Universidad Nacional de la Matanza

Elementos de Programación

UNIDAD 4. ESTRUCTURA DE SELECCIÓN SIMPLE / MULTIPLE

INDICE

UNIDAD 4
Estructura selección simple / Múltiple

OBJETIVOS: Conocer en detalle y confeccionar programas utilizando las estructuras de selección simple y múltiple. Elaborar juegos de prueba que abarquen las distintas variantes de los programas. Conocer y utilizar los operadores lógicos y de comparación.

1 Estructura de Selección Simple (Sentencia if)

Esta estructura permite realizar la selección entre dos posibles cursos de acción, en base a la verdad o falsedad de una expresión que se escribe en forma de "expresión lógica".

Para escribir una expresión lógica es necesario utilizar “operadores de comparación”. Estos operadores permiten evaluar una condición entre dos partes, ya sean dos variables, o una variable y una constante. Los operadores de comparación son los que se muestran en la tabla 1.

Tabla 1: Operadores de comparación

Operador	Símbolo
Mayor	>
Menor	<
Mayor o igual	>=
Menor o igual	<=
Igual	==
Distinto	!=

La representación gráfica de la estructura de selección simple puede verse en la Figura 1.

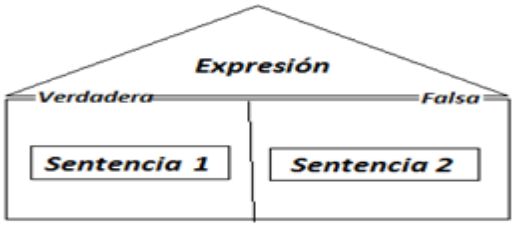


Figura 1: Representación gráfica de la estructura de selección simple

Por ejemplo, dada una variable A numérica, se puede escribir como expresión A > 6 en cuyo caso será verdadera ó falsa según el resultado de la evaluación del contenido ingresado en A. Si es verdadera, se ejecuta la “Sentencia 1”, y si es falsa se ejecuta la “Sentencia 2”. Dentro del verdadero y/o falso puede haber una o más sentencias.

El lenguaje C admite también que se utilice una expresión aritmética, tomando el resultado de la evaluación como “falso” si es igual a cero y verdadero por distinto de cero.

Ejemplo: si se evalúa 5-6+3 esta expresión devuelve un valor distinto de cero, por lo cual, continua la secuencia por verdadero. En cambio, al evaluar 4+3-7 esta expresión devuelve como resultado un valor igual a cero, por lo que la secuencia continua por el lado falso.

Se puede utilizar, si es que lo desea, solo la parte verdadera, o sea que las acciones por 'falso' son opcionales. Las expresiones que se utilizarán al principio serán simples, es decir, una relación entre dos valores y luego se ampliarán utilizando los operadores lógicos.

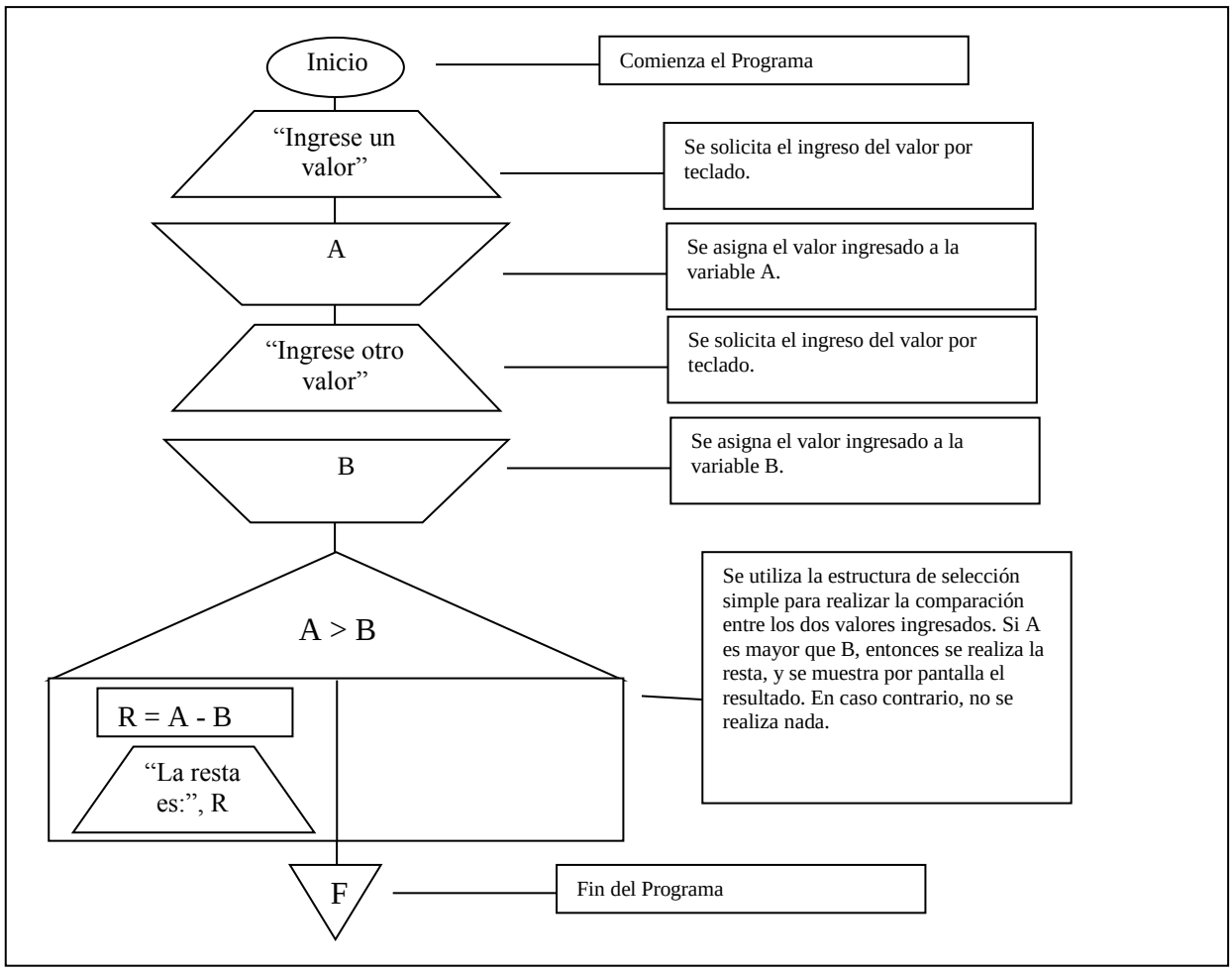
Estas estructuras se pueden enlazar en forma 'descendente', es decir, que dentro del verdadero y/o falso puede haber otras estructuras condicionales. Esto se verá en otros ejercicios más

adelante. Esta estructura tiene un solo punto de entrada y uno solo de salida cualquiera fuese la alternativa ejecutada.

Ejemplo 1: Ingresar dos valores reales en dos variables, que se llamaran A y B. Si A es mayor a B calcular e informar la diferencia entre A menos B.

ESTRATEGIA
1. Ingresar los dos valores en A y B
2. Si A es mayor que B calcular la diferencia
3. Informar la diferencia calculada si se cumplió la condición del punto 2.

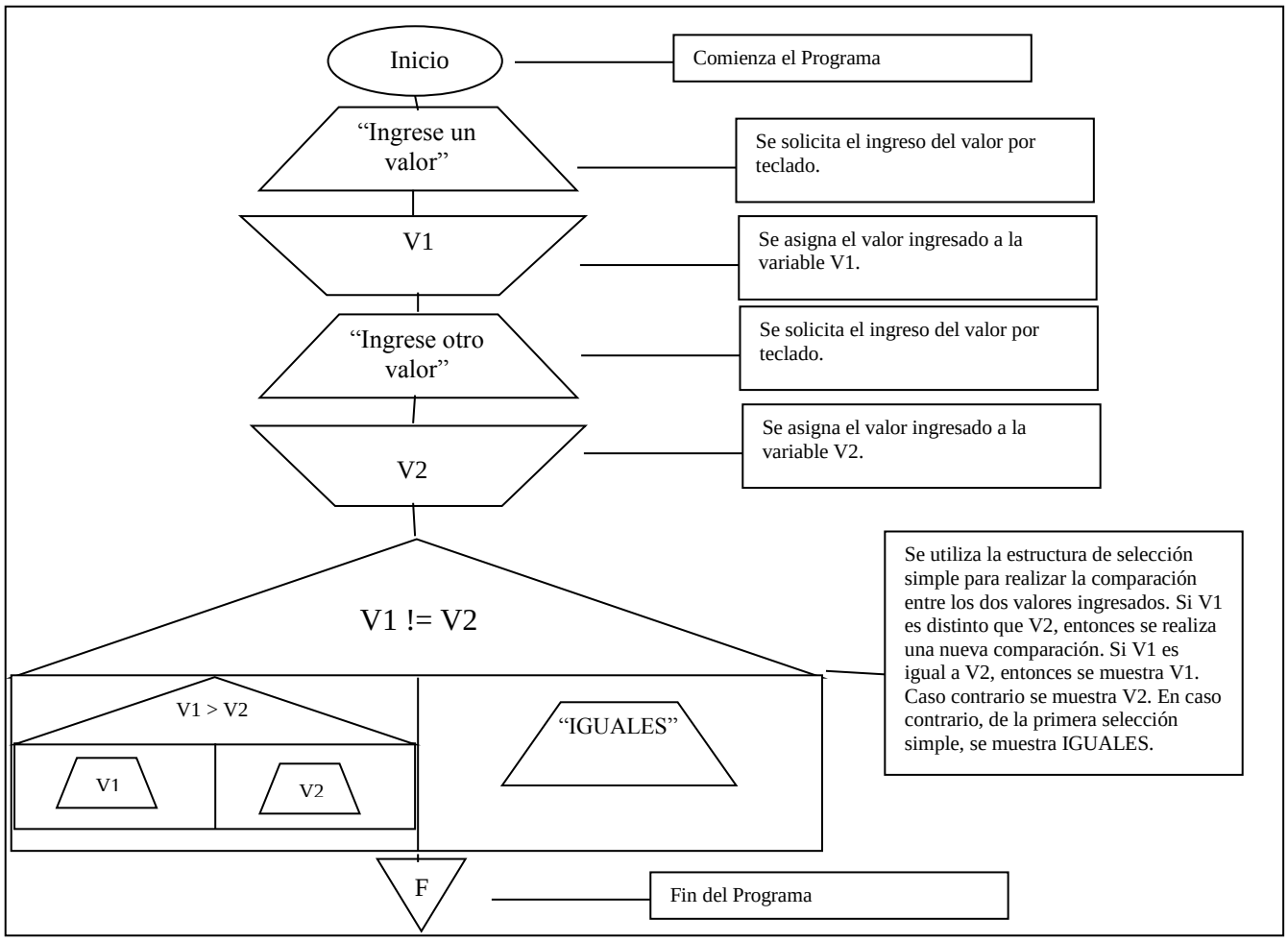
DIAGRAMA:



Ejemplo 2: Confeccionar un programa que ingrese dos valores numéricos y determine e informe al mayor de ellos si son distintos, o un mensaje que diga IGUALES en caso de serlo.

ESTRATEGIA
1] Ingresar los dos valores
2] Si no son iguales, determino el mayor
3] Informo el mayor ó el mensaje IGUALES

DIAGRAMA:

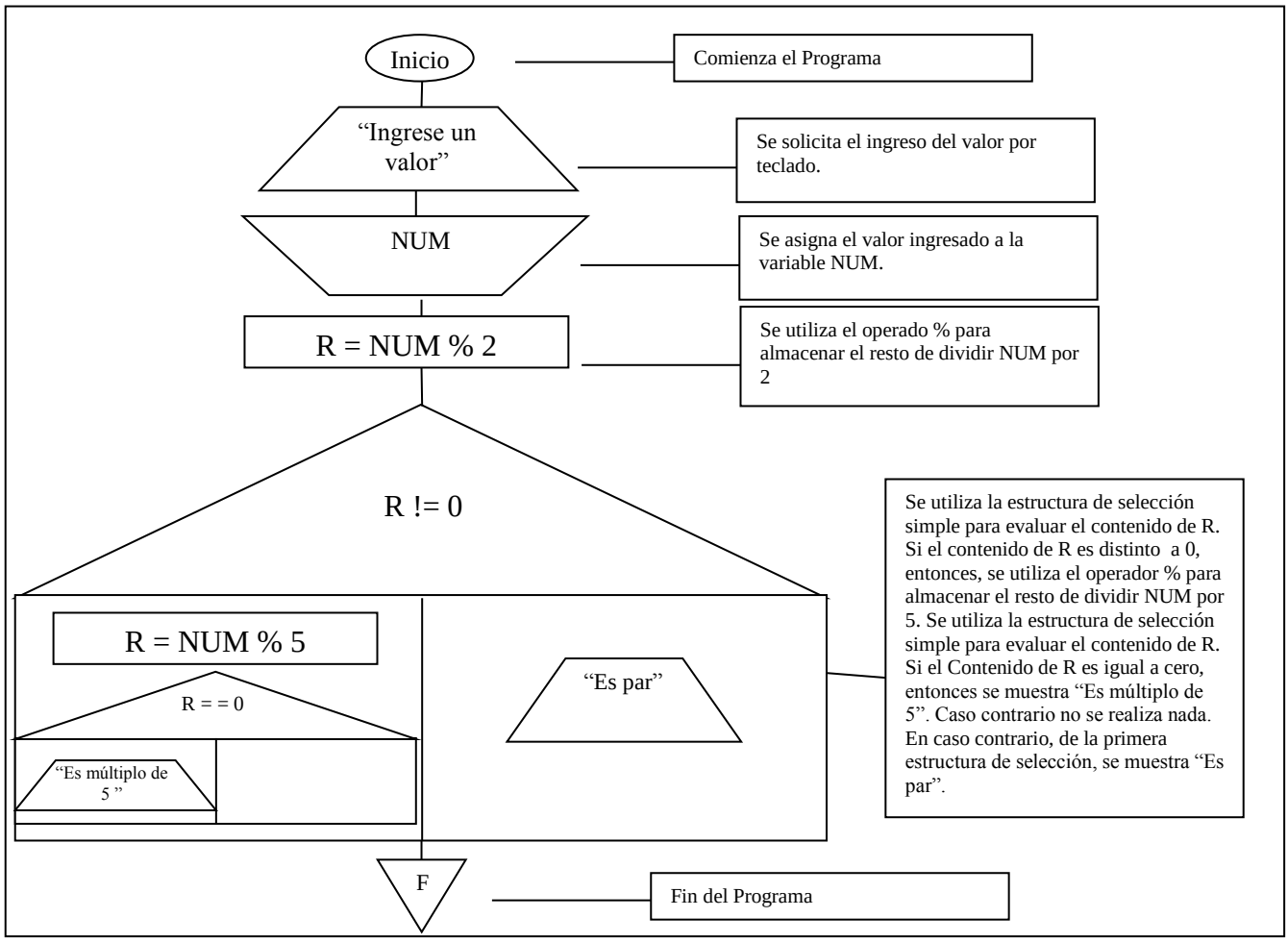


Ejemplo 3: Confeccionar un programa que pueda recibir un valor entero y nos informe si es un valor PAR, en caso de no serlo determinar e informar si dicho valor es múltiplo de 5.

ESTRATEGIA

1. Ingresar valor
2. Realizar operación para saber si es par, usando el operador % que almacena el resto.
3. Si R es igual a 0 informa "PAR"
4. De lo contrario, realiza operación para saber si es múltiplo de 5. Utilizando %
5. Si R es igual a 0 informa "MULT 5"

DIAGRAMA:



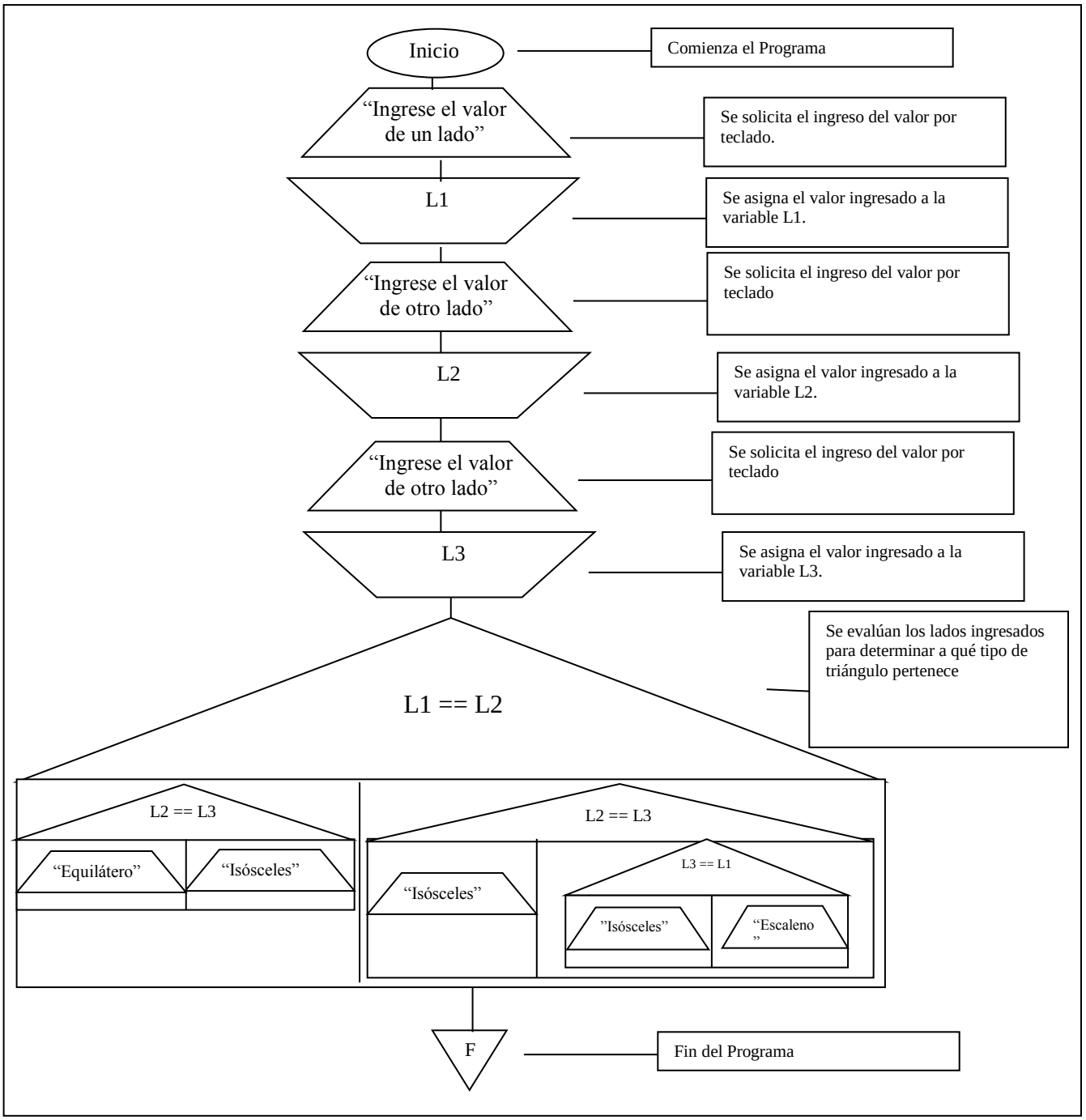
VARIANTE: ¿Cómo se debería modificar el diagrama si se quiere detectar los pares que a su vez son múltiplos de 5?

Ejemplo 4: Confeccionar un programa que ingresando los valores de los tres lados de un triángulo determine e informe si el triángulo es EQUILATERO, ISOSCELES ó ESCALENO.

ESTRATEGIA

1. Ingresar los valores de los 3 lados (L1, L2 y L3)
2. Determino si es equilátero (tres lados iguales), si no lo es, determino si es escaleno (tres lados distintos) y si o lo es, será isósceles (dos lados iguales y uno no).

DIAGRAMA



La única forma que hay de verificar si un programa que se realiza funciona adecuadamente para las diversas alternativas que se pueden presentar, es confeccionando un JUEGO o LOTE de PRUEBA. Este consiste en una serie de datos con los cuales se ejecuta el programa efectuando un seguimiento de las sentencias tal cual lo efectuaría la computadora y se analizan los resultados obtenidos. Es importante que al diseñar un lote de prueba se tenga en cuenta todos los casos posibles y asegurarse que se ejecuten todos los caminos posibles del programa. La tabla 2 muestra un posible lote de prueba para el ejemplo anterior.

Tabla 2: Posible lote de prueba para determinar el tipo de triángulo

L1	L2	L3	Resultado esperado
5	5	5	Equilátero
5	5	7	Isósceles
5	7	7	Isósceles
5	7	5	Isósceles

5	6	7	Escaleno

2 Codificación en C

La estructura de selección simple se codifica de la siguiente forma:

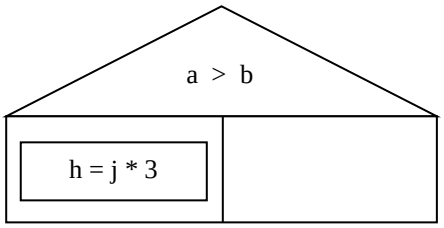
```
if ( Expresión Lógica )
{
      //instrucciones que se ejecutan si se cumple la condición es
      VERDADERA
}
else
{
      //instrucciones que se ejecutan si la condición es FALSA
}
```

Puede verse que tanto para la parte verdadera como para la falsa habrá un bloque que contiene las instrucciones entre llaves. La utilización del bloque entre llaves solo es necesario si dentro del verdadero o del falso hay más de una instrucción. Si solo hay una instrucción las llaves son opcionales.

Si desea realizar una acción "solo" si se cumple la condición, se utiliza solo la parte verdadera. El uso del else es opcional, por lo tanto si no hay instrucciones por la parte falsa directamente no se escribe el else.

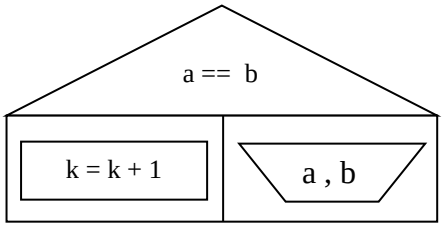
Es importante ser ordenados al escribir el código fuente para poder interpretarlo rápidamente al leerlo, para ello se utilizará lo que se conoce como identación. La identación es escribir el texto dejando espacio delante, se recomienda dejar ese espacio presionando la tacla tab (tabulación). En estas estructuras servirá para indicar cuáles son las instrucciones que están en la parte verdadera y cuales en la parte falsa, rápidamente en un golpe de vista. En los siguientes ejemplos se mostrará el código correctamente identado.

Ejemplo 1:



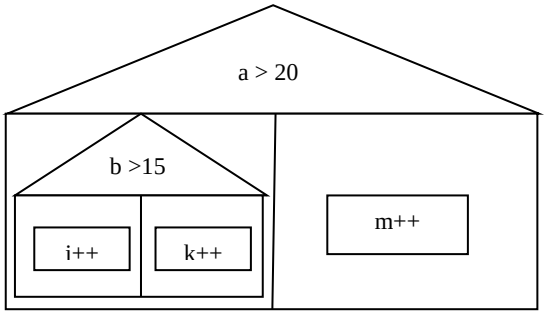

```
if (a>b)
      h=j*3;
```

Ejemplo 2:



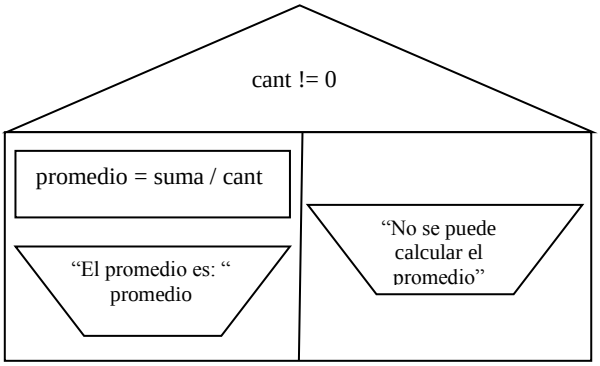

```
if ( a == b)
      k = k + 1;
else
      printf(" %f %f ",a,b);
```

Ejemplo 3:




```
if (a>20)
   if (b>15)
      j++;
   else
      k++;
else
   m++;
```

Ejemplo 4:




```
if (cant!=0)
{
   promedio = suma/cant;
   printf ("el promedio es: %f", promedio);
}
else
   printf ("no se puede calcular el promedio");
```

3 Operador Condicional (? :) [if reducido]

Forma general:

```
expresion?  Instruccion1: intruccion2;
```

Si la expresión es verdadera se ejecuta la instrucción1, caso contrario se ejecuta la instruccion2.

Por ejemplo, el siguiente programa muestra si un número ingresado es par o impar utilizando el operador condicional.

```
#include <stdio.h>
int main()
{
   int n;
   printf("Ingrese un numero:");
   scanf("%d", &n);
   (n%2==0)?printf("par"):printf("impar");
   return 0;
}
```

El mismo programa utilizando la sentencia if quedaría:

```
#include <stdio.h>
int main()
{
   int n;
   printf("Ingrese un numero:");
   scanf("%d", &n);
   if (n%2==0)
         printf("par");
   else
         printf("impar");
   return 0;
}
```

Solo sirve para el caso de que se tenga una única instrucción en el verdadero y una única instrucción en el falso. Se utiliza generalmente para asignar uno de dos valores a una variable. Por ejemplo, realizar un programa que ingrese dos números y calcula la resta si el primero es mayor o igual al segundo y sino calcule la suma.

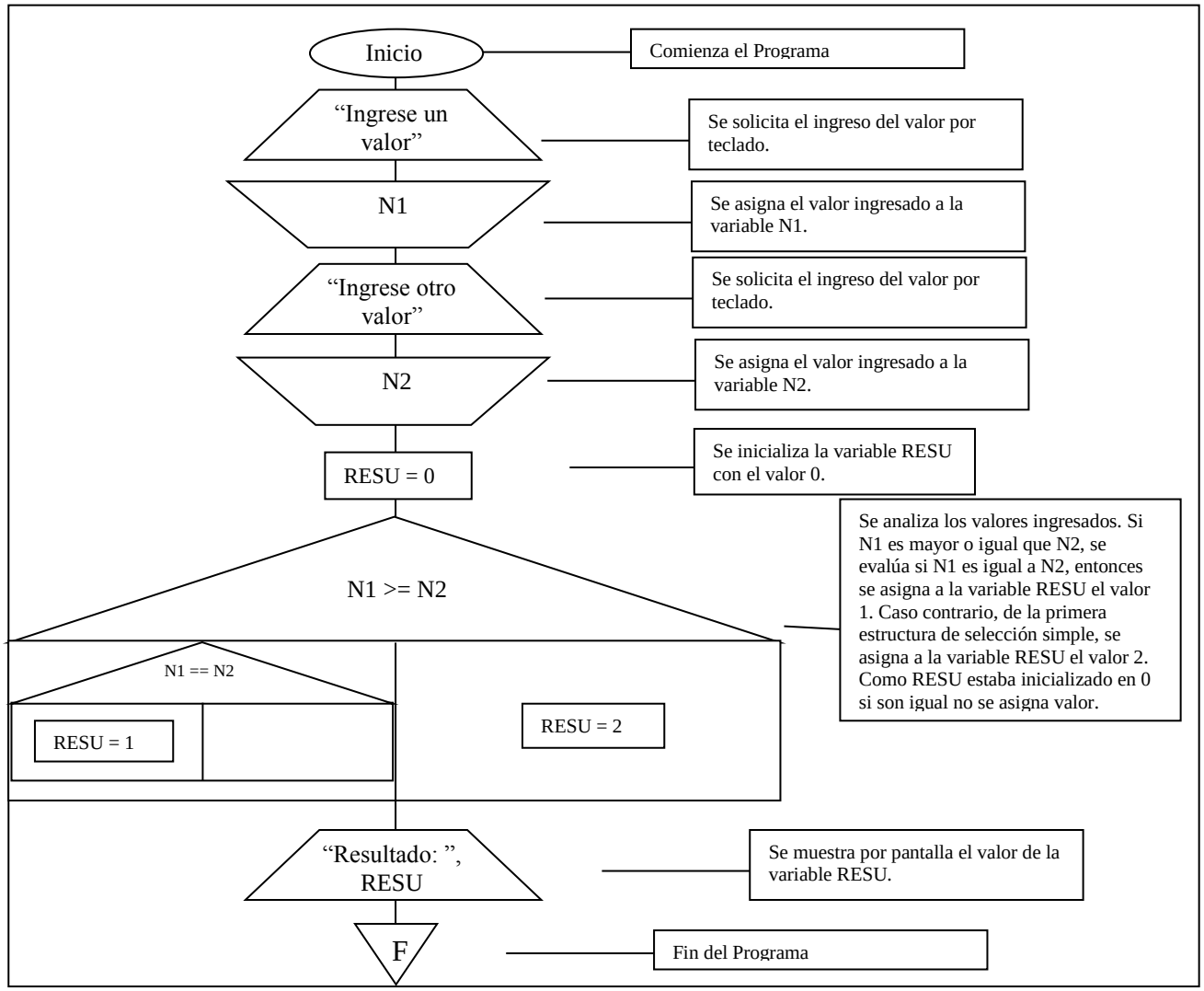
```
#include <stdio.h>
int main()
{
   int n1,n2, resultado;
   printf("Ingrese dos numeros:");
   scanf("%d%d",&n1,&n2);
   resultado = (n1>=n2)?n1-n2:n1+n2;
   printf ("El resultado es: %d", resultado);
   return 0;
}
```

4 Caso Particular de la sentencia IF

Este caso se presenta cuando una estructura "completa" tiene en su rama "verdadera" a una estructura de selección "incompleta". La solución para este caso llega con un par de llaves que encierren a la estructura de selección incompleta, transformándola en una estructura completa.

Recuerde también que cada estructura se caracteriza por tener un "único" punto de entrada y un "único" punto de salida, lo que equivale a hablar de una "sola" sentencia. La causa es debido a que por regla general "cada else siempre es apareado con el inmediato if anterior no cerrado"

Ejemplo: realizar un programa que permita ingresar dos números y de cómo resultado 0 si el primero es mayor que el segundo, 1 si son iguales, 2 si el segundo es mayor que el primero.



Recuerde que siempre para resolver un ejercicio hay muchas soluciones posibles, seguramente esta no es la mejor forma de plantearlo pero sirve para ejemplificar el caso particular que se quiere mostrar ya que dentro del verdadero del primer if hay un segundo if incompleto (sin else), pero el segundo if si tiene su parte falsa, por lo tanto, si se codifica de la siguiente forma:

```
#include <stdio.h>
int main()
{
   int n1,n2, resu=0;
   printf("Ingrese dos numeros:");
   scanf("%d%d",&n1,&n2);
   if (n1>=n2)
         if (n2==n1)
               resu=1;
   else
         resu=2;
   printf ("Resultado: %d",resu);
   return 0;
}
```

Al probarlo si se ingresa, por ejemplo, el valor 10 para el primero número y 2 para el segundo, se debería obtener como resultado un 0 ya que el primer número es mayor al segundo, sin embargo, muestra un 2 ya que el código está mal hecho.

Es incorrecto ya que el else será tomado como parte del if inmediatamente anterior. Para solucionarlo se debe armar un bloque para completar la secuencia if incompleta de la siguiente forma:

```
if (n1>=n2)
{
   if (n1==n2)
         resu =1;
}
else
   resu =2;
```

Otra forma de resolverlo sería incluyendo un else vació de la siguiente forma:

```
if (n1>=n2)
   if (n1==n2)
         resu =1;
   else;
else
   resu =2;
```

5 Operadores Lógicos

Los operadores lógicos permiten asociar varias expresiones lógicas ó aritméticas formando solo una, lo cual simplificará la cantidad estructuras de selección que se deben utilizar.

Los operadores lógicos son 3:

- AND (y)
- OR (ó)
- NOT (negación)

En el lenguaje C, se utilizan los símbolos de la tabla 3 para expresar los operadores:

Tabla 3: Operadores lógicos en el lenguaje C

Operador	Símbolo
AND	&&
OR	||
NOT	!

Estos operadores trabajan sobre valores lógicos, por lo tanto, pueden aplicarse sobre condiciones o sobre valores numéricos de una variable recordando que el 0 es falso y cualquier valor distinto de cero es verdadero.

El operador AND (&&) evalúa la dos condiciones y retorna verdadero solo en el caso de que ambas condiciones sean verdaderas, es decir, se deben cumplir ambas al mismo tiempo.

El operador OR (||) evalúa las dos condiciones y retorna verdadero cuando al menos una de las dos es verdadera, es decir, se debe cumplir al menos una de las condiciones para que sea verdadera la expresión.

El operador NOT (!) trabaja sobre una sola condición negándola. Es decir, que si es verdadera la hace falsa y si es falsa la hace verdadera.

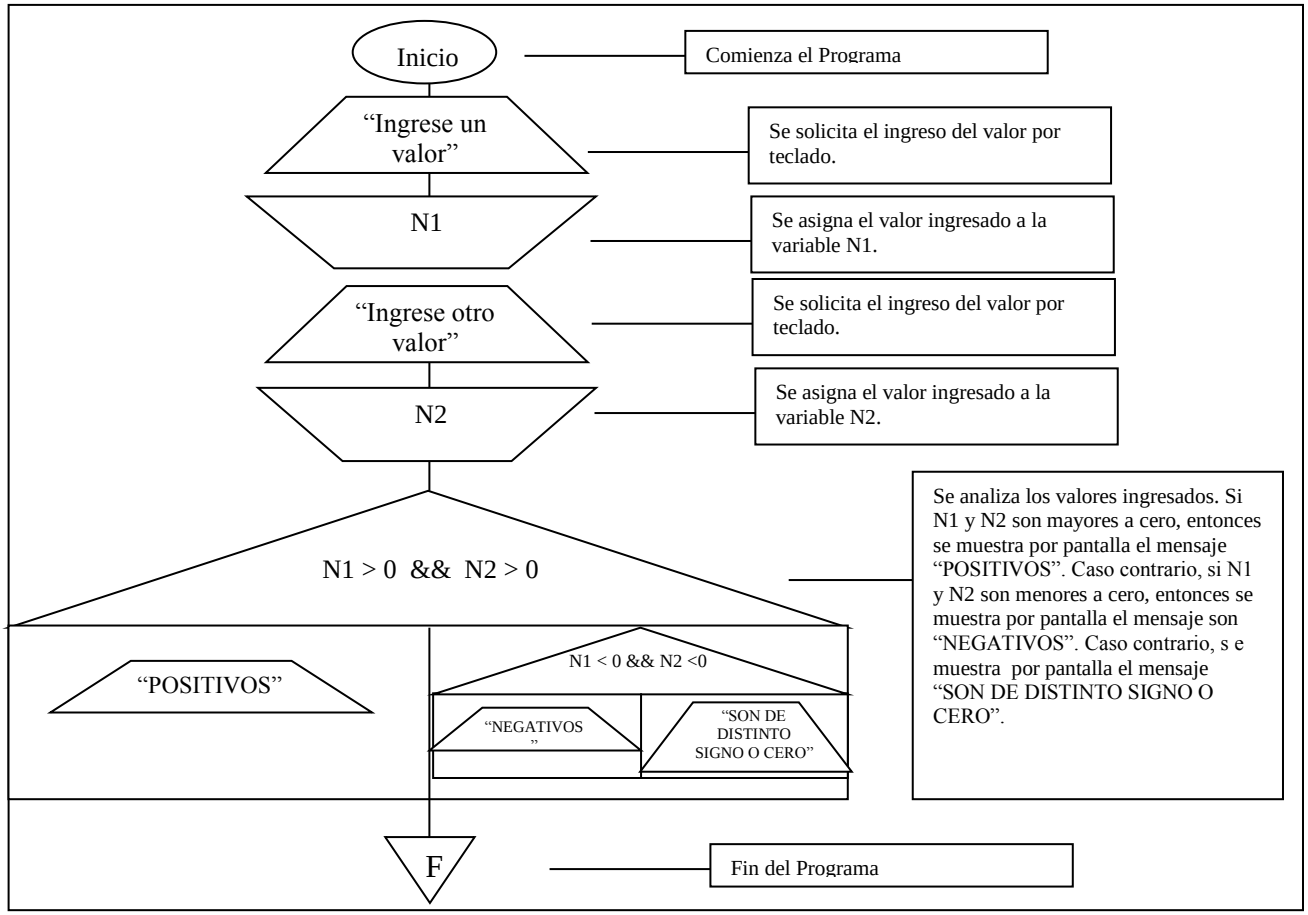
La tabla 4 muestra la tabla de verdad de los operadores lógicos, siendo A y B condiciones lógicas.

Tabla 4: Tabla de verdad de los operadores lógicos

A	B	A && B	A || B	!A
F	F	F	F	V
F	V	F	V	V
V	F	F	V	F
V	V	V	V	F

Las precedencias de los operadores son: NOT (!) - AND (&&) - OR (||)

Ejemplo 1: Ingresar dos números. Indicar si ambos son positivos, ambos son negativos, o son distinto signo o cero.

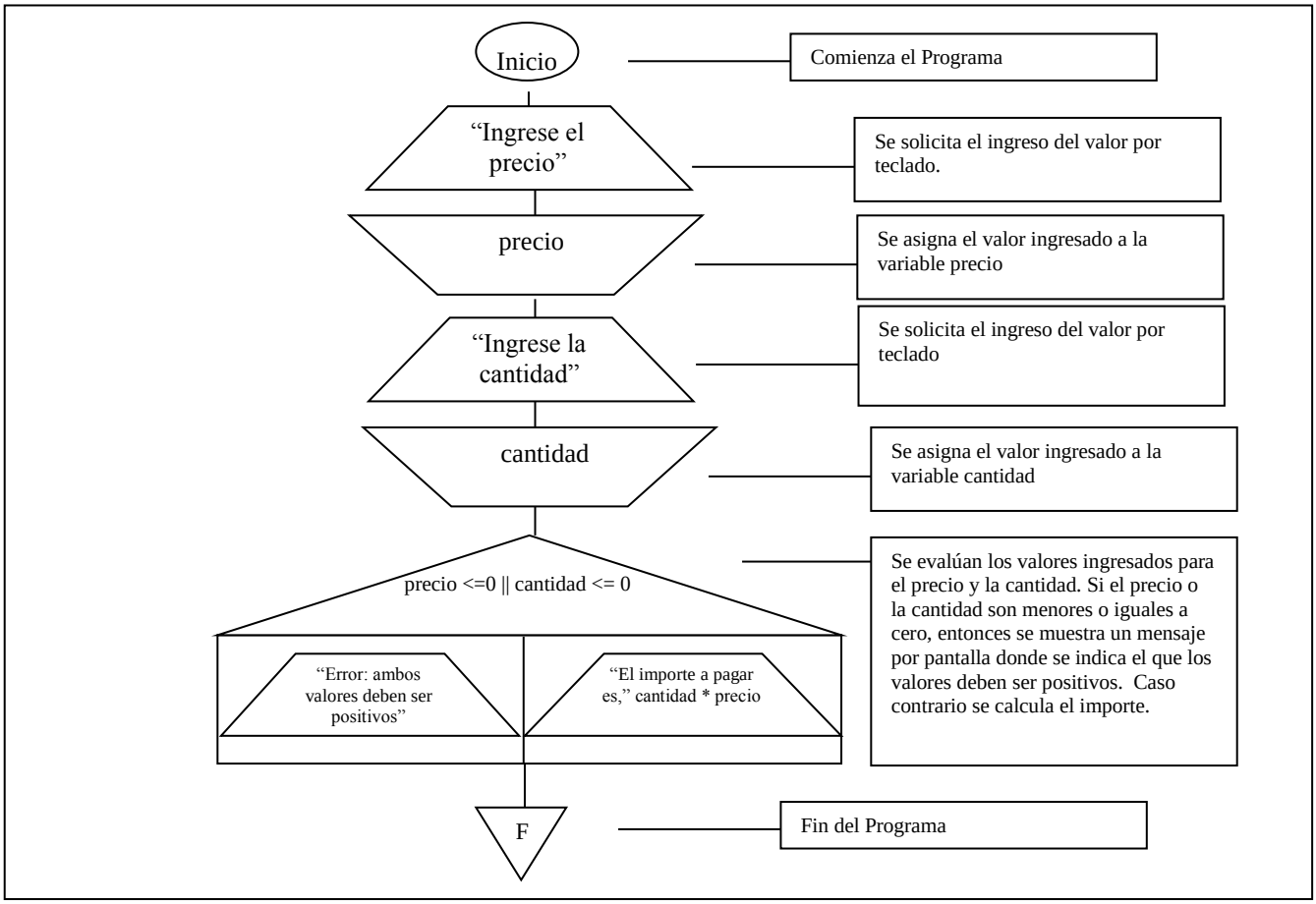


Esta es solo una de las formas de resolver este ejercicio ya que otra persona podría comenzar preguntando por ejemplo si ambos son negativos, o si son cero o de distinto signo. Recordemos que siempre hay varias formas para resolver un mismo ejercicio.

***ACLARACIÓN IMPORTANTE**:* El lenguaje C evalúa las condiciones en orden, si detecta que ya se cumple o no, no sigue evaluando las otras condiciones. Otros lenguajes evalúan todas las condiciones por más que ya se sepa el resultado.

Por lo tanto, en este ejemplo si en N1 se ingresa un número negativo o cero y al ser la primera condición un AND, la misma ya es falsa y no necesita evaluar el valor de N2.

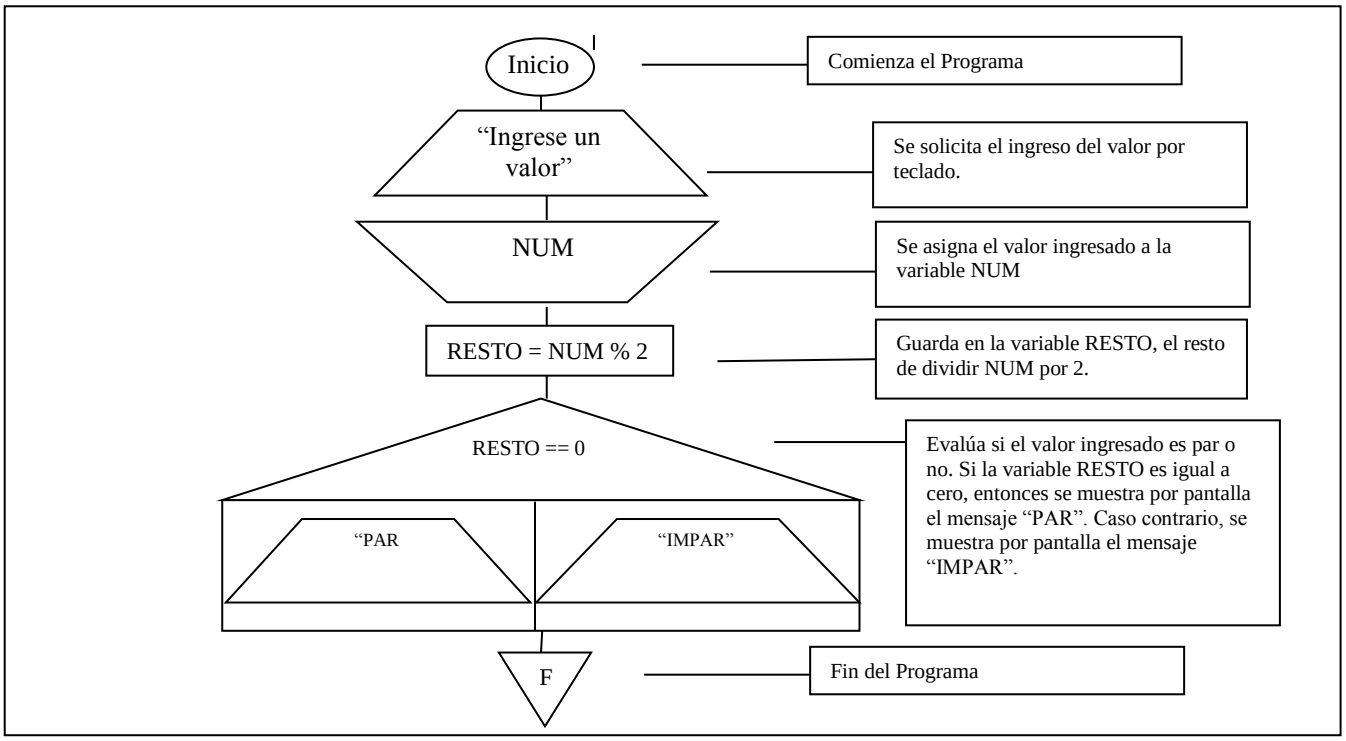
Ejemplo 2: Ingresar precio y cantidad de productos a vender. Mostrar el importe a abonar. Si alguno de los dos es menor o igual a cero informar un error.



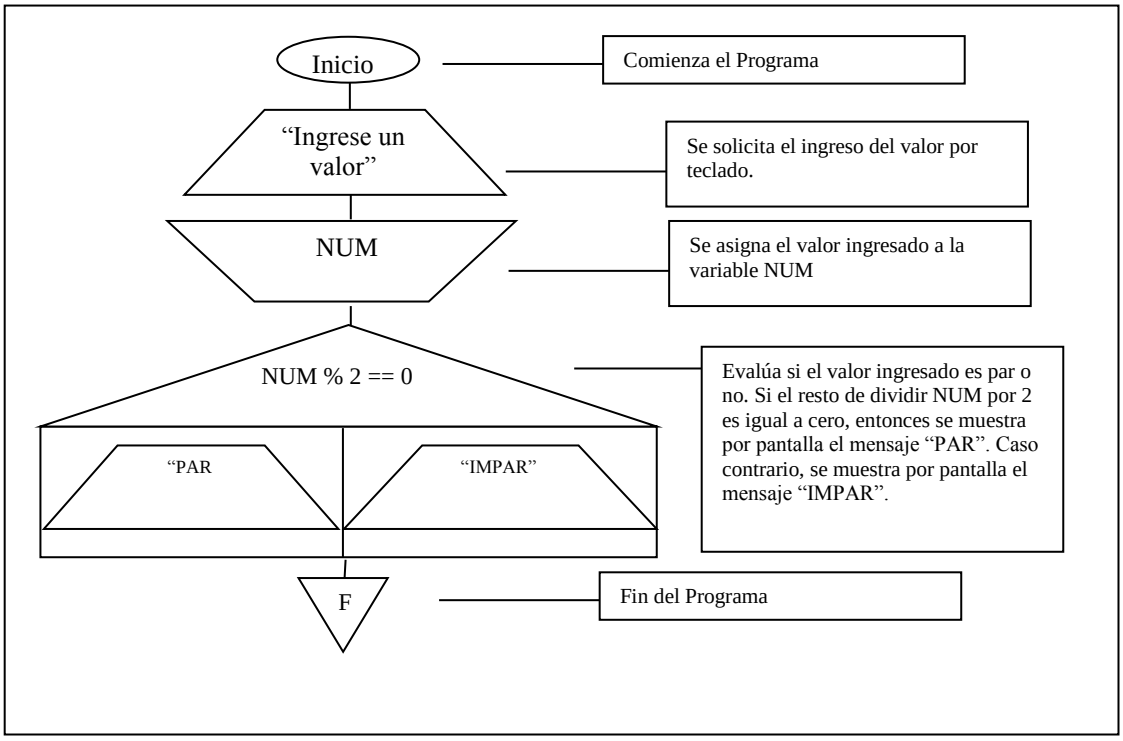
Nuevamente hay que tener en cuenta cómo evalúa las condiciones el lenguaje C. En la primera condición si precio es menor o igual a cero al ser un OR ya la condición es verdadera, y por lo tanto, no evalúa el valor de cantidad.

Ejemplo 3: Ingresar un número e indicar si es par o impar.

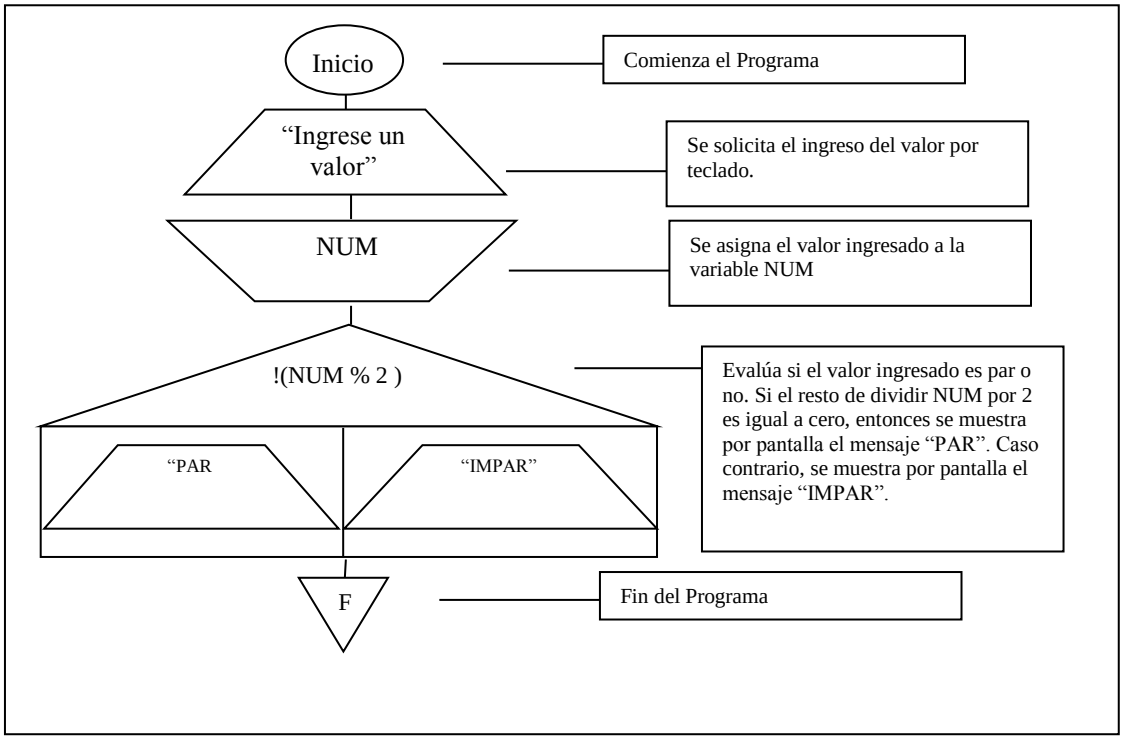
Una forma de resolver este ejercicio es la siguiente:



También se puede quitar la variable resto y hacer la pregunta directamente en la condición de la siguiente manera:



Si se toma en cuenta que en el lenguaje C el 0 es falso y cualquier valor distinto de cero es verdadero se puede directamente evaluar el resultado de la operación sin utilizar un comparador. Si la operación NUM % 2 da como resultado 0 será entonces falsa, si se quiere hacer verdadera cuando sea cero se debe negar quedando de la siguiente forma:

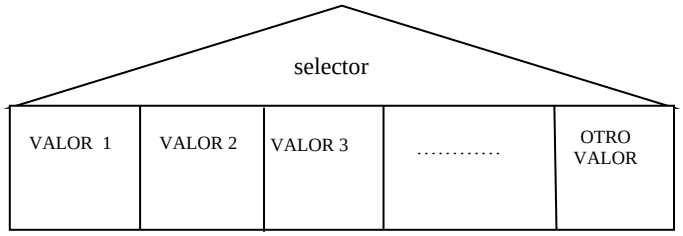


6 Estructura de Selección Múltiple (switch)

Piense como sería un programa que ingresando el número de un mes tiene que informar el nombre. Utilizando la sentencia if se tienen que anidar las condiciones preguntado si el mes es 1 mostrar enero, sino preguntar si el mes es dos mostrar febrero, sino preguntar si el mes es 3, y así sucesivamente, hasta abarcar todas las condiciones.

Para casos de este tipo existe una "estructura condicional múltiple", que generaliza a la sentencia <if –else>. Esta estructura es útil, por ejemplo, cuando hay un problema en el cual se deben realizar acciones diferentes en base al valor de un código que puede adoptar diversos valores, lo cual obliga a utilizar diversas sentencias if encadenadas.

Su diagrama y codificación son los siguientes:




```
switch (selector)
{
       case valor1:
                    //instrucciones a ejecutar si selector toma el valor1
                    break;
       case valor2:
                    //instrucciones a ejecutar si selector toma el valor1
                    break;
       case valor3:
                    //instrucciones a ejecutar si selector toma el valor1
                    break;
```

```
        ….

        default: //instrucciones a ejecutar si selector no toma ninguno de los valores
anteriores
}
```

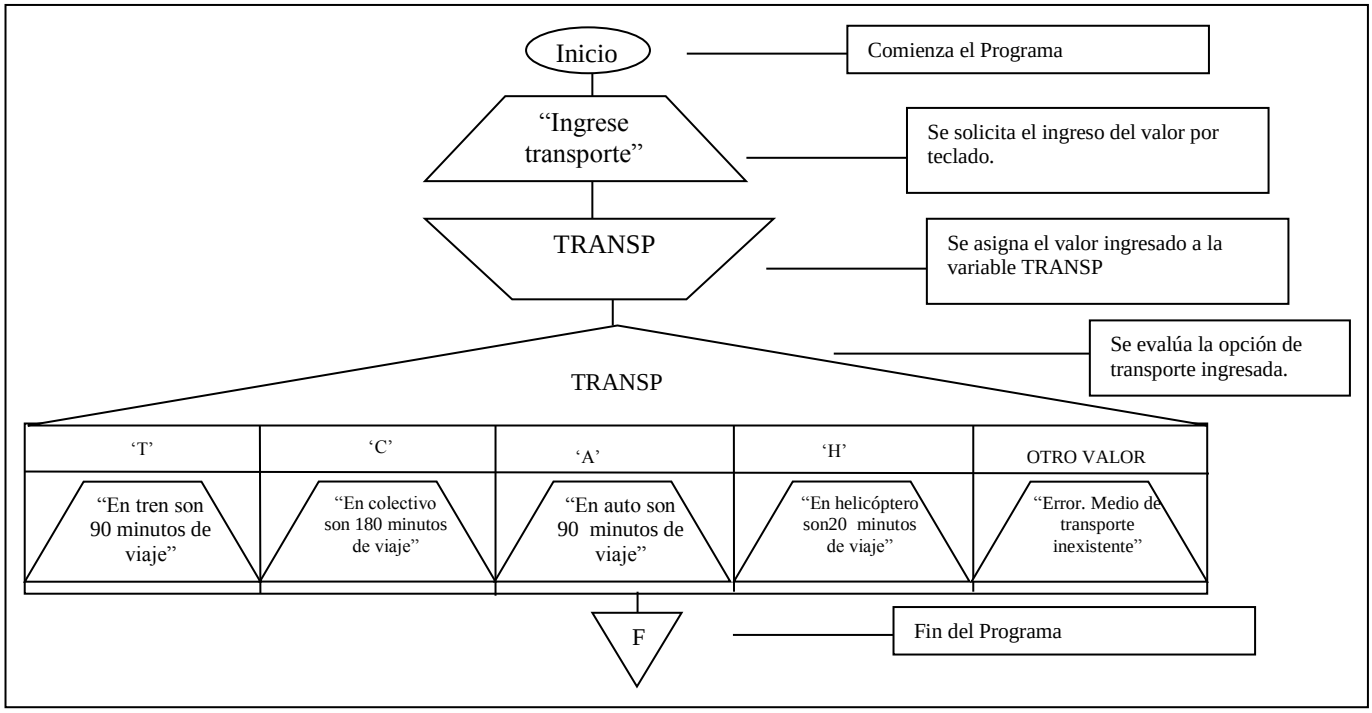
Consideraciones:

- Se evalúa el selector y si su valor coincide con alguno de los casos establecidos se pasa directamente a ejecutar el código de dicho caso
- El default es el valor por defecto, es opcional y se ejecuta si selector no toma ninguna de los valores anteriores.
- Selector puede ser una variable o una expresión pero debe ser del tipo **entera** o **carácter**.
- Los valores deben ser constantes (numéricas o caracteres). **NO pueden ser rangos**. Si se necesita evaluar rangos se debe utilizar la sentencia if.
- Cada caso termina con la instrucción break que hace que la ejecución se corte y salga de la estructura del switch. Si no se coloca el break en un caso seguirá ejecutando el código de los casos que están debajo hasta encontrar un break o llegar a la llave de fin. Es por esto que el default, o si el default no está, el último caso no necesita de la instrucción break. Esta particularidad permite que distintos casos compartan el mismo código.

Los casos no tiene necesidad de estar ordenados, pero cada valor solo puede aparecer una vez.

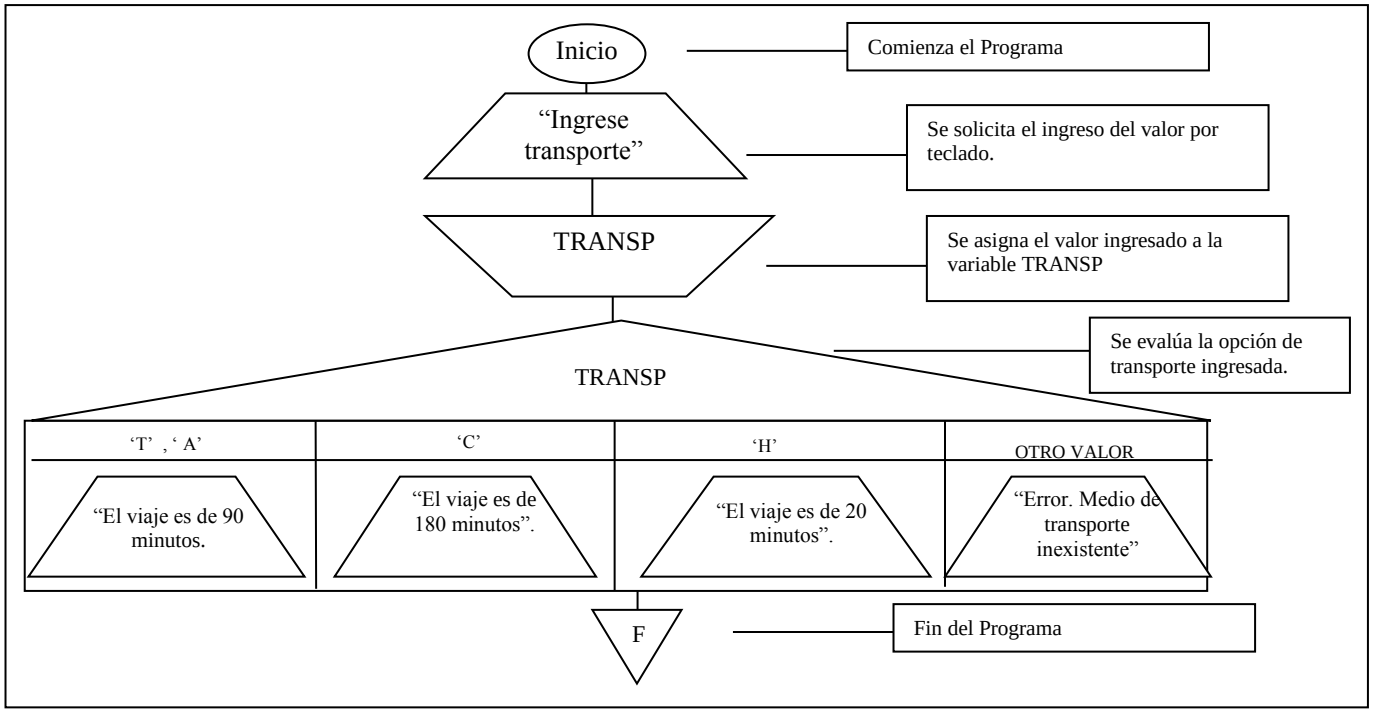
Ejemplo 1: Confeccionar un programa que solicite el medio de transporte para llegar desde La Plata hasta la UNLaM. Mostrando en pantalla el tiempo estimado de viaje para cada medio de transporte. El medio de transporte se ingresa con una letra siendo:

- T: tren, tiempo estimado de viaje 90 min.
- C: colectivo, tiempo estimado de viaje 180min
- A: auto, tiempo estimado de viaje, 90 min.
- H: helicóptero, tiempo estimado de viaje, 20 min.



Nótese que los tiempos en tren y auto son los mismos.

Entonces el algoritmo anterior se puede mejorar de la siguiente manera.



El diagrama quedará de la siguiente manera en código:

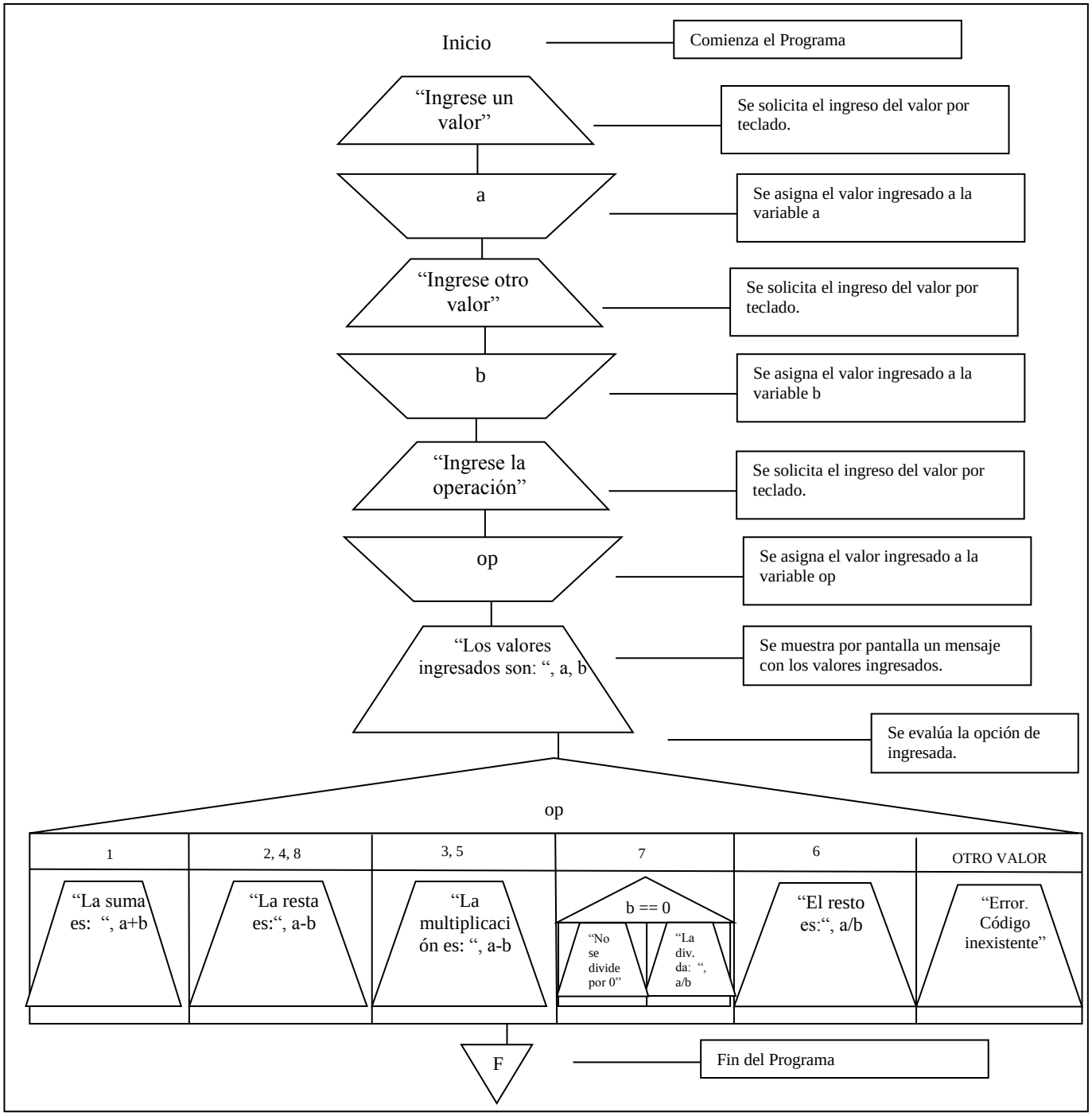
```
#include <stdio.h>
int main()
{
   char tansp;
   printf("Ingrese transporte:");
   scanf("%c",&transp);

   switch(transp)
   {
         case 'T':
         case 'A': printf("El viaje es de 90 minutos.");
                   break;
         case 'C': printf("El viaje es de 180 minutos.");
                   break;
         case 'H': printf("El viaje es de 20 minutos.");
                   break;
         default: printf("Error. Medio de transporte
inexistente.");
   }

   return 0;
}
```

Ejemplo 2: Confeccionar un programa que permita ingresar un par de valores enteros a y b, que corresponden a las temperaturas registradas en un día y un código que indica la operación que se debe realizar con dichos valores.

El código de operación puede variar entre 1 y 8. Según el valor del código se pide calcular e informar:

Valor código	Calcular
1	a + b
2, 4 ú 8	a – b
3 ó 5	a * b
7	a / b
6	a % b
< 1 ó > 8	rechazar

Por cada código ingresado imprimir los valores leídos y el resultado obtenido, con leyendas.



El diagrama quedará de la siguiente manera en código:

```
#include <stdio.h>
int main()
{
  int a,b,op;

  printf("Ingrese un valor:");
  scanf("%d",&a);

  printf("Ingrese otro valor:");
  scanf("%d",&b);

  printf("Ingrese la opción:");
  scanf("%d",&op);

  printf("Los valores ingresados son:%d %d",a,b);

  switch(op)
  {
        case 1: printf("La suma es: %d",a+b);
                break;
        case 2:
        case 4:
        case 8: printf("La resta es: %d",a-b);
                break;
        case 3:
        case 5: printf("La multiplicación es: %d",a*b);
                break;
        case 7: if(b==0)
                    printf("No se puede dividir por cero");
                  else
                    printf("La división es: %d",a/b);
                break;
        case 6: printf("El resto de la división es: %d",a%b);
                break;
        default: printf("Error. Código inexistente.");
  }

  return 0;
}
```