edad)
    Si edad < 15
        precio = 50
    si no
        precio = 150
    Fin-si
    escribir ("el precio es", precio)
    escribir ("¿Hay más clientes? Si/No")
    leer (Res)
fin-mientras
fin

```

Características Generales:

- El campo del bucle puede no llegar a ejecutarse nunca en el caso de que no se cumpla la condición inicial. La condición se evalúa al principio.
- El número de veces que se ejecuta el campo del bucle va en función de que se deje de cumplir la condición inicial; este hecho ocurre en el bucle.
- El cuerpo del bucle se ejecutará siempre que la condición sea verdadera.

Ejemplo:

Cantidad de dígitos de un número

INICIO

```

Leer número
contador <-- 0
MIENTRAS (número > 0) HACER
    número <-- número / 10
    contador <-- contador + 1

```

```

    fin-desde
Fin

```

Características Generales:

- Si se cumple la condición, el bucle DESDE se ejecuta al menos una vez siempre que el flujo del programa pase por él.
- El bucle desde se ejecuta un número finito de veces que determina el programador.

Ejemplo :

1) Imprimir todos los dígitos (0..9)

```

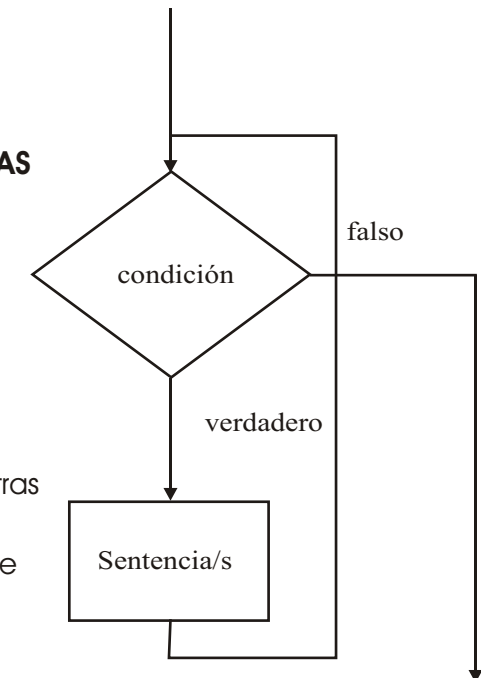
PARA i <-- 0 HASTA 9 HACER
    imprimir(i)
FIN PARA

```

11.3.2. ESTRUCTURA MIENTRAS

SENTENCIA WHILE

La estructura repetitiva While (Mientras) es aquella en la que el número de iteraciones no se conoce por anticipado y el cuerpo del bucle se repite mientras se cumple una determinada condición. También se le conoce como un bucle condicional.



SEUDOCÓDIGO

Mientras condición **hacer**
 Sentencia/s
Fin_mientras

```
mientras <condición> hacer
    acción 1
    acción 2
    :
    acción n
fin-mientras.
```

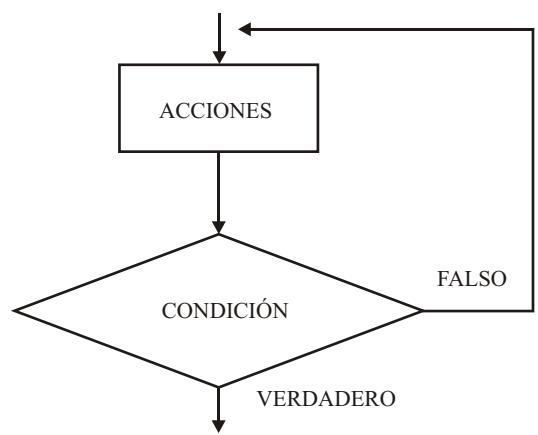
La condición, como ya viene siendo habitual, produce un valor lógico, verdadero o falso, que viene determinado por expresiones booleanas y definido por operadores lógicas, de relación, o por combinaciones de ambos. Cuando el flujo del programa se encuentra con una estructura MIENTRAS, lo primero que hace es evaluar la condición.

Si la condición se verifica, esto es, produce un resultado "verdadero", se ejecuta el cuerpo del bucle, determinado por las palabras reservadas HACER y FIN-MIENTRAS.

Una vez ejecutado el cuerpo del bucle, se vuelve a evaluar la condición y en caso de que se siga verificando, volverá a ejecutarse nuevamente el cuerpo del bucle: este proceso se repetirá hasta que deje de verificarse la condición.

Lógicamente, será necesario modificar la condición dentro del cuerpo del bucle, con el fin de no provocar un bucle infinito.

La representación de esta estructura en diagrama de flujo se encuentra en la siguiente figura:



Ejemplo:
Reescribir el algoritmo Parque de manera que evitemos utilizar la sentencia IR A.

Algoritmo Parque 2.

```
Var
    edad: entero
    precio: entero
    res: cadena
Inicio
    Res = "Si"
    mientras Res="Si" hacer
```

encuentra la palabra reservada SALIR-SI.
→ En este punto se evalúa la condición; si ésta se cumple, se sale del bucle y pasa a la sentencia siguiente, a la palabra reservada FIN-ITERAR.
→ En caso de no cumplirse la condición, se ejecuta el campo de bucle completo y se comienza de nuevo hasta llegar otra vez a la iteración SALIR-SI.

Ejemplo:
Realizar un algoritmo que, al suministrarle un número entre 1 y 12, nos muestre el nombre del mes correspondiente, y que repita este proceso hasta que el número introducido sea menor que 1 ó mayor que 12.

```
Algoritmo Mes
  Var
    número: entero
    nombre: cadena
  Inicio
    Iterar
      leer (número)
      salir-si número < 1 ó número > 12
      según sea número hacer
        1: cadena = "enero"
        2: cadena = "febrero"
        :
        12: cadena = "diciembre"
      Fin-según
      escribir (cadena)
    fin-iterar
  Fin
```

```
FIN MIENTRAS
  imprimir("cantidad de dígitos : " + contador)
FIN
```

11.3.3 ESTRUCTURA REPETIR/HASTA

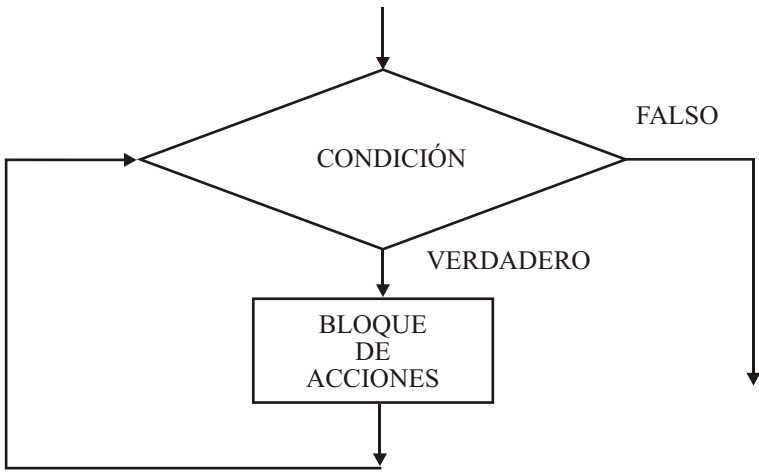
```
repetir
  acción 1
  acción 2
  :
  acción n
hasta que <condición>
```

El cuerpo del bucle se halla comprendido entre las palabras reservadas.

REPETIR y HASTA QUE

La forma de trabajar de esta estructura es la siguiente:
→ Se ejecutan todas las acciones del cuerpo del bucle, al llegar a la palabra reservada hasta que se evalúa la condición.
→ Si la condición no se satisface (el resultado es "falso"), se vuelve a la palabra reservada REPETIR y se ejecuta de nuevo el cuerpo del bucle.
→ Este proceso se repetirá hasta que se satisfaga la condición.

La sintaxis de esta estructura en diagramas de flujo se halla en la figura siguiente:



Ejemplo:
Realizar un algoritmo que calcule el factorial de un número introducido por teclado.

Algoritmo Factorial

```
Var
    número: entero
    acumulador: entero
Inicio
    leer (número)
    acumulador = 1
    repetir
        acumulador = acumulador * número
        número = número - 1
    hasta que número = 1
Fin
```

- Características Generales:
- El bucle se ejecuta al menos una vez.
 - El bucle se ejecuta mientras la condición es falsa.
 - La condición se evalúa al final.
 - El número de veces que se ejecutará el bucle depende de que se cumpla o no la condición.

11.3.4. ESTRUCTURA ITERAR

La estructura ITERAR, por sus características, se sale del grupo de estructuras de control de flujo recomendadas para la práctica de programación estructurada.

No obstante, y dado que la utilizan muchos lenguajes de programación, se va a estudiar aquí.

La principal característica de esta estructura radica en que permite la salida del bucle en un punto cualquiera del interior del mismo, mediante una instrucción (palabra reservada) determinada.

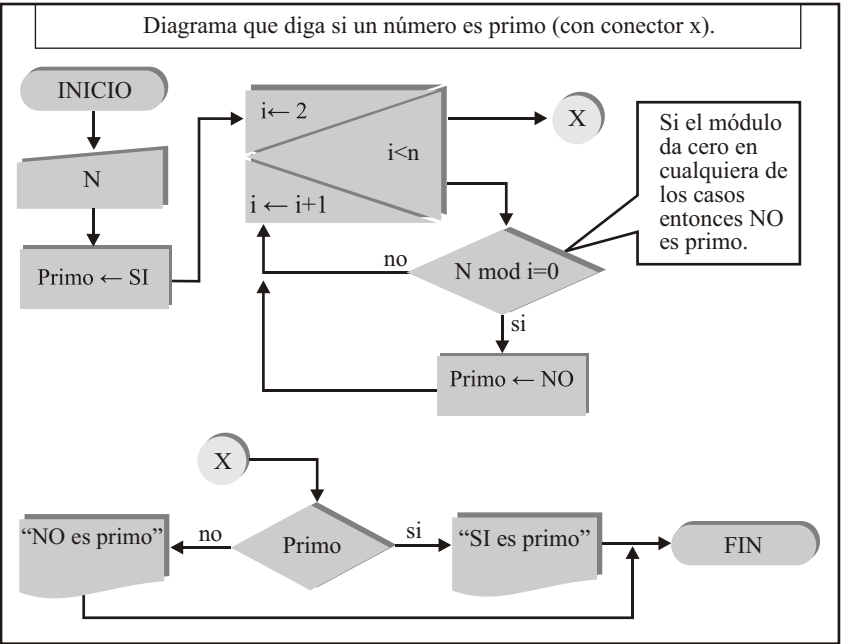
```
Iterar
    acción 1
    acción 2
    :
    acción x
    salir-si <condición>
    acción x + 2
    :
    acción n
Fin-iterar
```

- Funcionamiento:
- El cuerpo de bucle comienza a ejecutarse hasta que

El diagrama deberá de solicitar los datos necesarios y mostrar el resultado.

NOTA: Se dice que un número es primo cuando solamente es divisible por 1 y por si mismo. Para determinar esto, tenemos que probar el módulo contra todos los números desde 2 hasta N-1 y si alguno es igual 0 quiere decir que si es divisible y por lo tanto no es primo.

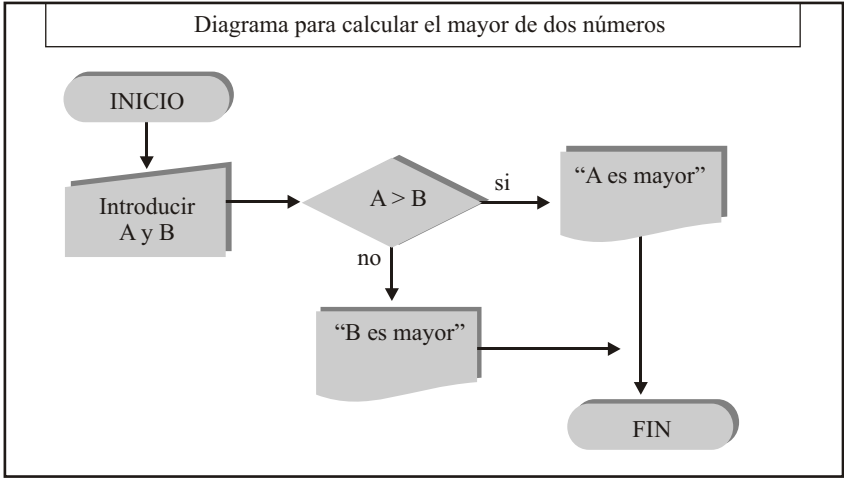
RESPUESTA:



Ejercicio No. 6

Diseñar un diagrama de flujo que pida cuatro calificaciones, las analice y determine si el alumno a que pertenecen reprobó o aprobó. Para considerar aprobado a un alumno, sus calificaciones deben cumplir las siguientes.

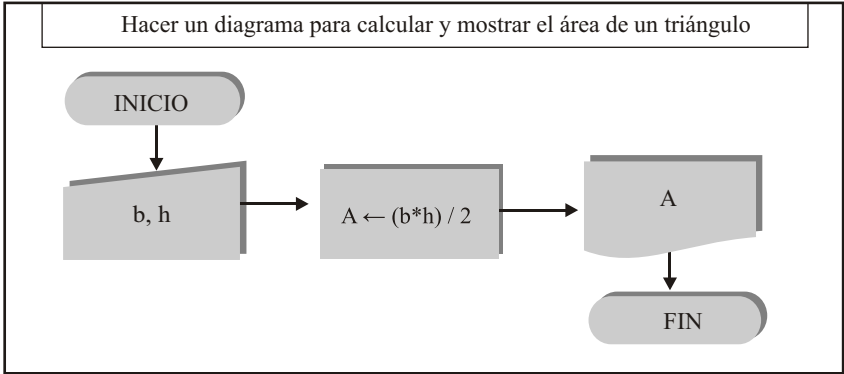
EJEMPLO DE DIAGRAMAS DE FLUJO



Ejercicio No. 1

Diseñar un diagrama de flujo para calcular y mostrar el área de un triángulo. El diagrama deberá de solicitar los datos necesarios y mostrar el resultado.

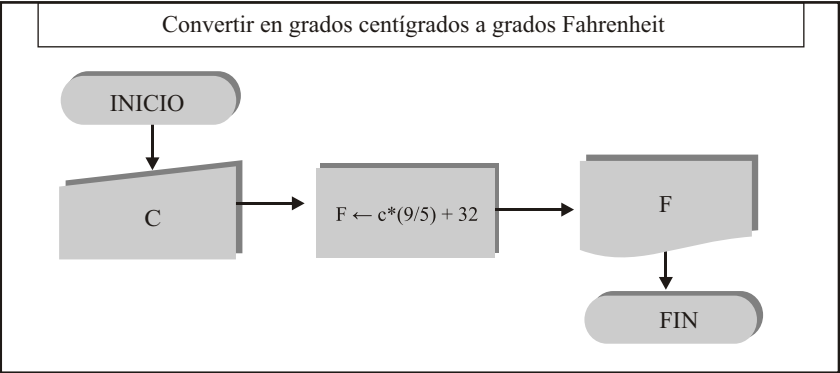
RESPUESTA:



Ejercicio No. 2

Diseñar un diagrama de flujo para convertir de grados centígrados a grados Fahrenheit.
El diagrama deberá de solicitar los datos necesarios y mostrar el resultado.

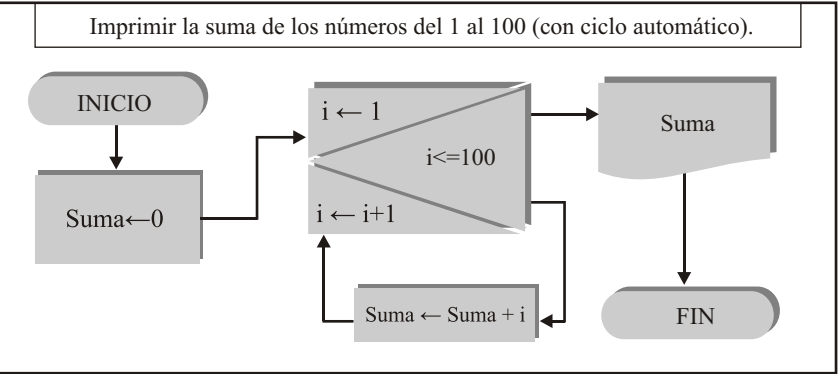
RESPUESTA:



Ejercicio No. 3

Diseñar un diagrama de flujo para calcular e imprimir la suma de los números del 1 al 100, utilizando un ciclo automático.
El diagrama deberá de solicitar los datos necesarios y mostrar el resultado.

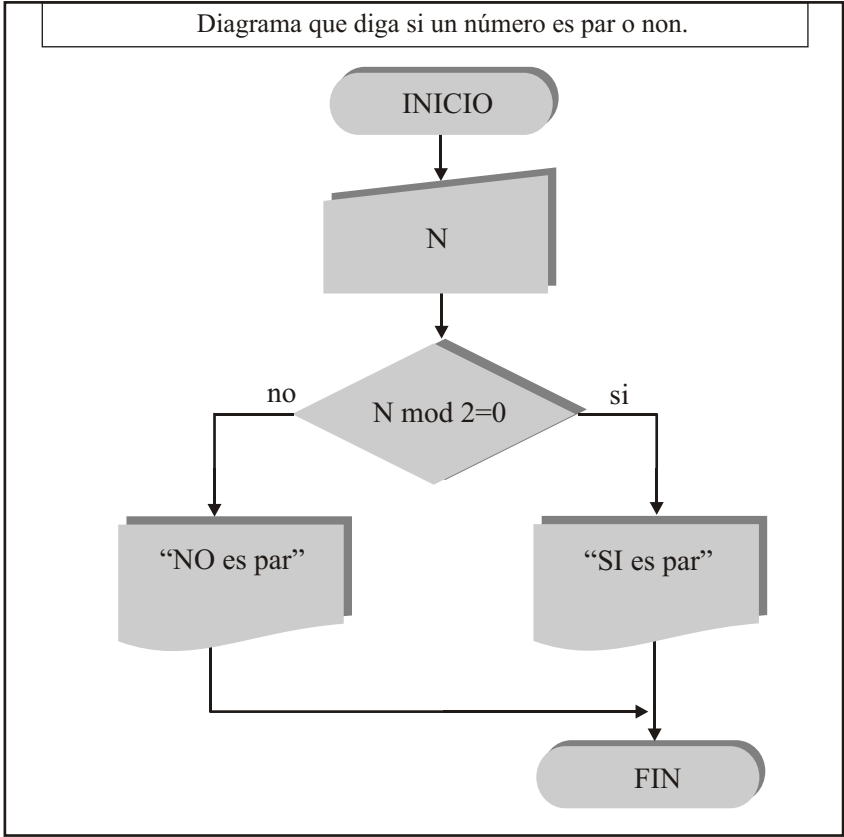
RESPUESTA:



Ejercicio No. 4

Diseñar un diagrama de flujo que solicite un número y determine si este es par o es non.
El diagrama deberá de solicitar los datos necesarios y mostrar el resultado.

RESPUESTA:



Ejercicio No. 5

Diseñar un diagrama de flujo que solicite un número y determine si este es un número primo.

```
gan=cap_inv*.02;
printf("\nLa ganancia es:%F", gan);
}
```

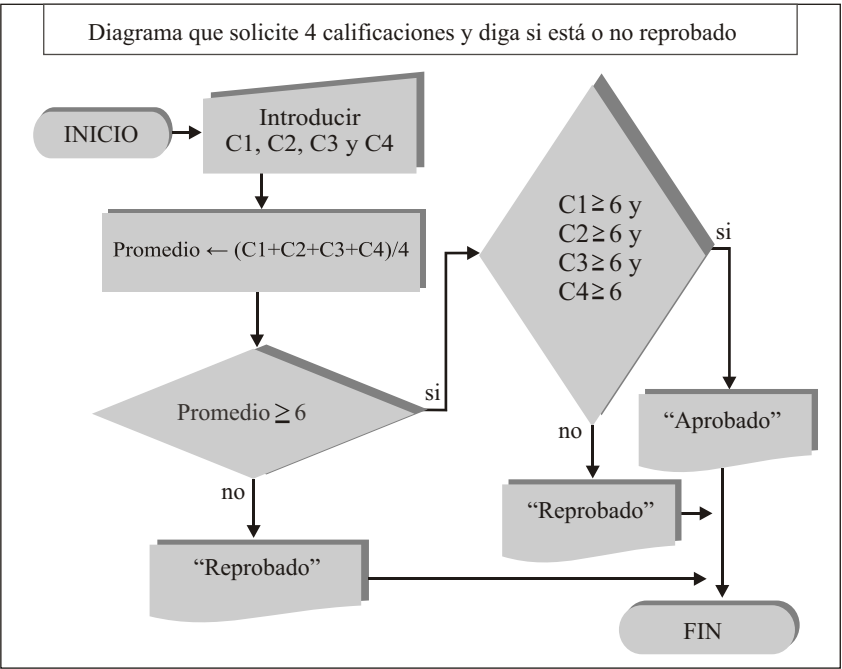
EJERCICIO 2

Un vendedor recibe un sueldo base más un 10% extra por comisión de sus ventas, el vendedor desea saber cuanto dinero obtendrá por concepto de comisiones por las tres ventas que realiza en el mes y el total que recibirá en el mes tomando en cuenta su sueldo base y comisiones.

```
Inicio
Leer sb, v1, v2, v3
tot_vta = v1 + v2 + v3
com = tot_vta * 0.10
tpag = sb + com
Imprimir tpag, com
Fin
#include <stdio.h>
main()
{
float sb,v1,v2,v3,com,tot_ven,tot_pag;
sb=5000;
printf("\nIngresa la venta 1:");
scanf ("%f",&v1);
printf("\nIngresa la venta 2:");
scanf ("%f",&v2);
printf("\nIngresa la venta 3");
scanf ("%f",&v3);
tot_ven=v1+v2+v3;
com=tot_ven*.10;
tot_pag=sb+com;
printf("\nTotal de pago:%f",tot_pag);
printf("\nComisión:%f",com);
}
```

Reglas:

- 1. El promedio de las 4 calificaciones debe ser Mayor o igual a 6.0.
- 2. Ninguna de las calificaciones individuales debe ser menor que 6.

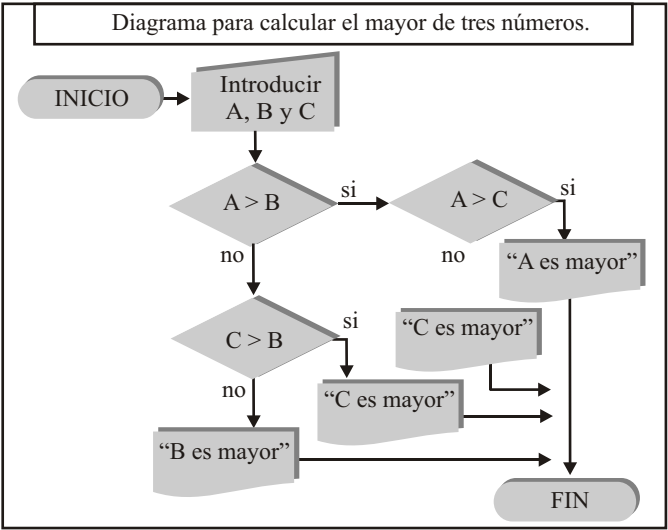


Ejercicio No 7.

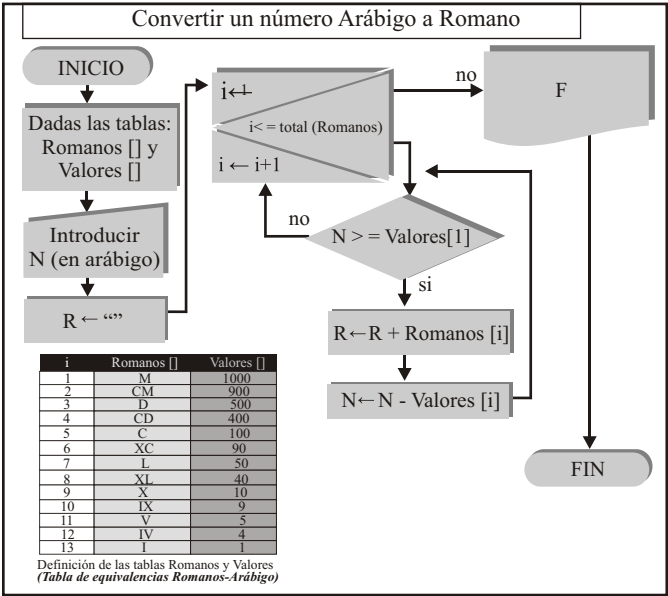
Diseñe un diagrama de flujo que solicite tres números A B y C y que diga cual es el mayor de ellos.

El diagrama deberá de solicitar los datos necesarios y mostrar el resultado.

Respuesta:



Ejercicio 8



CODIFICACIÓN

De acuerdo al término de codificación (Pag 22) Existen muchos lenguajes de programación donde se pueden transcribir los diagramas o pseudocódigos, encontraras algunos ejemplos en diferentes lenguajes de programación. A continuación se presentan algunos ejercicios los cuales son desarrollados en pseudocódigo y posterior mente codificados en lenguaje de programación.

EJERCICIO 1

Suponga que un individuo desea invertir su capital en un banco y desea saber cuanto dinero ganará después de un mes si el banco paga a razón de 2% mensual.

Inicio
Leer cap_inv
gan = cap_inv * 0.02
Imprimir gan
Fin

```
#include <stdio.h>
main()
{
    int cap_inv;
    float gan;
    printf("Cantidad a invertir:");
    scanf ("%i",& cap_inv);
```



```
printf("resultado %d",a-b);
break;
case '*':
printf("resultado %d",a*b);
break;
case '/':
printf("resultado %d",a/b);
break;
default:
printf("error");
}
}
```

EJERCICIO 7

```
/* Leer un número entero y visualizar su tabla de multiplicar. */
#include <stdio.h>
void main()
{
int a,b;
printf ("Introduce el número:");
scanf("%d",&a);
b= 1;
while (b<= 10)
{
printf("%d * %d = %d\n",a,b,a*b);
b++;
}
}
```

EJERCICIO 8

```
/* Leer un entero positivo y averiguar si es perfecto. Un n° es
perfecto cuando es igual a la suma de sus divisores excepto el
mismo */
#include <stdio.h>
```

EJERCICIO 3

Una tienda ofrece un descuento del 15% sobre el total de la compra y un cliente desea saber cuanto deberá pagar finalmente por su compra.

```
Inicio
Leer tc
d = tc * 0.15
tp = tc - d
Imprimir tp
Fin
#include <stdio.h>
main()
{
float tc,d,tp;
printf ("\nIngresa la compra:");
scanf ("%f",& tc);
d=tc*.15;
tp=tc-d;
printf ("\nEl total de la compra es:%f",tp);
}
```

EJERCICIO 4

Un alumno desea saber cual será su calificación final en la materia de Física. Dicha calificación se compone de los siguientes porcentajes:

- 55% del promedio de sus tres calificaciones parciales.
- 30% de la calificación del examen final.
- 15% de la calificación de un trabajo final.

```
Inicio
Leer c1, c2, c3, ef, tf
Prom = (c1 + c2 + c3)/3
Ppar = prom * 0.55
```

```

Pef = ef * 0.30
Ptf = tf * 0.15 Cf = ppar + pef + ptf
Imprimir cf
Fin

```

```

#include <stdio.h>
main()
{
float c1,ef,c2,c3,cf,tf,prom,ppar,pef,ptf;
printf("Ingrese la primer calificación:");
scanf ("%f",& c1);
printf ("\nIngrese la segunda calificación:");
scanf ("%f",& c2);
printf ("\nIngrese la tercer calificación:");
scanf ("%f",& c3);
printf ("\nIntroduce la calificación del trabajo final:");
scanf ("%f",& tf);
printf ("\nIntroduce la calificación de tu examen final:");
scanf ("%f",& cf);
prom=(c1+c2+c3)/3;
ppar=prom*.55;
pef=ef*.15;
ptf=tf*.15;
cf=ppar+pef+ptf;
printf ("\nTu calificación final es:%f", cf);
}

```

EJERCICIO 5

```

/*Leer dos números enteros y escribir el mayor de ambos o un
mensaje si son
iguales.*/
#include <stdio.h>
void main()

```

```

{
int a,b;
printf("Dame dos números... ");
scanf ("%d%d",&a,&b);
if (a>b)
printf("El mayor número es... %d",a);
else
if (b>a)
printf("El número mayor es... %d",b);
else
printf("Los números son iguales");
}

```

EJERCICIO 6

/* Leer un carácter y dos números enteros. Si el carácter leído es un operador aritmético calcular la operación correspondiente, si es cualquier otro mostrar error. Hacer el programa utilizando la instrucción **switch()** */

```

#include <stdio.h>
void main()
{
char c;
int a,b;
printf("dame un carácter\n");
c=getchar();
fflush (stdin);
printf("dame dos números\n");
scanf ("%d %d",&a,&b);
switch (c)
{
case '+':
printf("resultado %d",a+b);
break;
case '-':

```

```

System.out.print("\n\t\tseguir(1)/salir(0) : ");
seguir = Integer.parseInt(in.readLine());

}while (seguir == 1);
}
}

```

PROGRAMA 2 EN JAVA

```
import java.io.*;
```

```
public class Anillo {
```

```

    private static double redondear(double num) {
        double aux = num * 1000;
        int tmp = (int) aux;
        return (double) tmp / 1000;
    }

```

```

    public static void main(String[ ] args) throws IOException {
        double distancia = 0.0;
        double tiempo = 0.0;
        double tiempoAcum = 0.0;
        double radio;
        int k;
        double velocidad;
        int seguir = 1;

```

```

        BufferedReader in = new BufferedReader(new
        InputStreamReader(System.in));

```

```

        do {
            tiempoAcum = 0.0; /* en vcada ciclo se debe

```

```

void main()
{
    int i,número,suma=0;
    do{
        printf("Dame un número:");
        scanf("%d",&número);
    } while (número<=0);
    for (i=1;i<=(número/2);i++)
        if ((número%i)==0)
            suma+=i;
    if (número==suma)
        printf("El número es perfecto.");
    else
        printf("El número NO es perfecto.");
}

```

EJERCICIO 9

/* Leer dos números enteros a y b mayores o iguales que cero y calcular su producto mediante sumas sucesivas. (Se usan, a modo de ejemplo, los tres esquemas repetitivos existentes en C.*/

```
#include <stdio.h>
```

```

void main()
{
    int a,b,c,i,producto=0;
    do{
        printf("Dame un a:");
        scanf("%d",&a);
    } while (a<0);
    do{
        printf("Dame b:");
        scanf("%d",&b);
    } while (b<0);
}

```

// Suponemos que a es mayor que b en caso contrario se hace

```

un intercambio
if (a<b)
{
c=a;
a=b;
b=c;
}
printf("a=%d b=%d\n",a,b);
for (i=0;i<b;i++)
producto+=a;
printf("El producto con for es: %d\n",producto);
producto=0;
i=0;
while (i<b)
{
producto+=a;
i++;
}
printf("El producto con while es: %d\n",producto);
producto=0;
i=0;
do /* con este bucle al multiplicar por cero */
{ /* me daría "a" */
producto+=a;
i++;
} while (i<b);
printf("El producto con do-while es: %d",producto);
}

```

PROGRAMA 1 EN JAVA

```
import java.io.*;
```

```

public class Trampolin {
public static void main(String[] args) throws IOException {

```

```

int cs = 1;
double h = 0.60;
int seguir = 1;
double sep,aumento;

```

```

BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));

```

```
do {
```

```

h = 0.6;
cs = 1;

```

```

System.out.print("\n\nseparacion entre trampolines : ");
sep = Double.parseDouble(in.readLine());

```

```

System.out.print("porcentaje de aumento (0..1) : ");
aumento = Double.parseDouble(in.readLine());

```

```
aumento += 1.0;
```

```

System.out.println("\n\tcs\taltura");
System.out.println("\t--\t-----");

```

```
while ( h < sep ) {
```

```

System.out.println("\t" + cs + "\t" + h);

```

```

cs++;
h *= aumento;
}

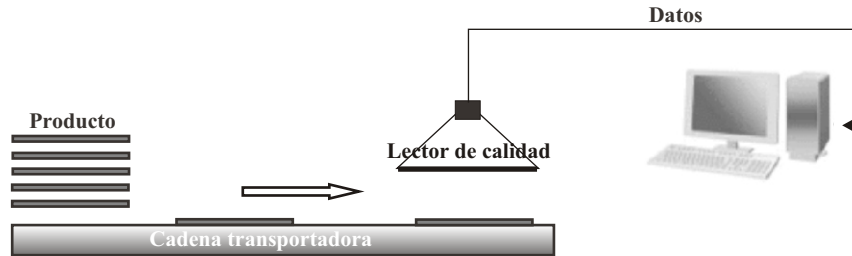
```

```

System.out.println("\n\tCantidad de saltos : " + cs);
System.out.println("\tAltura maxima alcanzada : " + h);

```

0 : Producto MALO
1 : Producto DEFECTUOSO
2 : Producto BUENO



Especificaciones

- a) Los datos que llegan en forma iterativa (mientras no se haya acabado el lote de productos a revisar), son rescatados por el método **test()**.
- b) El término de la revisión está determinado cuando el método **fin()** retorne el valor **true**.
- c) El algoritmo debe llevar un conteo de cuántos productos corresponden a cada estado {MALO : 2 , DEFECTUOSO : 5 , BUENO : 43}, para posteriormente aplicar un cálculo de porcentaje.
- d) Inicialmente no se sabe la cantidad total de productos del lote, por lo tanto es fundamental que se implemente un contador para los cálculos posteriores.
- e) El precio original de compra del lote, tiene que ser ingresado por el usuario. Después del análisis, se deberá imprimir según los **criterios** el precio propuesto de compra.

Una vez terminado el proceso de análisis y conteo, se determinará el

inicializar en cero */

```
System.out.print("\n\ningrese número de anillos : ");
k = Integer.parseInt(in.readLine());
```

```
System.out.print("ingrese radio inicial : ");  
radio = Double.parseDouble(in.readLine());
```

```
System.out.print("ingrese velocidad inicial : ");  
velocidad = Double.parseDouble(in.readLine());
```

```
System.out.println("\n\nk\radio\t\tDistancia\tVelocidad\tTiempo");  
System.out.println("---\t---\t\t-----\t-----\t-----");
```

```
while (k > 0) {
```

```
distancia = 2.0 * Math.PI * radio;
```

$$\text{tiempo} = \text{distancia} / \text{velocidad};$$

```
tiempoAcum = tiempoAcum + tiempo;
```

```
System.out.println(k + "\t" + redondear(radio) + "\t" +
redondear(distancia) + "\t"
redondear(velocidad) + "\t\t" + tiempo);
```

```
velocidad *= 1.12;
```

```
radio = radio * 0.65;
```

 $k_{-};$

```

}
velocidad = 2.0 * velocidad * Math.pow(1.0 + radio,

```

```

2.0) ;

    System.out.println("\n\tVelocidad disparo : " +
redondear(velocidad));
    System.out.println("\n\ttiempo total : " + tiempoAcum);

    System.out.print("\n\n\tseguir(1)/salir(0) : ");
    seguir = Integer.parseInt(in.readLine());

}while (seguir == 1);
}
}

```

EJEMPLOS EN PSEUDOCÓDIGO

1. Problema de detención en un semáforo

Escribir el algoritmo, que describe las acciones que el conductor realiza ante el estado verde, amarillo o rojo de un semáforo. Tener en cuenta, que ante una luz amarilla, el conductor acelerará si es arriesgado, o desacelerará si es que es prudente.

Variables

luz : {VERDE , AMARILLA , ROJA}
 estaDetenido : {true,false}

Métodos

parte() , pasa() , acelera() , desacelera() , frena() , estaDetenido() , espera()

INICIO

leer luz

SELECCIÓN (luz)

VERDE : SI (estaDetenido == false) ENTONCES

```

    parte()
    SINO
    pasa()
    FIN SI

```

```

    AMARILLA : SI (conductorEsArriesgado == true) ENTONCES
        acelerarara()
    SINO
        desacelera()
        frena()
    FIN SI

```

```

    ROJA : SI (estaDetenido == false) ENTONCES
        desacelera()
        frena()
    SINO
        espera()
    FIN SI

```

FIN SELECCIÓN

FIN

2. Problema de compra en función de la calidad de un lote de productos

Se requiere implementar un algoritmo, que permita registrar la calidad de un lote de productos, para agilizar el proceso de compra.

El sistema consiste, en un lector de calidad que va leyendo mediante sensores a un producto dejado a través de una correa transportadora, para transmitir a continuación los datos a un software especial, depositado en un computador, que entrega hacia afuera uno de los siguientes estados numéricos:

| | | | | |
|---|-------|-------|-------|---------------------|
| 5 | 0.58 | 3.645 | 3.524 | 1.0341809815535743 |
| 4 | 0.377 | 2.369 | 3.947 | 0.6001943196516281 |
| 3 | 0.245 | 1.54 | 4.421 | 0.3483270605121055 |
| 2 | 0.159 | 1.001 | 4.951 | 0.20215409761863268 |
| 1 | 0.103 | 0.65 | 5.546 | 0.11732157451081357 |

Velocidad disparo : 14.152
tiempo total : 37.26948901419321

Algoritmo

INICIO

```
PI <-- 3.14 /* constante */
k <-- 10
radio <-- 5.0
velocidad <-- 2.0
distancia <-- 0
tiempo <-- 0
tiempoAcum <-- 0

imprimir("k radio Distancia Velocidad Tiempo");
imprimir("-----");

MIENTRAS (k > 0) HACER

    distancia <-- 2 * PI * radio
    tiempo <-- distancia / velocidad
    tiempoAcum <-- tiempoAcum + tiempo
    imprimir(k + " " + radio + " " + distancia + " " +
Velocidad" + " " + tiempo)
    velocidad <-- 1.12 * velocidad
    radio <-- radio * 0.65
```

precio de compra propuesto para el lote en cuestión.

Criterios

- 1) Si el porcentaje de productos malos es superior al 15%, el lote no se compra (precio <-- 0). En caso contrario, pasar al punto 2).
- 2) Si el porcentaje de productos defectuosos es menor a un 10%, se compra a un 70% del precio original. En caso contrario, pasar al punto 3).
- 3) Si el porcentaje de productos buenos es mayor o igual a un 70%, el lote se compra al precio original. En caso contrario, se estaría comprando a un 60% del precio original.

Algoritmo

```
INICIO
N <-- 0 /* cantidad de productos analizados */
contMalos <-- 0
contDefect <-- 0
contBuenos <-- 0
precio <-- 0 /* precio en función del análisis */

leer precioInicial

MIENTRAS (fin() != true) HACER

    calidad <-- test()

    SELECCIÓN (calidad)

        0 : contMalos <-- contMalos + 1
        1 : contDefect <-- contDefect + 1
        2 : contBuenos <-- contBuenos + 1
```

```
FIN SELECCIÓN

N <-- N + 1

FIN MIENTRAS

SI (contMalos / N > 0.15) ENTONCES
    precio <-- 0
SINO
    SI (contDefect / N < 0.1 ) ENTONCES
        precio <-- precioInicial * 0.7
    SINO
        SI (contBuenos / N >= 0.7 ) ENTONCES
            precio <-- precioOriginal
        SINO
            precio <-- precioOriginal * 0.6
        FIN SI
    FIN SI
FIN SI

imprimir(precio)

FIN
```

3. Problema del electrón en los anillos

a) Un electrón se mueve en un conjunto de 10 anillos, con una velocidad constante en cada uno de ellos . Una característica de la trayectoria, es que una vez que completa el recorrido en un anillo, cambia de estado pasándose al anillo siguiente o más cercano al centro, con una velocidad 1.12 veces mayor a la anterior.

El electrón, después de recorrer el último anillo y llegar

teóricamente al centro ($k = 0$), sale disparado perpendicularmente al radio del anillo, a una velocidad igual al doble de velocidad del radio uno ($k = 1$) por el último radio al cuadrado.

Además, la trayectoria cuenta con la característica que el radio de cada anillo, es un 35% menor con respecto al anterior, siendo el radio inicial de 5.0 nm.

Finalmente, se sabe que el electrón tiene una velocidad inicial de 2.0 ns.

Datos

Para $k : 1, \dots, 10$ se tiene que
radio de cada anillo : $\text{radio_}k-1 = \text{radio_}k * 0.65$
Velocidad constante en cada anillo : $v_{k-1} = 1.12 * v_k$
 $\text{Tiempo_}k = \text{Distancia_}k / \text{Velocidad_}k$
radio inicial = 5.0 [nm]
 $\text{velocidad disparo} = 2 * v_1 * (1 + \text{radio})^2$

Determinar

Velocidad, distancia y tiempo en recorrer cada anillo, además del tiempo acumulado para recorrer los 10 anillos y la velocidad de disparo del electrón.

El algoritmo debe generar la siguiente salida :

ingrese numero de anillos : 10
ingrese radio inicial : 5.0
ingrese velocidad inicial : 2.0

| k | radio | Distancia | Velocidad | Tiempo |
|-----|-------|-----------|-----------|--------------------|
| --- | ---- | ----- | ----- | ----- |
| 10 | 5.0 | 31.415 | 2.0 | 15.707963267948966 |
| 9 | 3.25 | 20.42 | 2.24 | 9.11622868229181 |
| 8 | 2.112 | 13.273 | 2.508 | 5.2906684316872115 |
| 7 | 1.373 | 8.627 | 2.809 | 3.07047721481847 |
| 6 | 0.892 | 5.607 | 3.147 | 1.781973383600005 |


```
k <-- k - 1
FIN MIENTRAS

velocidad <-- 2 * velocidad * (1 + radio) * (1 + radio)

imprimir("Velocidad disparo : " + velocidad)
imprimir("tiempo total : " + tiempoAcum)

FIN
```

Ejercicios propuestos

1. Hacer un diagrama para calcular el área de un triángulo.
2. Hacer un diagrama para convertir de grados centígrados a grados Fahrenheit.
3. Hacer un diagrama para imprimir la suma de los números del 1 al 100.
4. Hacer un diagrama que te pida un número y te diga si es par, es non y/o es primo.
5. Hacer un diagrama para imprimir la sucesión de Fibonacci.
6. Hacer un diagrama que pida 10 números y muestre el promedio.
7. Hacer un diagrama que pida 3 números y diga cual es el mayor.
8. Hacer un diagrama que pida la edad y despliegue si es menor de edad (<18), mayor (≥ 18) o si pertenece a la 3ª edad (≥ 60).
9. Hacer un diagrama que te pida un número y te diga si es par, es non y/o es primo.
10. Hacer un diagrama para calcular el factorial de un número.
11. Hacer un diagrama que calcule e imprima N números primos.
12. Hacer un diagrama que solicite 4 calificaciones y diga si está reprobado o no, según las reglas de tu escuela.

13. Hacer un diagrama que pida un número N y despliegue todas las combinaciones de dos números que sumados den N .
14. Hacer un diagrama que despliegue la tabla de multiplicar de un número X .
15. Hacer un diagrama que calcule la probabilidad de que dos dados lanzados sumen 7.
16. Hacer un diagrama que pida 100 números y diga cual es la mediana.
17. Hacer un diagrama que solicite los datos de una matriz de 4×4 y la muestre invertida.
18. Hacer un diagrama que pida 3 números y calcule el común denominador.
19. Hacer un diagrama que llene una matriz de 3×3 y despliegue los valores de la diagonal principal.
20. Hacer un diagrama que pida 2 matrices y despliegue el producto cruz de las mismas.

Bibliografía

<http://juegosdelogica.net/ingeniero/ingenio/ingenio.php>

http://es.wikipedia.org/wiki/diagrama_de_flujo

<http://www.monografias.com/trabajos59/diagrama-flujo/diagrama-flujo.shtml>

<http://www.adrformacion.com/cursos/calidad/leccion3/tutorial2.html>

<http://pjsml.50megs.com/java/algoritmos.html>
[/mx.geocities.com/ademar_17/Diagramas.doc](http://mx.geocities.com/ademar_17/Diagramas.doc)

<http://jose87.iespana.es/c.html>

<http://www.mailxmail.com/curso/informatica/cplusplus2>

<http://www.fismat.umich.mx/rmn1/manual/>