UT del Valle del Mezquital

Lógica de Programación

01/09/2008

Mtra. Gloria Zúñiga Chávez

Contenido

Conceptos básicos

Lógica: Es la ciencia que estudia el razonamiento, el producto o resultado de razonar, las leyes de la lógica determinan cuando ciertas creencias constituyen una buena razón para afirmar una conclusión. La lógica investiga la relación de *consecuencia* que se da entre una serie de premisas y la conclusión de un argumento correcto. Se dice que el argumento *es correcto* si su conclusión *se sigue* o es *consecuencia* de sus premisas; de otro modo es *incorrecto*. (Iranzo, 2005)

Ejemplos:

Todos los hombres son mortales

Todos los griegos son hombres

Por lo tanto, Todos los griegos son mortales

En una despensa hay exactamente 18 botes de garbanzos;

Cada bote contiene al menos 150 garbanzos;

Ningún bote contiene más de 155 garbanzos;

Por lo tanto, hay en la despensa al menos 3 botes que contienen el mismo número de garbanzos.

Funcionamiento de una computadora

Proceso de programación (Farrell, 2001)

1. Entender el problema
2. Plantear la lógica
3. Codificar el programa
4. Traducir el programa a lenguaje máquina
5. Probar el programa
6. Poner el programa en ejecución

Metodología para la solución de problemas por medio de computadora (Cabrera)

1. Definición del problema
2. Análisis del problema
3. Diseño del algoritmo
4. Codificación
5. Prueba y depuración
6. Mantenimiento
7. Documentación

Tipos y estructuras de datos

Dato: cualquier objeto manipulable por la computadora, puede ser un carácter leído de un teclado, información almacenada en un disco, un número que se encuentra en la memoria principal, etc.

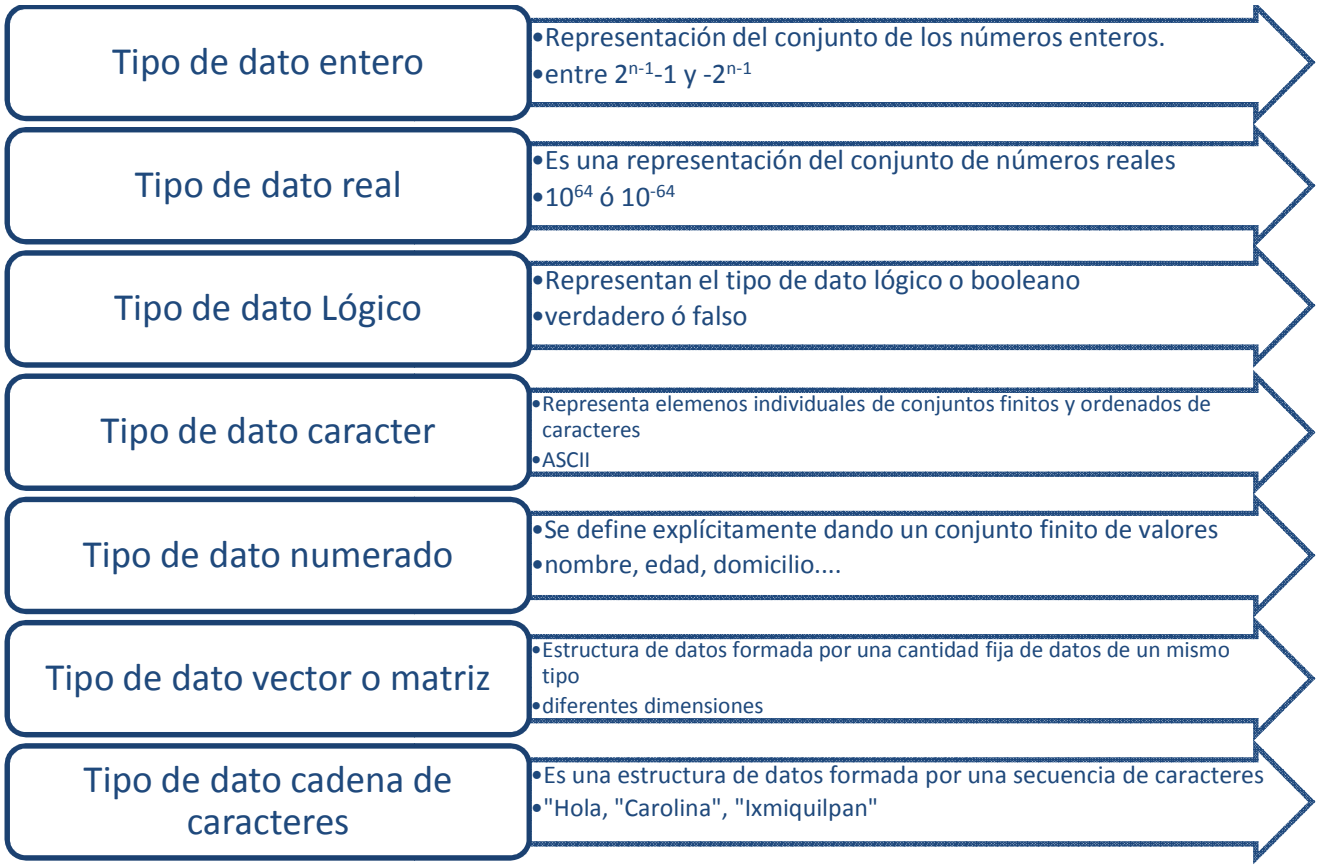


Fig. 1.1 Tipos y estructuras de datos

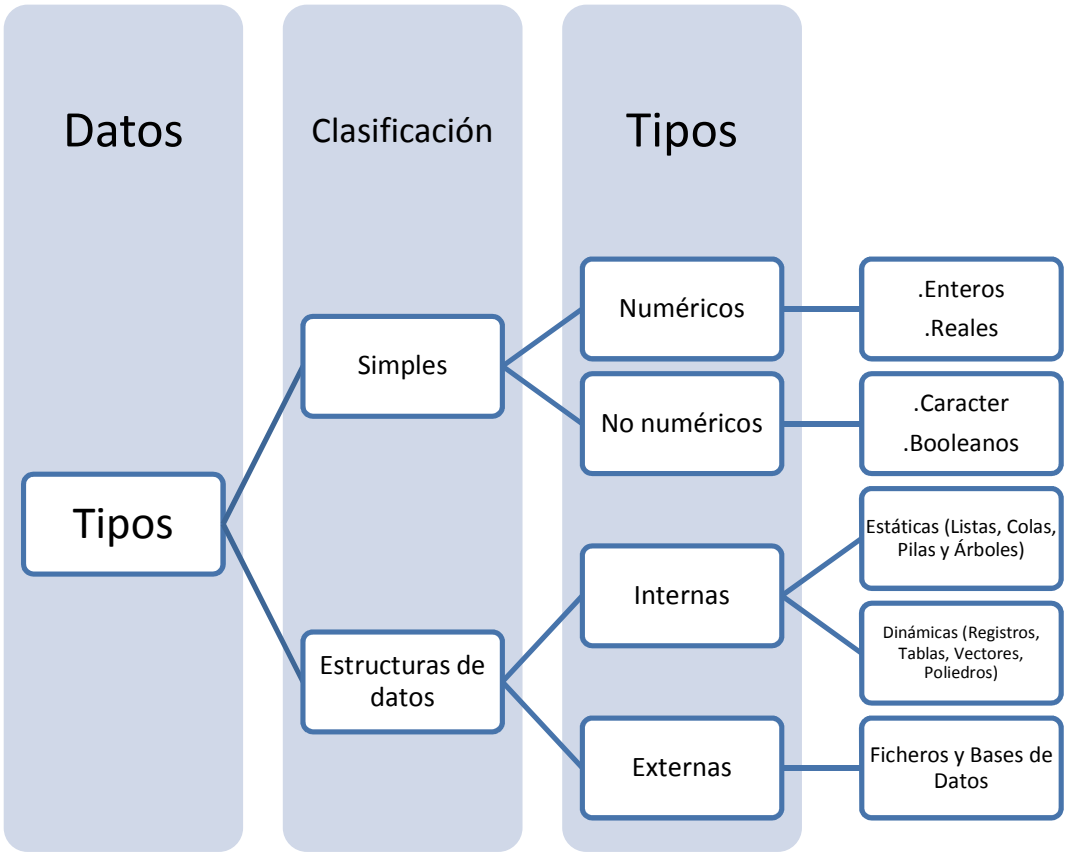


Fig. 1.2 Tipos y estructuras de de datos

Constante: No alteran su valor en le ejecución de un programa.

Variable: Alteran su valor en la ejecución de un programa

Identificadores: En la mayoría de los programas, es necesario manejar datos de entrada o de salida, los cuales necesitan almacenarse en la memoria principal de la computadora en el tiempo de ejecución. Los identificadores también se utilizan para los nombres de los programas, los nombres de los procedimientos y los nombres de las funciones, así como para las etiquetas, constantes y variables.

Los identificadores representan los datos de un programa (constantes, variables, tipos de datos). Un identificador es una secuencia de caracteres que sirve para identificar una posición en la memoria de la computadora, que nos permite accesar a su contenido.

Ejemplo:	**Reglas para formar un Identificador**
Nombre	Debe comenzar con una letra (A a Z, mayúsculas o minúsculas) y no deben contener
Num_hrs	espacios en blanco.
Calif2	Letras, dígitos y caracteres como la subraya (_) están permitidos, los número solo
_ciudad	después del primer carácter.
Z	Se recomienda longitud de identificadores de hasta 8 caracteres.

Tipos de variables

Por su Contenido

Variable Numéricas: Son aquellas en las cuales se almacenan valores numéricos, positivos o negativos, es decir almacenan números del 0 al 9, signos (+ y -) y el punto decimal. Ejemplo:
iva=0.15 pi=3.1416 costo=2500

Variables Lógicas: Son aquellas que solo pueden tener dos valores (cierto o falso) estos representan el resultado de una comparación entre otros datos.

Variables Alfanuméricas: Esta formada por caracteres alfanuméricos (letras, números y caracteres especiales). Ejemplo:
letra='a' apellido='lopez' direccion='Av. Libertad #190'

Por su Uso

Variables de Trabajo: Variables que reciben el resultado de una operación matemática completa y que se usan normalmente dentro de un programa. Ejemplo:
suma=a+b/c

Contadores: Se utilizan para llevar el control del número de ocasiones en que se realiza una operación o se cumple una condición. Con los incrementos generalmente de uno en uno.

Acumuladores: Forma que toma una variable y que sirve para llevar la suma acumulativa de una serie de valores que se van leyendo o calculando progresivamente.

Nombre de variable: identificador que consta de varios caracteres alfanuméricos, de los cuales el primero normalmente es una letra.

Ejemplos de nombres de variables (ó identificadores): A221, notas, NOMBRES, NROCuenta, Pepe, temp, suma, Suma, NotaMaxima, sueldo_base, dias_en_anyo, ConjuntoDatos1, Ganancias95, etc.

Las variables contienen el estado de un programa.
1. Se definen (declaración)
2. Se crean
3. Se cargan con un valor inicial (datos de entrada)
4. Su valor se modifica (programa)
5. Llegan a un valor final (resultados de salida)

Criterios a seguir con las variables:
* Cuantas menos, mejor.
* Cada una con un significado muy claro e inmutable.
* No olvidarse de darles un valor inicial.
* Controlar (y comprobar) que van tomando valores sensatos: regularmente aplicar predicados que deben satisfacer y notificar si alguna se sale de lo previsto.
* Los lenguajes modernos obligan a declarar las variables antes de usarlas, lo que permite chequear su existencia y la coherencia en su uso.

Declaración de variables

En la mayoría de los lenguajes de programación, las variables que van a ser usadas en un programa deben ser declaradas antes de ser usadas.

Los objetivos de la declaración de variables son:

Asociar un tipo de dato y un identificador (o nombre) a la variable para que el compilador pueda verificar la corrección de las operaciones en donde interviene la variable.

Permitir que el compilador sepa cuánto espacio de memoria se necesita para almacenar el valor de la variable, y asignar la dirección de memoria donde este valor se va a almacenar.

Tipos de variables

Variable local: Es aquella que está declarada dentro de un bloque delimitado. Sólo se puede usar dentro del bloque en el que ha sido declarada.

Variable global: Es aquella que está declarada para todo el programa, es decir, fuera de cualquier bloque o función. Retiene su valor durante la ejecución de todo el programa

Algoritmo: Conjunto de pasos para resolver todos los problemas de un cierto tipo y debe ser

- **Finito:** Acabará en un número determinado y razonable de pasos.
- **Definible:** Cada paso se definirá de forma precisa. Ya que el lenguaje humano puede ser ambiguo, los algoritmos se definen con un lenguaje formal.
- **Efectivo:** Una persona será capaz de realizarlo de un modo exacto y sin ayuda de una máquina en un tiempo finito.
- **Entrada/Salida:** Tendrá al menos una salida en relación con las entradas. Éstas podrán ser cero o más, y pertenecer a conjuntos de objetos especificados. (Manuel Santos,2006)

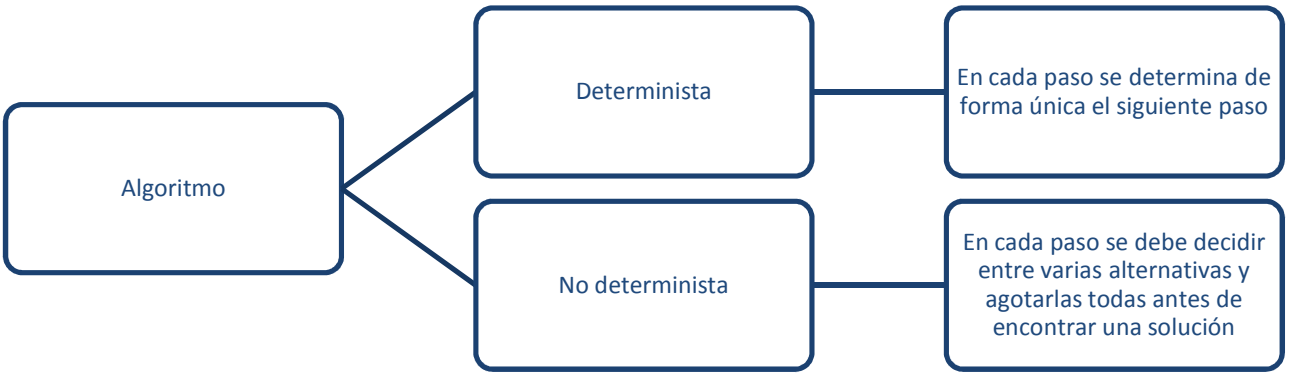


Fig. 1.3 Algoritmo

Algoritmo: Es un procedimiento no ambiguo que resuelve un problema. Un procedimiento es una secuencia de operaciones bien definidas, cada una de las cuales requieren una cantidad finita de memoria y se realiza en un tiempo finito. (Alberto Prieto Espinoza, 2006)

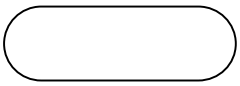
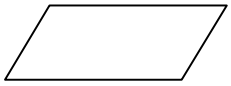
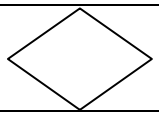
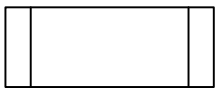
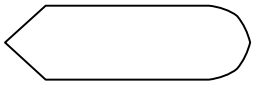
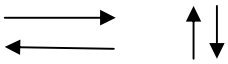
Ejemplo:

Definir el algoritmo necesario para intercambiar los valores de dos variables numéricas.

```
INICIO
   leer(A,B)
   AUX=A
   A=B
   B=AUX
   escribir (A,B)
FIN
```

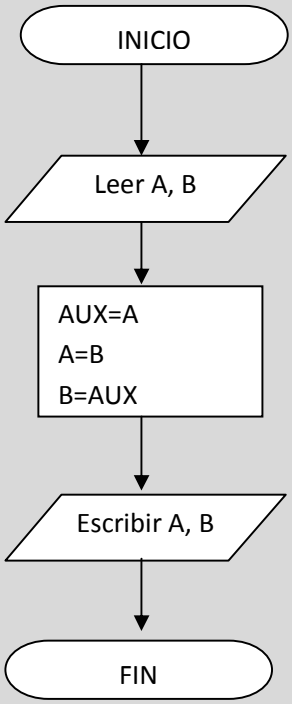
Diagrama de flujo: El diagrama de flujo es una la representación grafica del algoritmo, sobre este se plasman los pasos que se estipulan en el algoritmo.

Simbología (Aguilar, 2003)

	Indica el inicio y el final del diagrama de flujo
	Indica la entrada y salida de datos
	Símbolo de proceso y nos indica la asignación de un valor en la memoria y/o la ejecución de una operación aritmética.
	Símbolo de decisión que indica la realización de una comparación de variables.
	Se utiliza para representar subprogramas (o módulos)
	Conector dentro de la página, representa continuidad del diagrama dentro d ela misma página.
	Conector fuera de página, representa la continuidad del diagrama en otra página.
	Indica la salida de información en la pantalla o monitor.
	Líneas de flujo o dirección. Indican la secuencia en que se realizan las operaciones.

Ejemplo:

Definir el diagrama de flujo necesario para intercambiar los valores de dos variables numéricas.



Operadores

Operadores: Son elementos que relacionan de forma diferente, los valores de una o más variables y / o constantes. Es decir, los operadores nos permiten manipular valores.

La clasificación de operadores se puede dividir en tres grande grupos:

- Aritméticos
- Relaciónales
- Lógicos

Operadores Aritméticos: Los operadores aritméticos permiten la realización de operaciones matemáticas con los valores (variables y constantes). Los operadores aritméticos pueden ser utilizados con tipos de datos enteros o reales. Si ambos son enteros, el resultado es entero; si alguno de ellos es real, el resultado es real.

^ Exponente
+ Suma
- Resta
* Multiplicación
/ División, Mod Modulo (residuo de la división entera)

Prioridad de los Operadores Aritméticos

Todas las expresiones entre paréntesis se evalúan primero. Las expresiones con paréntesis anidados se evalúan de dentro a fuera, el paréntesis más interno se evalúa primero. Dentro de una misma expresión los operadores se evalúan en el siguiente orden.

1.- ^ Exponenciación
2.- *, /, mod Multiplicación, división, modulo.
3.- +, - Suma y resta.

Ejemplos:

Expresión	**Resultado**
7 / 2	3.5
12 mod 7	5
4 + 2 * 5	14

Los operadores en una misma expresión con igual nivel de prioridad se evalúan de izquierda a derecha.

Ejemplos:

4 + 2 * 5 = 14
23 * 2 / 5 = 9.2
46 / 5 = 9.2
3 + 5 * (10 - (2 + 4)) = 23
3 + 5 * (10 - 6) = 3 + 5 * 4 = 3 + 20 = 23
3.5 + 5.09 - 14.0 / 40 = 5.09
3.5 + 5.09 - 3.5 = 8.59 - 3.5 = 5.09
2.1 * (1.5 + 3.0 * 4.1) = 28.98
2.1 * (1.5 + 12.3) = 2.1 * 13.8 = 28.98

Los operadores +, -, *, /, se pueden usar con operandos (constante, variable o expresión) tanto enteros como reales.
La división (div) de enteros da como resultado un entero.

Ejemplo: 17 div 5 = 3

El operador módulo (mod) sólo puede ser utilizado con operandos enteros. La expresión a mod b da como resultado el resto de la división entera de dos números a y b.

Ejemplos:

17 mod 5=2

Los operandos de una expresión aritmética deben satisfacer las siguientes reglas:

*Si ambos operandos son de tipo entero entonces el resultado es de tipo entero.
*Si uno de los operandos o ambos operandos son de tipo real entonces el resultado es de tipo real.
*Si la expresión evaluada da como resultado un valor de tipo entero y la variable a la cual se le va a asignar este valor es de tipo real, entonces el resultado será convertido automáticamente al tipo real antes de realizar la asignación.
*Si la expresión evaluada da como resultado un valor de tipo real y la variable a la cual se le va a asignar este valor es de tipo entero, entonces el resultado será convertido automáticamente al tipo entero antes de realizar la asignación.

Operadores Relaciónales: Los operadores relaciónales (de comparación) comparan dos expresiones y devuelven un valor Boolean que representa el resultado de la comparación. Los operadores se pueden utilizar para comparar valores numéricos, para comparar cadenas y para comparar objetos.

Se utilizan para establecer una relación entre dos valores. Compara estos valores entre si y esta comparación produce un resultado de certeza o falsedad (verdadero o falso).

Los operadores relaciónales comparan valores del mismo tipo (numéricos o cadenas). Tienen el mismo nivel de prioridad en su evaluación. Los operadores relaciónales tiene menor prioridad que los aritméticos.

Operadores relacionales

> Mayor que
< Menor que
> = Mayor o igual que
< = Menor o igual que
< > Diferente
= Igual

Operadores Lógicos: Al igual que las relaciones, en las operaciones con operadores lógicos se tienen resultados cuyo valor de verdad toma uno de los valores boléanos true o false.

Operadores Lógicos: Operan sobre operandos (constantes, variables, expresiones) de tipo lógico (cierto o falso) y evalúan a un valor de tipo lógico (cierto o falso). Estos operadores se utilizan para establecer relaciones entre valores lógicos. Estos valores pueden ser resultado de una expresión relacional.

Operadores Lógicos

And Y
Or O
Not Negación

Operador And

Operando1	Operador	Operando2	Resultado
T	AND	T	T
T		F	F
F		T	F
F		F	F

Operador Or

Operando1	Operador	Operando2	Resultado
T	OR	T	T
T		F	T
F		T	T
F		F	F

Operador Not

Operando	Resultado
T	F
F	T

Prioridad de operadores lógicos

Not
And
Or

Jerarquía de operadores

Prioridad general de los operadores:

1.- ( )
2.- ^
3.- *, /, div, Mod, Not
4.- +, -, And
5.- >, <, > =, < =, < >, =, Or

Expresiones

Son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales.

a+(b + 3)/c

EXPRESIONES ARITMÉTICAS

Conversión de expresiones algebraicas a expresiones aritméticas:

Ejemplos:

5 x 7 se representa como 5*7

6
-- se representa como 6/4
4

3^7 se representa como 3^7 ó también 3**7

Ejercicios:

1. 5(x+y)
2. $a^2 + b^2$
3. a+(b+3)+√c
4. $1x2+5-(2^3)$
5. b x h

   2

Determina el resultado de las siguientes expresiones aritméticas.

1. 3+6*14
2. 8+7*3+4*6
3. -4*7+8/4-5
4. 12+3*7+5*4
5. 4 + 2 * 5
6. 23 * 2 / 5
7. 3 + 5 * (10 - (2 + 4))
8. 3.5 + 5.09 - 14.0 / 40

9. 2.1 * (1.5 + 3.0 * 4.1)
10. 6 – 3 * 4 / 4 + 5 * 2
11. (2 * 3 + (6 div 3 – 1)) ** 2
12. 6 mod 3 - 9 / 4 * 2 + 1

Algunas funciones predefinidas para los algoritmos ó pseudocódigos:

Función	**Descripción**	**Tipo de argumento**	**Resultado**
abs(x)	Valor absoluto de x	Entero o real	Igual que el argumento
arctan(x)	Arcotengente de x	Entero o real	Real
cos(x)	Coseno de x	Entero o real	Real
cuadrado(x)	Cuadrado de x	Entero o real	Igual que el argumento
ent(x)	Entero de x	Real	Entero
exp(x)	E elevado a x	Entero o real	Real
in(x)	Logaritmo neperiano de x	Entero o real	Real
log10(x)	Logaritmo base 10 de x	Entero o real	Real
raíz2(x)	Raíz cuadrada de x	Entero de x	Real
redondeo(x)	Redondea a x al entero más próximo	Real	Entero
sen(x)	Seno de x	Entero o real	Real
trunc(x)	Parte entera de x	Real	Entero

EXPRESIONES LÓGICAS (BOOLEANAS)

Se forman combinando constantes lógicas, variables lógicas y otras expresiones lógicas, utilizando *los operadores lógicos not, and y or* y *los operadores relacionales =,<,>,<=,>=,<>*. El resultado de estas expresiones es falso ó verdadero.

Ejemplos:

a) Si A=4 y B=3

A > B es verdadero

Mientras que

(A-2) < (B-4) es falso

b) (1<5) y (5<10) es verdadera

(5>10) o (‘A’ < ‘B’) es verdadera

Ejercicios:

Si a = 10 b = 20 c = 30

1. a + b > c R=
2. a - b < c R=
3. a - b = c R=
4. a * b < > c R=
5. (a*b) < (a+10) R=

Determina el resultado de las siguientes expresiones lógicas.

a = 10 b = 12 c = 13 d =10

1. a > b)or(a < c)) and ((a = c) or (a > = b))
2. ((a > = b) or (a < d)) and ((a > = d) and (c > d))
3. not (a = c) and (c > b)
4. no (15 > = 7 ** 2) o (43 – 8 * 2 div 4 <> 3 * 2 div 2)
5. (15 > = 7 * 3 ** 2 y 8 > 3 y 15 > 6) o no (7 * 3 < 5 + 12 * 2 div 3 ** 2)
6. no ((7 * 3 div 2 * 4 < 20) o (15 / 2 * 6 < = 15 * 2 / 17))

Dadas las siguientes declaraciones:

TRES = 3

entero a = 5, b = 4

real x = 5e-2, y = 2.

¿Qué valores se obtienen de evaluar las siguientes expresiones?

1) a = b mod TRES

2) 6 div TRES < TRES mod 6

3) TRES + b - 1 <> a o b >= -b * a y a ** 2 <= 10

4) x * y * 10 = 10.E-1

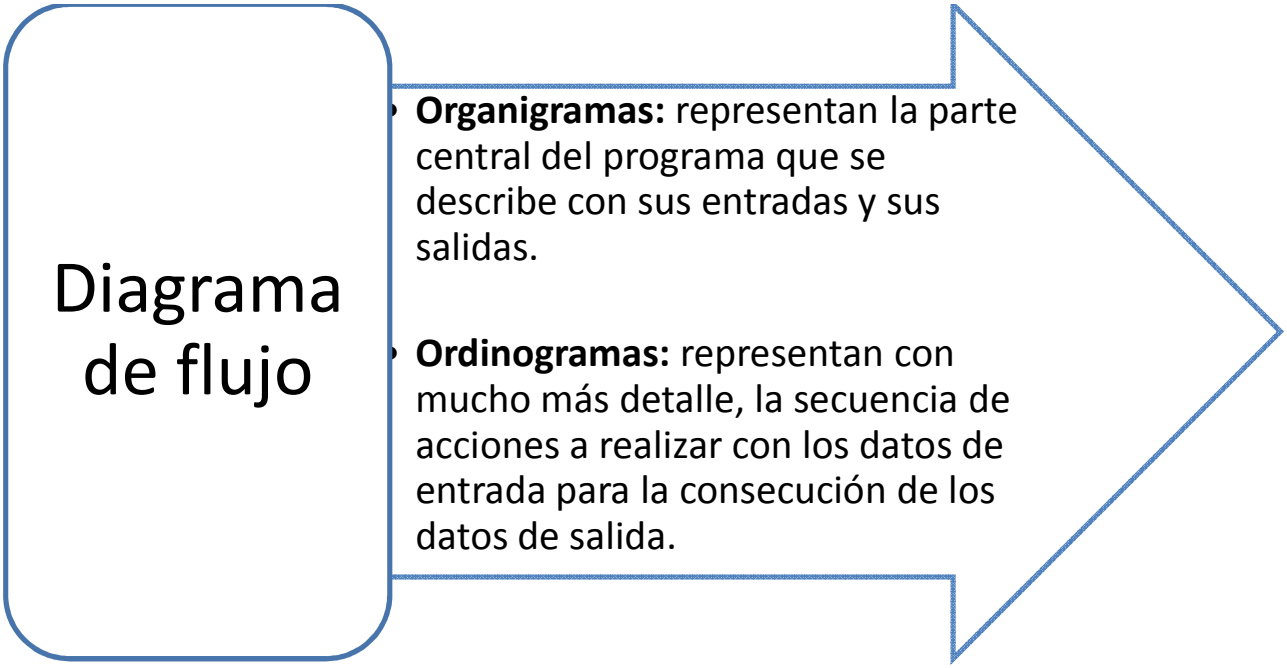
5) b mod a div TRES

6) no (x * a > y / b)

Diagramas de flujo y pseudocódigo

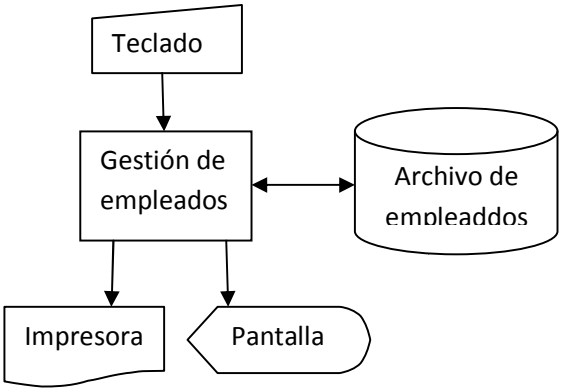
Las dos técnicas utilizadas comúnmente para diseñar algoritmos son el *Diagrama de Flujo* y *Pseudocódigo.*

Diagrama de flujo: Un diagrama de flujo es una técnica descriptiva que está a disposición del analista y el programador para representar visualmente el flujo de datos y la secuencia de ejecución de un programa, es decir, los elementos que intervienen y la secuencia de acciones.



Ejemplos:

ORGANIGRAMA



ORDINOGRAMA (portero)

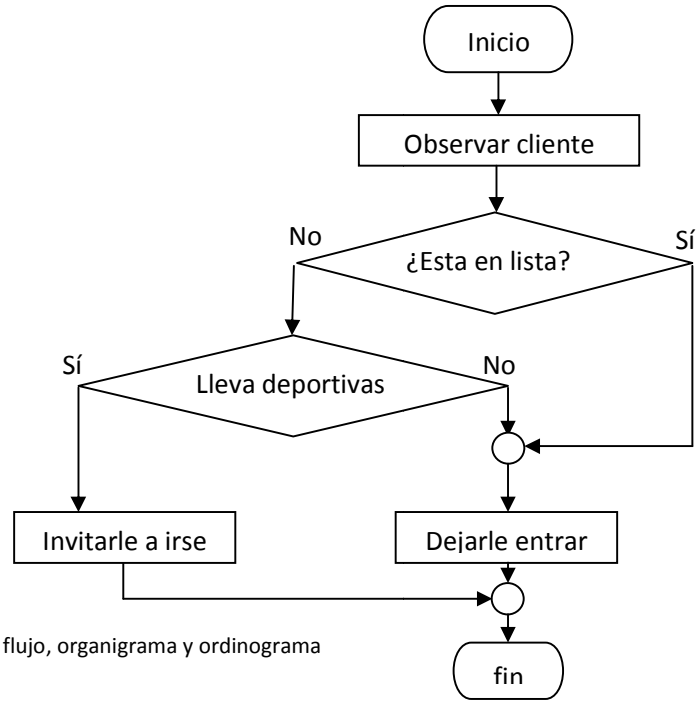
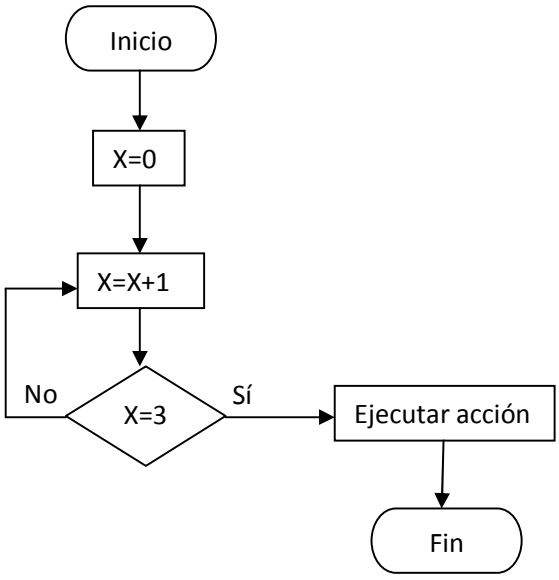


Fig. 1.4 Diagrama de flujo, organigrama y ordinograma

Ejemplo de un ordinograma de una variable de uso: Contador encargado de incrementar un número X cuyo valor inicial es 0 hasta 3. Una vez llegado a este número, se ejecutará una acción cualquiera y finalizará.



Pseudocódigo: Es una herramienta utilizada en programación para resolver problemas usando un lenguaje común, independiente de cualquier lenguaje de programación, se puede considerar como un paso intermedio, casi final, entre la solución de un problema y su codificación en un lenguaje de programación. (Aguilar, 2003)

Ejemplo:

PROGRAMA: Nombre del programa o función
ENTORNO:
Declaración de variables, constantes, tablas, etc. Que se vayan a utilizar para resolver el algoritmo en cuestión.
ALGORITMO:
Sentencia de instrucciones que forman el programa
Instrucción 1
instrucción 2
instrucción 3
….
FIN DEL ALGORITMO
FIN DEL PROGRAMA

Estructura de un algoritmo: La estructura de un algoritmo sirve para organizar a los elementos que aparecen en él. En pseudocódigo, todos los algoritmos tienen la misma estructura, la cual viene definida por tres secciones: (Cabrera Filomeno, 2008)

- Cabecera
- Declaraciones
- Cuerpo

La sintaxis completa para escribir un algoritmo en pseudocódigo CEE[1] es:

algoritmo <nombre_del_algoritmo>

[constantes <declaraciones_de_constantes> **]**

[tipos_de_datos <declaraciones_de_tipos_de_datos> **] [variables** <declaraciones_de_variables> **]**

inicio

<bloque_de_instrucciones>

fin

Ejemplo:

/*CABECERA*/

algoritmo Area_de_una_circunferencia

/*DECLARACIONES*/

constantes

PI=3.141592

variables

real area, radio

/*CUERPO*/

inicio

escribir("Introduzca radio: "**)**

leer (radio**)**

area←PI*radio**2

escribir ("El área de la circunferencia es: ", area**)**

fin

[1] Esta nomenclatura (CEE – C En Español-) esta definida por Carlos Pes en su página Web: http://carlospes.blogspot.com/2007/10/qu-es-el-pseudocdigo-cee.html

Clasificación de las instrucciones.

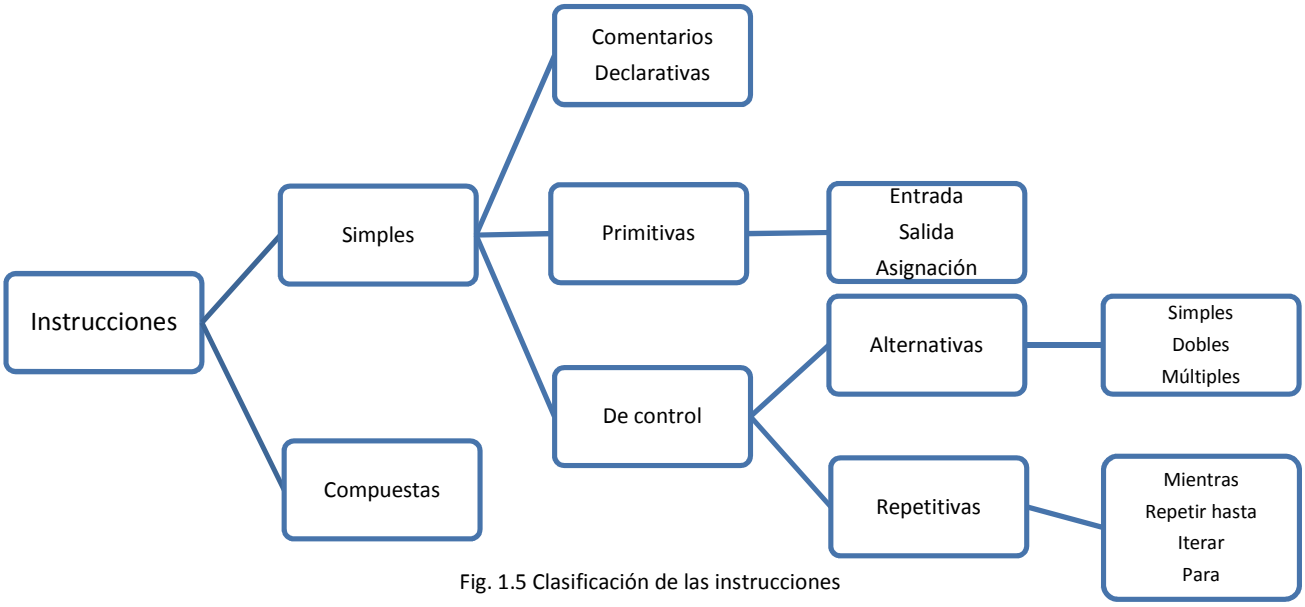


Fig. 1.5 Clasificación de las instrucciones

Cometarios: Son líneas de texto que el compilador ignora pero que son de mucha utilidad para el programador.

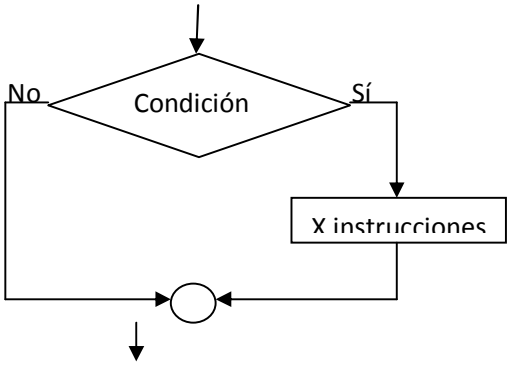
Declarativas: Instrucciones que se utilizan para declarar objetos que se van a usar en el programa (constantes, variables, etc.)

Primitivas: Instrucciones que se ejecutan de forma inmediata por el procesador.

- **Entrada:** se toman los datos de entrada y se almacenan en variables.
- **Salida:** Se usan para representar un medio externo (pantalla, impresora) mensaje para el usuario, contenido de variables, el resultado de una expresión.
- **Asignación:** este tipo de instrucción se usa para asignar valores a las variables. (= ó ←)

De control: Son instrucciones que permiten controlar la ejecución de otras instrucciones bajo ciertas circunstancias, Evalúan una expresión lógica y, dependiendo de si se cumple o no, fuerzan a que se ejecuta parte del programa. La condición no tiene por qué ser una simple, sino que puede haber más de una unidas por operadores lógicos.

- **Alternativas:** Permiten la ejecución de una serie de instrucciones u otras dependiendo si se cumple la condición o no.
    - **Simples:** Se usan cuando un programa quiere que, cuando se cumpla la condición, se ejecuten una serie de instrucciones determinadas. En caso de que esa condición no se cumpla, el programa seguirá su curso normal.



SI condición **ENTONCES**

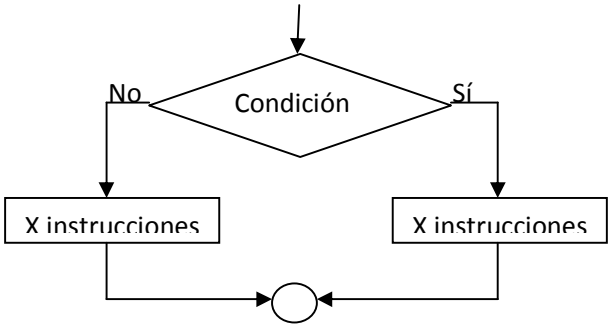
Instrucción 1

Instrucción 2

…

FIN SI

- **Dobles:** Sucede igual que la simple, solo que ahora, en caso de que la condición no se cumpla, se ejecutará otro determinado bloque de instrucciones.



SI condición **ENTONCES**

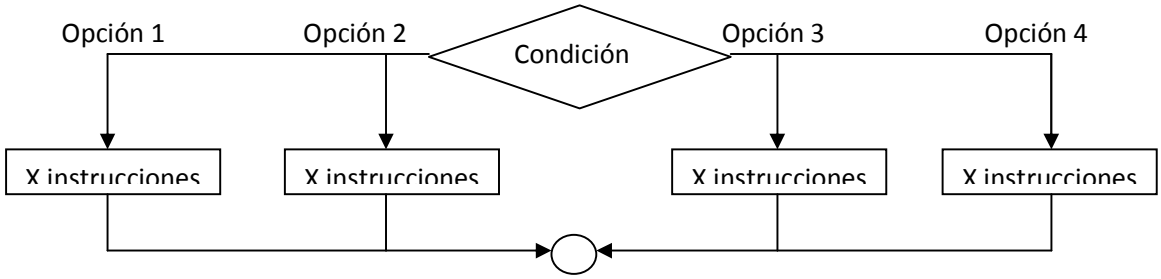
Instrucción 1 …

SI NO

Instrucción 2 …

FIN SI

- **Múltiples:** Estas instrucciones se usan en casos donde la ejecución de un grupo de instrucciones u otro depende de valores concretos y conocidos de una expresión viable.



CASE condición

Condición=opción 1

Instrucciones

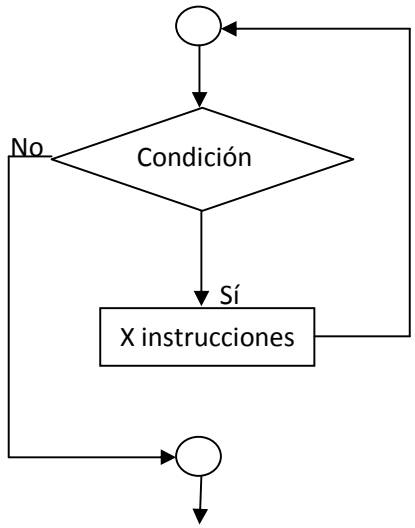
Condición=opción 1

Instrucciones

FIN CASE

- **Repetitivas:** Este grupo de instrucciones engloba aquellas que provocan que un bloque de instrucciones se repita un cierto número de veces. Normalmente se componen de una o varias instrucciones que se repetirán y una condición que determina el número de repeticiones. Es necesario que la condición se cumpla en algún momento, ya que, de lo contrario, las instrucciones se repetirán de forma indefinida, formándose así un bucle "infinito."

- **Mientras:** En este caso la condición se evalúa antes de entrar en el bucle (grupo de instrucciones que se repiten) y determina si se entra o no en el mismo. Así, si la condición no se cumple al principio, no se ejecutaría ninguna vez este conjunto de instrucciones.

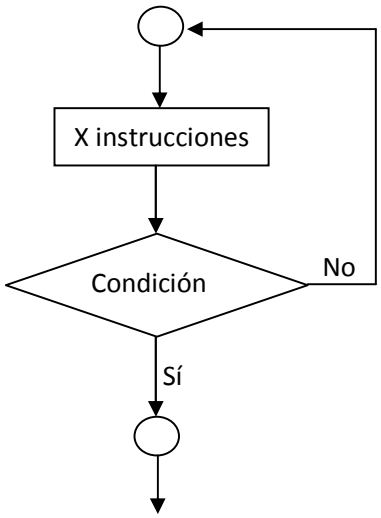


MIENTRAS Condición **HACER**

Instrucción 1 …

FIN MIENTRAS

- **Repetir… hasta:** En esta instrucción, la condición se encuentra al final del bucle y determina si se sale del mismo o no. El hecho de que la condición esté a la salida y no a la entrada, obliga a que al menos, se pase una vez por el bucle hasta llegar a la condición de salida.

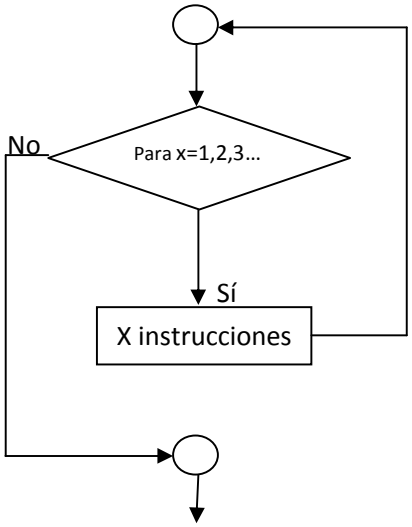


REPETIR

Instrucción 1 …

HASTA Condición

- **Para:** Esta instrucción se usa cuando se conoce el número de veces que ha de ejecutarse el bucle. La condición está al principio y contiene un contador encargado de llevar la cuenta del número de repeticiones.

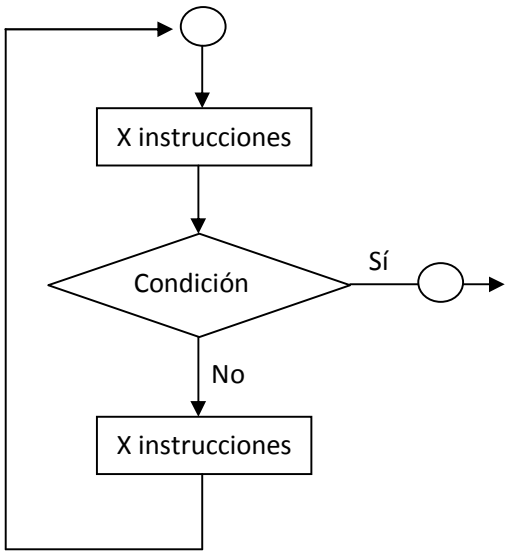


PARA Condición=1,2,3…. **HACER**

Instrucción 1 …

FIN PARA

- **Iterar:** En esta instrucción la condición de evaluación se puede encontrar en cualquier lugar del bucle. La condición determina si se debe salir del bucle o no.



ITERAR

Instrucción 1

…..

SI Condición **SALIR**

Instrucción 2

…..

FIN ITERAR

COMPUESTAS.

Estas instrucciones están compuestas por un conjunto de instrucciones simples agrupadas bajo un nombre y que se desarrollan en un lugar apartado del resto del código. Pueden ser llamadas desde cualquier lugar del programa invocando su nombre. En ese momento, se ejecutará esta instrucción compuesta y, cuando finalice su ejecución, el control del programa volverá a la instrucción siguiente a la que realizó dicha llamada.

Ejercicios resueltos

Escribir un algoritmo que lea un valor entero, lo doble, se multiplique por 25 y visualice el resultado.

/*CABECERA*/

Algoritmo: valor_entero_multiplicado

/*DECLARACIONES*/

Variables

entero numero, resultado

/*CUERPO*/

Inicio

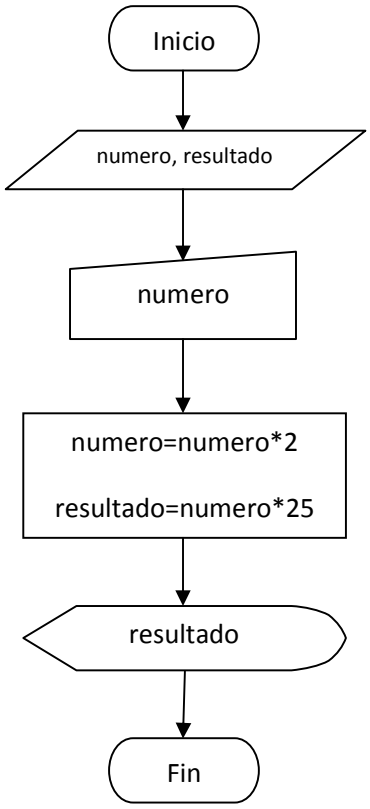
leer (numero**)**

numero ← numero*2

resultado ← numero*25

escribir (resultado**)**

Fin



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.

Algoritmo: capital_intereses

Variables

real cap_inv, gan

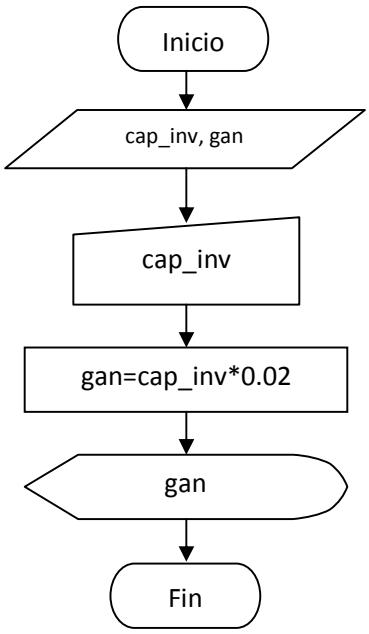
Inicio

leer (cap_inv)

gan = cap_inv * 0.02

escribir(gan**)**

Fin



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.

Algoritmo: descuento

Variables

real tc, d, tp

Inicio

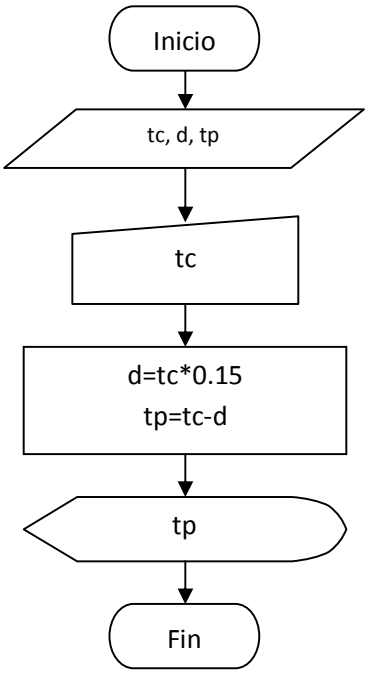
leer (tc)

d=tc*0.15

tp=tc-d

escribir(tp **)**

Fin



Un alumno desea saber cual será su calificación final en la materia de Algoritmos. 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.

Algoritmo: calif_final

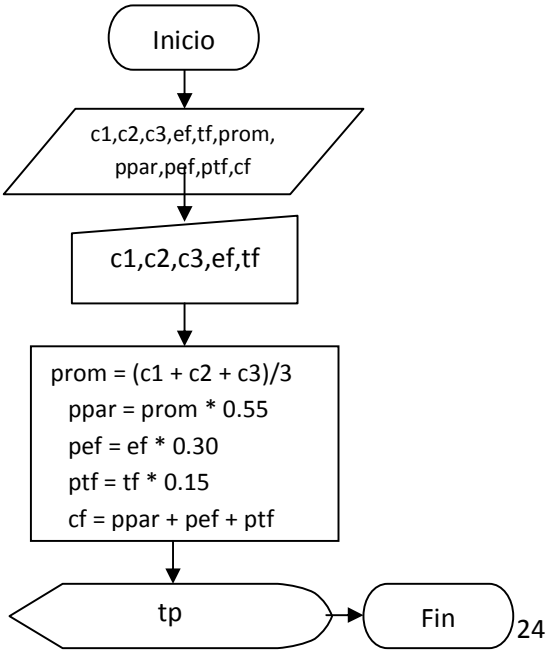
Variables

real c1, c2, c3, ef, tf, prom, ppar, pef, ptf, cf

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
escribir (cf)

Fin



Un hombre desea saber cuanto dinero se genera por concepto de intereses sobre la cantidad que tiene en inversión en el banco. El decidirá reinvertir los intereses siempre y cuando estos excedan a $7000, y en ese caso desea saber cuanto dinero tendrá finalmente en su cuenta.

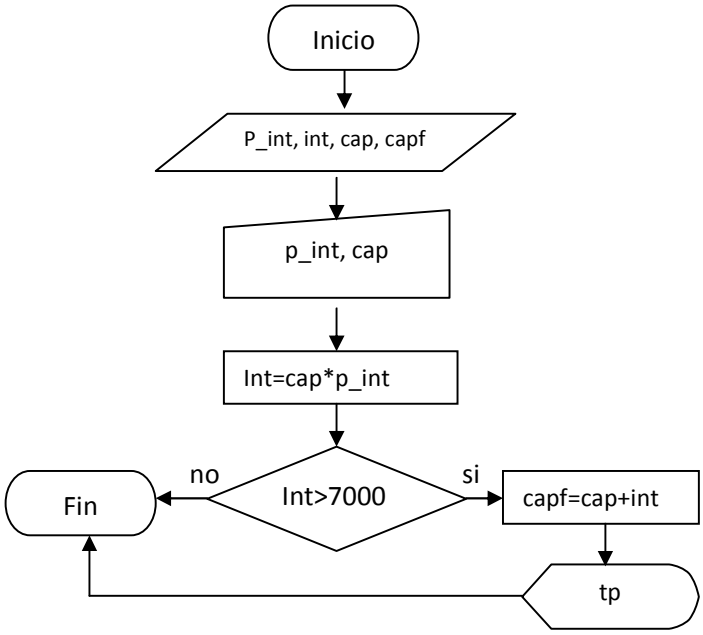
Algoritmo: intereses

Variables

real p_int, int, cap, capfv

Inicio
leer (p_int, cap)
int = cap*p_int
si int>7000 **entonces**
capf=cap+int
fin-si
escribir (capf)

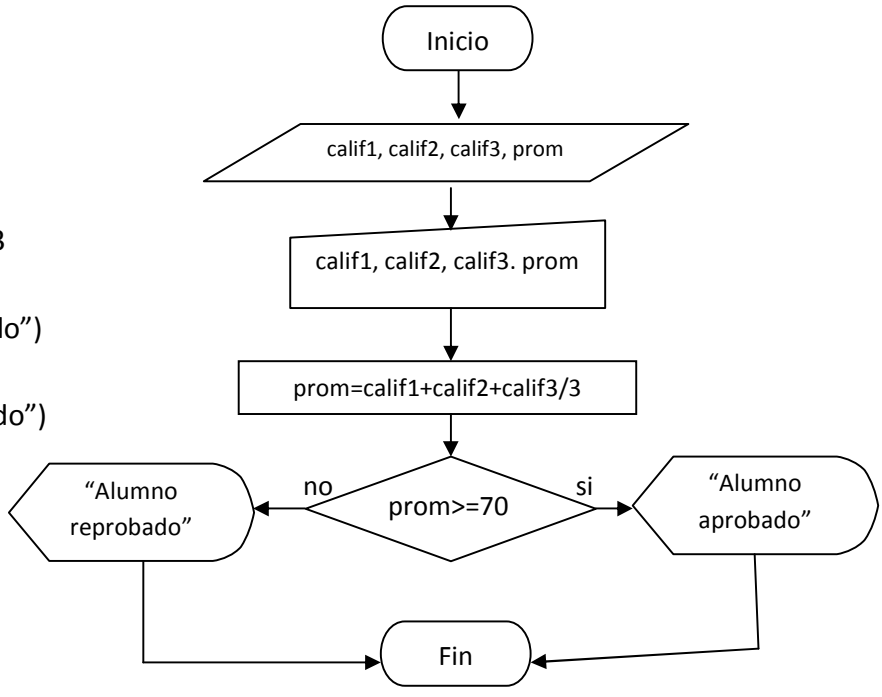
Fin



Determinar si un alumno aprueba a reprueba un curso, sabiendo que aprobara si su promedio de tres calificaciones es mayor o igual a 70; reprueba en caso contrario.

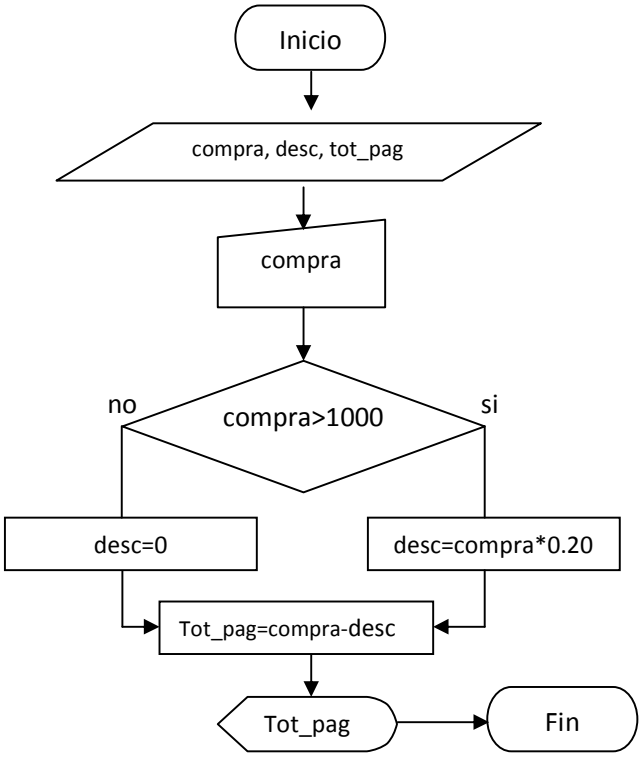
Algoritmo: status_calificacion
Variables
entero calif1, calif2, calif3, prom
Inicio
leer (calif1, calif2, calif3)
prom = (calif1 + calif2 + calif3)/3
Si prom >= 70 **entonces**
escribir (“alumno aprobado”)
si no
escribir(“alumno reprobado”)
fin-si

Fin



En un almacén se hace un 20% de descuento a los clientes cuya compra supere los $1000 ¿Cual será la cantidad que pagara una persona por su compra?

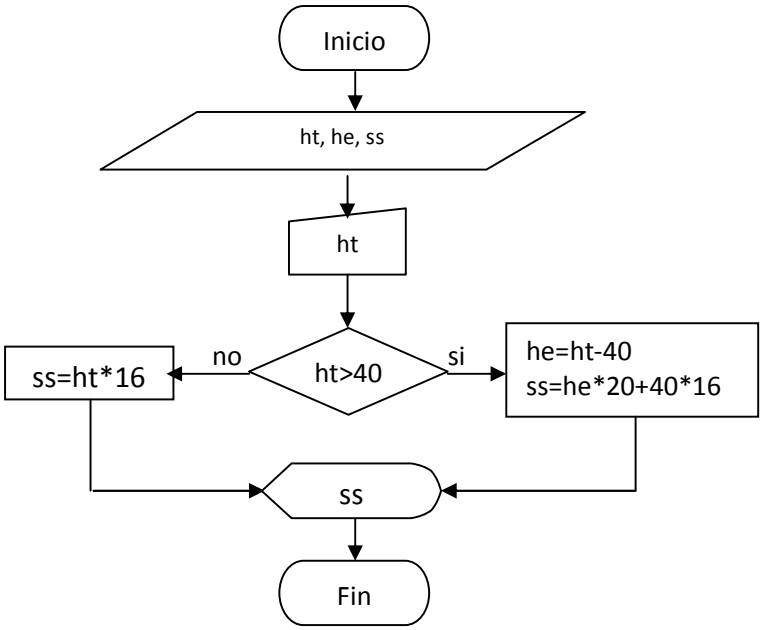
Algoritmo: descuento_20
Variables
real compra, desc, tot_pag
Inicio
leer (compra)
Si compra > 1000 **entonces**
desc = compra * 0.20
si no
desc = 0
fin-si
tot_pag = compra - desc
escribir (tot_pag)
Fin.



Un obrero necesita calcular su salario semanal, el cual se obtiene de la sig. manera:

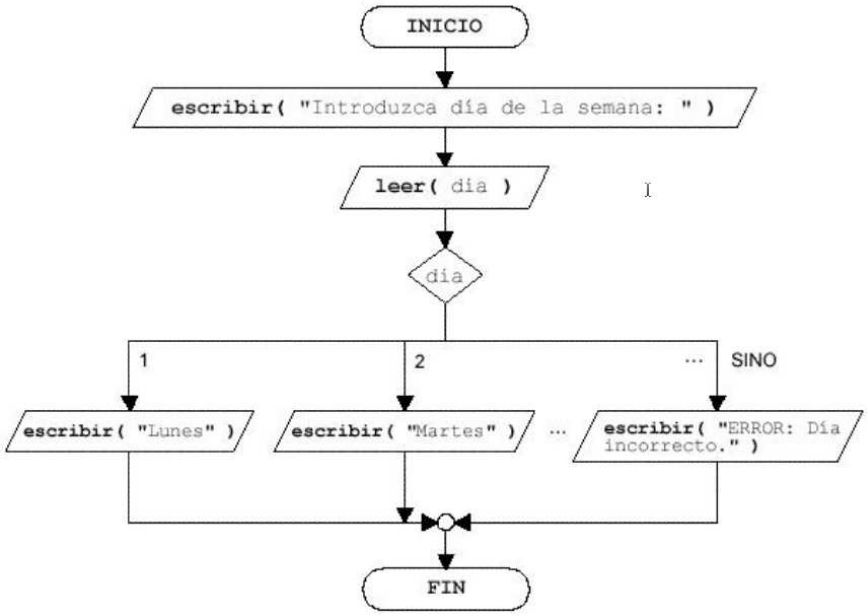
- Si trabaja 40 horas o menos se le paga $16 por hora
- Si trabaja mas de 40 horas se le paga $16 por cada una de las primeras 40 horas y $20 por cada hora extra.

Algoritmo: salario
Variables
entero ht, he
real ss
Inicio
leer (ht)
Si ht > 40 **entonces**
he = ht - 40
ss = he * 20 + 40 * 16
si no
ss = ht * 16
fin-si
escribir (ss)
Fin



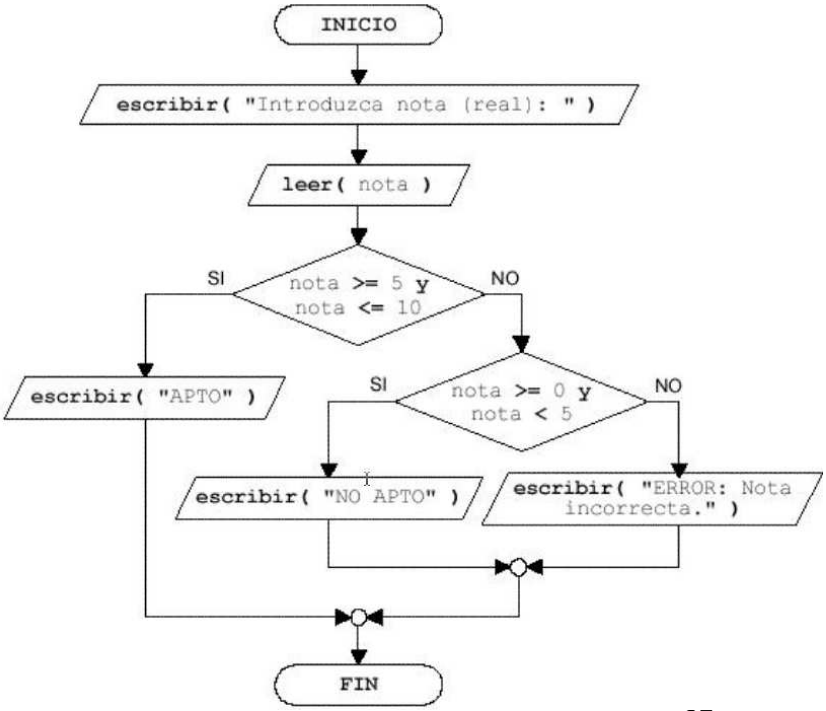
Diseñar un algoritmo que solicite por teclado un número de un día de la semana y que muestre en pantalla el nombre correspondiente a dicho día.

Algoritmo: dia_de_la_semana
Variables
entero dia
Inicio
escribir (“Introduzca el día de la semana:”)
leer (dia)
según_sea (dia**)**
1: escribir (“Lunes”)
2: escribir (“Martes”)
3: escribir (“Miércoles”)
4: escribir (“Jueves”)
5: escribir (“Viernes”)
6: escribir (“Sábado”)
7: escribir (“Domingo”)
sino: escribir (“ERROR: día incorrecto”)
fin_segun_sea
Fin



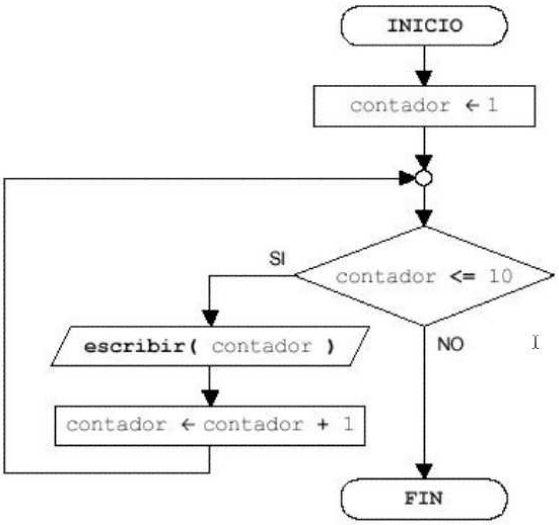
Diseñar un algoritmo que solicite por teclado la calificación de una asignatura y que muestre por pantalla:
“APTO”, en el caso de que la nota sea mayor o igual que 5 y menor o igual a 10
“NO APTO”, en el caso de que la nota sea mayor o igual que 0 y menor que 5
“ERROR: Nota incorrecta”, en caso de que la nota sea menor que 0 o mayor que 10

Algoritmo: calificación_segun_nota
Variables
real nota
Inicio
escribir (“Introduzca nota:”)
leer (nota)
si (nota>=5 y nota <=10**)**
escribir (“APTO”)
sino
si (nota>=0 y nota <5**)**
escribir (“NO APTO”)
sino
escribir (“ERROR: Nota incorrecta”)
fin si
fin si
Fin

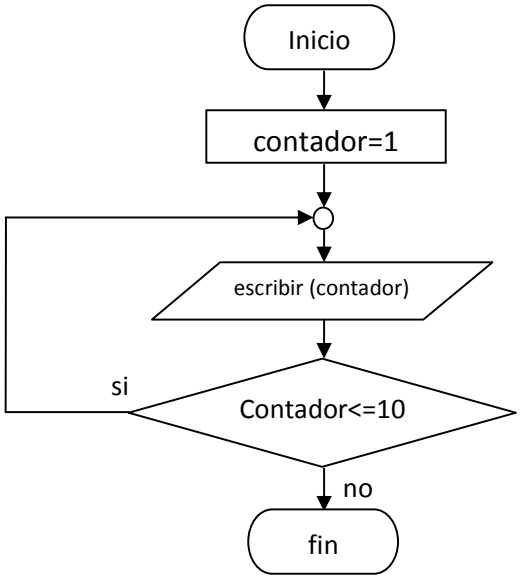


Diseñar un algoritmo que muestre en pantalla los primeros 10 números naturale.

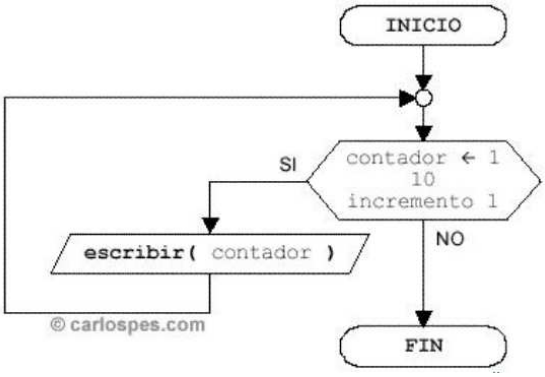
```
//estructura mientras
Algoritmo: números_del_1_al_10
Variables
    entero  contador
Inicio
    contador=1
    mientras (contador<=10)
          escribir (contador)
          contador=contador+1
    fin mientras
Fin
```




```
//estructura hacer mientras
Algoritmo: números_del_1_al_10
Variables
    entero  contador
Inicio
    Contador=1
    hacer
          escribir (contador)
          contador=contador+1
    mientras (contador<=10)
Fin
```




```
//estructura para
Algoritmo: números_del_1_al_10
Variables
    entero  contador
Inicio
     Para contador=1 hasta 10 hacer
          escribir (contador)
     fin para
Fin
```



Arreglos

Un *arreglo* es un conjunto finito y ordenado de elementos homogéneos. La propiedad ordenado significa que el primer elemento, el segundo, el tercero,..., el enésimo puede ser identificado. La homogeneidad implica que todos los elementos del arreglo son datos del mismo tipo.

Un *vector* o *arreglo unidimensional* consta de n elementos que pueden representarse como:

A [1]	A[2]	A[3]	...	A[i]	...	A[n]

Nótese que todos los elementos comparten el mismo identificador (A en este caso) mientras que el subíndice identifica a cada elemento individual.

Al número de elementos de un vector se le denomina ***rango*** del vector.

Los vectores se almacenan en la memoria central del computador en un orden adyacente de posiciones de memoria sucesivas. Cada elemento de un vector se puede procesar como si fuera una variable simple que ocupa una posición de memoria dada, de manera tal que cada elemento del vector es accesible directamente.

Por ejemplo: el vector $X_{[6]}$ está compuesto por seis elementos. Su identificador (nombre del vector) es X y cada elemento se individualiza por su subíndice. (Introducción a la Programación)

$X_{(1)}$	$X_{(2)}$	$X_{(3)}$	$X_{(4)}$	$X_{(5)}$	$X_{(6)}$
30	33	28	30	31	29

En Basic, Java y otros lenguajes es posible declarar matrices multidimensionales, entendiéndolas como un vector de vectores. En dichos casos en número de elementos del vector es el producto resultante de cada dimensión.

Por ejemplo el vector v(4,1) tiene 10 elementos se calcula del siguiente modo: (0-4) * (0-1). Los elementos de la primera dimensión del vector contiene 5 elementos que van del '0' al '4' y la 2º dimensión tiene 2 elementos que van desde '0' a '1'. Los elementos serían accedidos del siguiente modo:

elemento 1: (0,0)
elemento 2: (0,1)
elemento 3: (1,0)
...
elemento 8: (3,1)
elemento 9: (4,0)
elemento 10: (4,1)

Problemas propuestos

1. Dada un cantidad en pesos, obtener la equivalencia en dólares, asumiendo que la unidad cambiaría es un dato desconocido.

2. Leer un numero y escribir el valor absoluto del mismo

3. La presión, el volumen y la temperatura de una masa de aire se relacionan por la formula:

   masa = (presión * volumen)/(0.37 * (temperatura + 460))

4. Calcular el numero de pulsaciones que una persona debe tener por cada 10 segundos de ejercicio, si la formula es:

   num. pulsaciones = (220 - edad)/10

5. Calcular el nuevo salario de un obrero si obtuvo un incremento del 25% sobre su salario anterior.

6. En un hospital existen tres áreas: Ginecología, Pediatría, Traumatologia. El presupuesto anual del hospital se reparte conforme a la sig. tabla:

Área	Porcentaje del presupuesto
Ginecología	40%
Traumatologia	30%
Pediatría	30%

Obtener la cantidad de dinero que recibirá cada área, para cualquier monto presupuestal.

7. El dueño de una tienda compra un articulo a un precio determinado. Obtener el precio en que lo debe vender para obtener una ganancia del 30%.

8. Todos los lunes, miércoles y viernes, una persona corre la misma ruta y cronometra los tiempos obtenidos. Determinar el tiempo promedio que la persona tarda en recorrer la ruta en una semana cualquiera.

9. Tres personas deciden invertir su dinero para fundar una empresa. Cada una de ellas invierte una cantidad distinta. Obtener el porcentaje que cada quien invierte con respecto a la cantidad total invertida.

10. Un alumno desea saber cual será su promedio general en las tres materias mas difíciles que cursa y cual será el promedio que obtendrá en cada una de ellas. Estas materias se evalúan como se muestra a continuación:

    La calificación de Matemáticas se obtiene de la sig. manera:
    Examen 90%
    Promedio de tareas 10%
    En esta materia se pidió un total de tres tareas.

    La calificación de Física se obtiene de la sig. manera:
    Examen 80%
    Promedio de tareas 20%
    En esta materia se pidió un total de dos tareas.

    La calificación de Química se obtiene de la sig. manera:
    Examen 85%
    Promedio de tareas 15%
    En esta materia se pidió un promedio de tres tareas.

11. Un proveedor de estéreos ofrece un descuento del 10% sobre el precio sin IVA, de algún aparato si este cuesta $2000 o mas. Además, independientemente de esto, ofrece un 5% de descuento si la marca es “NOSY”. Determinar cuanto pagara, con IVA incluido, un cliente cualquiera por la compra de su aparato.

12. Una persona debe realizar un muestreo con 50 personas para determinar el promedio de peso de los niños, jóvenes, adultos y viejos que existen en su zona habitacional. Se determinan las categorías con base en la sig, tabla:

CATEGORIA	EDAD
Niños	0 - 12
Jóvenes	13 - 29
Adultos	30 - 59
Viejos	60 en adelante

13. Al cerrar un expendio de naranjas, 15 clientes que aun no han pagado recibirán un 15% de descuento si compran mas de 10 kilos. Determinar cuanto pagara cada cliente y cuanto percibirá la tienda por esas compras.

14. En una empresa se requiere calcular el salario semanal de cada uno de los *n* obreros que laboran en ella. El salario se obtiene de la sig. Forma:
    - Si el obrero trabaja 40 horas o menos se le paga $20 por hora
    - Si trabaja más de 40 horas se le paga $20 por cada una de las primeras 40 horas y $25 por cada hora extra.

15. En un supermercado un cajero captura los precios de los artículos que los clientes compran e indica a cada cliente cual es el monto de lo que deben pagar. Al final del día le indica a su supervisor cuanto fue lo que cobro en total a todos los clientes que pasaron por su caja.

16. En un supermercado un ama de casa pone en su carrito los artículos que va tomando de los estantes. La señora quiere asegurarse de que el cajero le cobre bien lo que ella ha comprado, por lo que cada vez que toma un articulo anota su precio junto con la cantidad de artículos iguales que ha tomado y determina cuanto dinero gastara en ese articulo; a esto le suma lo que ira gastando en los demás artículos, hasta que decide que ya tomo todo lo que necesitaba. Ayúdale a esta señora a obtener el total de sus compras.

17. En la Cámara de Diputados se levanta una encuesta con todos los integrantes con el fin de determinar que porcentaje de los *n* diputados esta a favor del Tratado de Libre Comercio, que porcentaje esta en contra y que porcentaje se abstiene de opinar.

18. Leer 50 calificaciones de un grupo de alumnos. Calcule y escriba el porcentaje de reprobados. Tomando en cuenta que la calificación mínima aprobatoria es de 70.

19. Suponga que tiene usted una tienda y desea registrar las ventas en su computadora. Diseñe un algoritmo que lea por cada cliente, el monto total de su compra. Al final del día que escriba la cantidad total de ventas y el numero de clientes atendidos.

20. Diseñar un pseudocódigo que convierta un número del sistema decimal a :

    a) sistema binario b)sistema octal c)sistema hexadecimal. Según se elija.

21. Calcular el promedio de 50 valores almacenados en un vector. Determinar además cuantos son mayores que el promedio, imprimir el promedio, el número de datos mayores que el promedio y una lista de valores mayores que el promedio.

22. Llenar dos vectores A y B de 45 elementos cada uno, sumar el elemento uno del vector A con el elemento uno del vector B y así sucesivamente hasta 45, almacenar el resultado en un vector C, e imprimir el vector resultante.

23. Llenar un vector de 20 elementos, imprimir la posición y el valor del elemento mayor almacenado en el vector. Suponga que todos los elementos del vector son diferentes.

24. Almacenar 500 números en un vector, elevar al cuadrado cada valor almacenado en el vector, almacenar el resultado en otro vector. Imprimir el vector original y el vector resultante.

25. Almacenar 300 números en un vector, imprimir cuantos son ceros, cuantos son negativos, cuantos positivos. Imprimir además la suma de los negativos y la suma de los positivos.

26. Almacenar 150 números en un vector, almacenarlos en otro vector en orden inverso al vector original e imprimir el vector resultante.

27. Se tienen almacenados en la memoria dos vectores M y N de cien elementos cada uno. Hacer un algoritmo que escriba la palabra "Iguales" si ambos vectores son iguales y "Diferentes" si no lo son.
    Serán iguales cuando en la misma posición de ambos vectores se tenga el mismo valor para todos los elementos.

28. Se tiene el vector A con 100 elementos almacenados. Diseñe un algoritmo que escriba "SI" si el vector esta ordenado ascendentemente o "NO" si el vector no esta ordenado

29. Diseñe un algoritmo que lea un número cualquiera y lo busque en el vector X, el cual tiene almacenados 80 elementos. Escribir la posición donde se encuentra almacenado el número en el vector o el mensaje "NO" si no lo encuentra. Búsqueda secuencial.

30. Diseñe un algoritmo que lea dos vectores A y B de 20 elementos cada uno y multiplique el primer elemento de A con el ultimo elemento de B y luego el segundo elemento de A por el diecinueveavo elemento de B y así sucesivamente hasta llegar al veinteavo elemento de A por el primer elemento de B. El resultado de la multiplicación almacenarlo en un vector C.

Bibliografía

Aguilar, L. J. (2003). *Fundamentos de Programación.* Aravaca (Madrid): McGrawHill.

Alberto Prieto Espinoza, A. L. (2006). *Introducción a la Informática.* Madrid: McGraw Hill.

Cabrera, J. (s.f.). *Unidad 1. Conceptos básicos.* Recuperado el 08 de septiembre de 2008, de http://10.100.96.4/BibliotecaVirtual/POLILIBROS/TIC/LogicaProgramacion/Contenido/Unidad1.html#5

Farrell, J. (2001). *Introducción a la Programación.* México: Internacional Thompson Editores.

Iranzo, P. J. (2005). *Lógica Simbólica.* México, D.F.: Alfaomega Ra-Ma.

Manuel Santos, I. P. (2006). *Fundamentos de programación.* México, D.F.: Alfaomega Ra-Ma.