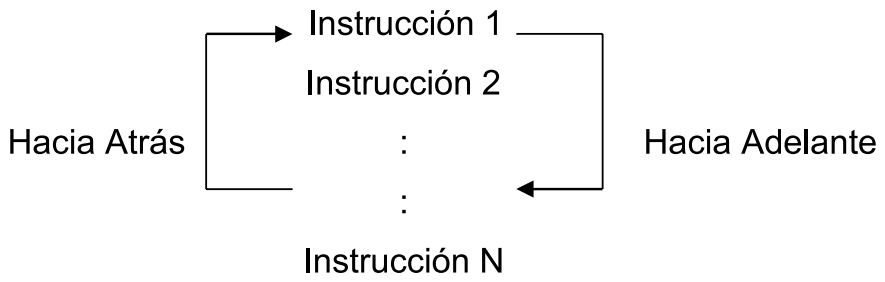
Capítulo 3

3. INSTRUCCIONES DE SELECCIÓN

En pseudocódigo existe un tipo de instrucción llamada de Bifurcación. Una instrucción de bifurcación obliga a tomar un camino basado en la evaluación de una condición.

Las instrucciones de bifurcación se subdividen en: Estructuras de Selección y Estructuras de Repetición. En el caso de las estructuras de selección, las instrucciones se ejecutan hacia adelante, siguiendo la secuencia del cuerpo de instrucciones del programa; en las estructuras de repetición las instrucciones se pueden ejecutar hacia adelante o hacia atrás dentro del bloque de ejecución del programa.



3.1 ESTRUCTURAS DE SELECCIÓN SIMPLE

La estructura de selección simple, tiene el siguiente formato general en el pseudocódigo, ver Ilustración 8:

Si (**condición**) **entonces**
Instrucción 1
Instrucción 2
:
:
Instrucción N
Fin_si

Ilustración 8. Formato general pseudocódigo para la Estructura de Selección Simple.

La estructura de selección simple contiene tres palabras reservadas: **Si**, **entonces** y **Fin_si**. Siempre comienza con la palabra **Si** y termina con **Fin_si**. Además, contiene un cuerpo de instrucciones entre ambas palabras, y una **condición** dentro de paréntesis. La **condición** al ser evaluada debe siempre generar un valor de tipo booleano: **verdadero** o **falso**.

La **condición** se puede construir de dos formas:

- Usando los operadores relacionales y/o operadores lógicos en una expresión que al ser resuelta genera un valor booleano
- O usando una variable de tipo booleano.

En ambos casos, las variables empleadas en la construcción de la **condición** deben estar declaradas y contener un valor antes de ser evaluadas.

Cuando la **condición** genera un resultado **verdadero** automáticamente se ejecuta de una manera secuencial el cuerpo de instrucciones, al ejecutarse la última instrucción -Instrucción N- el programa continúa con el **Fin_si**, dando por terminada la ejecución del **Si**. A continuación ejecutará las instrucciones que se encuentren después del **Fin_si**. Véase Ilustración 9.

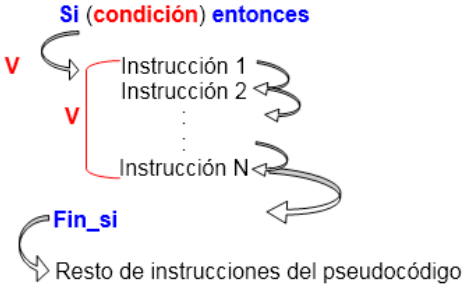


Ilustración 9. Funcionamiento de la estructura de selección simple cuando la condición da Verdadero.

Cuando la **condición** genera un resultado Falso automáticamente se salta a la palabra reservada **Fin_si**, se termina la ejecución del **Si** y se sigue con la ejecución del resto de instrucciones del programa, ver Ilustración 10.

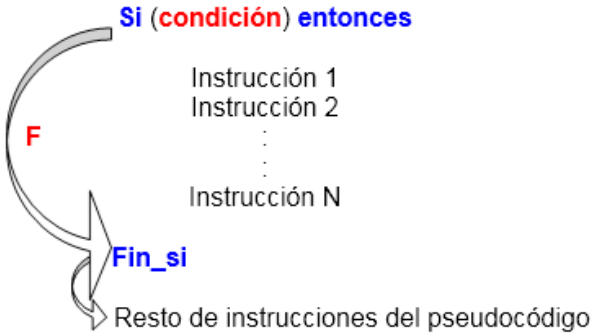


Ilustración 10. Funcionamiento de la estructura de selección simple cuando la condición da Falso.

La Ilustración 11 muestra el formato general en el diagrama de flujo para la estructura de selección simple.

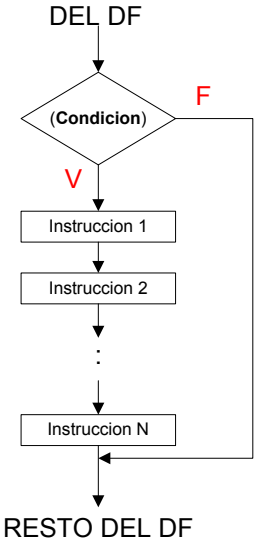


Ilustración 11. Formato general Diagrama de Flujo para las estructura de selección simple.

Note que en el diagrama de flujo de la Ilustración 11 no se coloca las palabras reservadas **Si**, **entonces** y **Fin_si**. El **Fin_si** se representa por la unión de dos flechas. En el camino del Falso no existe instrucción alguna y llega direct-

amente a la flecha que sale de la ejecución de la última instrucción para así continuar con la ejecución del resto de instrucciones. Las palabras DEL DF y RESTO DEL DF indican que existen otras porciones del diagrama de flujo que no se muestra aquí.

3.2 ESTRUCTURAS DE SELECCIÓN COMPUESTA

La estructura de selección compuesta tiene el siguiente formato general en el pseudocódigo, ver Ilustración 12:

Si (**condición**) **entonces**

Instrucción 1
Instrucción 2
:
:
Instrucción N

De lo contrario

Instrucción O
Instrucción P
:
:
Instrucción Z

Fin_si

Ilustración 12. Formato general pseudocódigo para la Estructura de Selección Compuesta.

La Estructura de Selección Compuesta contiene cuatro palabras reservadas: **Si**, **entonces**, **De lo contrario** y **Fin_si**. Siempre comienza con la palabra **Si** y termina con **Fin_si** y contiene dos cuerpos de instrucciones: el primero entre las palabras reservadas **Si** y **De lo contrario** y el segundo, entre las palabras **De lo contrario** y **Fin_si**. Además, contiene una **condición** dentro de paréntesis. La **condición** al ser evaluada debe siempre generar un valor booleano: verdadero o falso.

La siguiente figura ilustra su funcionamiento. Cuando la **condición** genera un resultado **verdadero** se ejecuta el primer cuerpo de instrucciones. El segundo cuerpo de instrucciones se ejecuta cuando la **condición** genera un resultado **falso, ver** Ilustración 13.

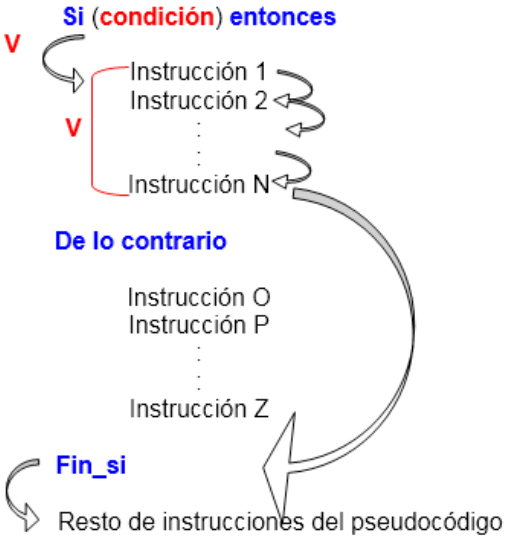


Ilustración 13. Funcionamiento de la estructura de selección compuesta cuando la condición da verdadero.

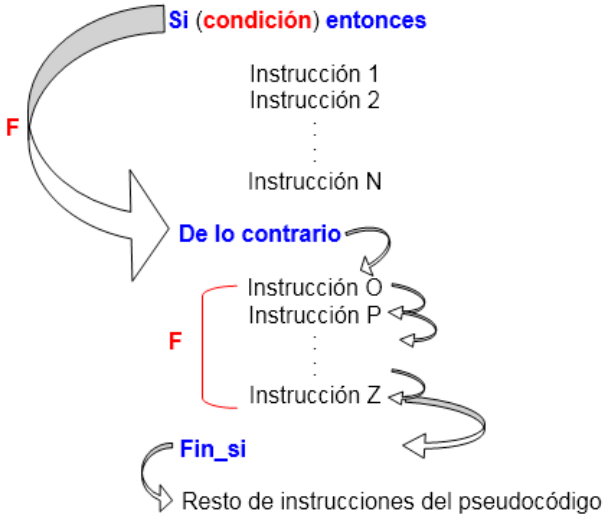


Ilustración 14. Funcionamiento de la estructura de selección compuesta cuando la condición da falso.

La Ilustración 13 y la Ilustración 14, muestran que después de ejecutar la última instrucción del cuerpo de instrucciones correspondiente, el programa salta inmediatamente a ejecutar el **Fin_si** y sigue con la ejecución del resto de instrucciones.

La **condición** de la estructura de selección compuesta se construye de igual forma que la **condición** de la estructura de selección simple.

La siguiente Ilustración 15 muestra el formato general en el diagrama de flujo para la estructura de selección compuesta.

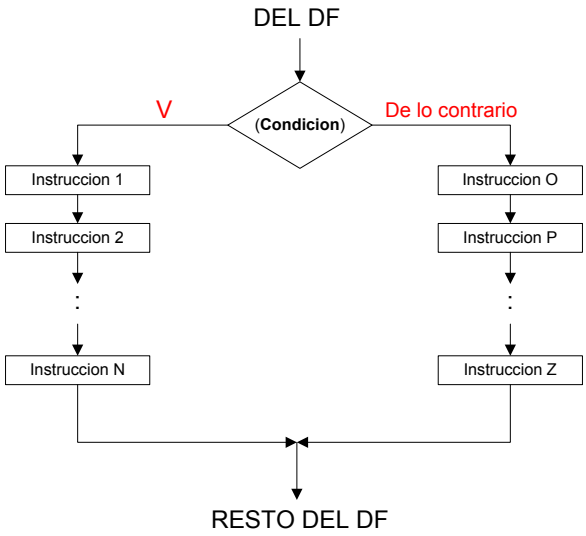


Ilustración 15. Formato general diagrama de flujo estructura de selección compuesta.

3.3 ESTRUCTURAS DE SELECCIÓN MÚLTIPLE

La estructura de selección múltiple tiene el siguiente formato general en el pseudocódigo, ver Ilustración 16:

Si (**selector**) **igual**

Valor 1 **:** Instrucción 1
Instrucción 2
:
Instrucción N

```
        Valor 2 :        Instrucción 1
                         Instrucción 2
                               :
                         Instrucción N

                  :              :
        Valor N :        Instrucción 1
                         Instrucción 2
                               :
                         Instrucción N

   De lo contrario :     Instrucción 1
                         Instrucción 2
                               :
                         Instrucción N

Fin_si
```

Ilustración 16. Formato general pseudocódigo para la estructura de selección múltiple.

La estructura de selección múltiple utiliza un **selector** en vez de una **condición**. Está conformada por las palabras reservadas: **Si**, **igual**, **De lo contario** y **Fin_si**. Entre las palabras **Si** y **Fin_si,** al lado izquierdo de dos puntos, se coloca toda la gama de valores posibles que puede generar el **selector**. Al lado derecho de los dos puntos se colocan las instrucciones que se deben ejecutar cuando exista una igualdad entre el valor generado por el **selector** y el valor que se encuentra al lado izquierdo de los dos puntos. Algunos autores utilizan **Según sea**, **haga** y **Fin_según_sea** en vez de **Si**, **igual** y **Fin_si**.

El **selector** es una variable o una expresión conformada por operadores que al ser resuelta genera un valor. El valor y el tipo de dato se comparan con Valor 1, Valor 2,…, Valor N y si es igual a alguno de ellos, el programa ejecuta el cuerpo de instrucciones que se encuentra a su lado. Cuando se ejecuta la última instrucción, el programa automáticamente salta a ejecutar el **Fin_si** y luego sigue con el resto de instrucciones del programa.

Si el valor generado por el **selector** no se encuentra dentro de la gama de valores posibles, automáticamente el programa busca la opción **De lo contrario** y ejecuta el cuerpo de instrucciones asociado a ella. El **De lo contrario** es opcional, esto significa que no es obligatorio colocarlo.

La(s) variable(s) que forman el **selector** deben: estar declaradas, contener un valor y coincidir en tipo con la gama de valores posibles.

La siguiente Ilustración 17 muestra uno de los posibles funcionamientos de la estructura de selección múltiple:

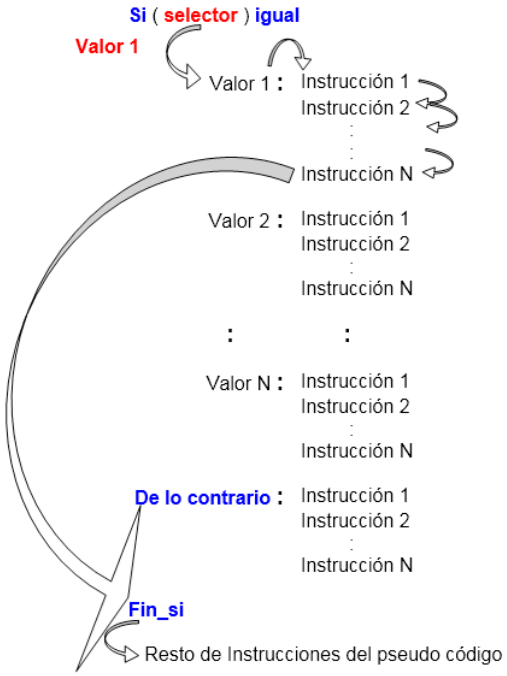


Ilustración 17. Funcionamiento de la estructura de selección múltiple cuando selector genera Valor 1.

La siguiente Ilustración 18 muestra el funcionamiento cuando **selector** genera un dato que no coincide con ninguno de los valores, Valor 1, Valor 2, … , Valor N.

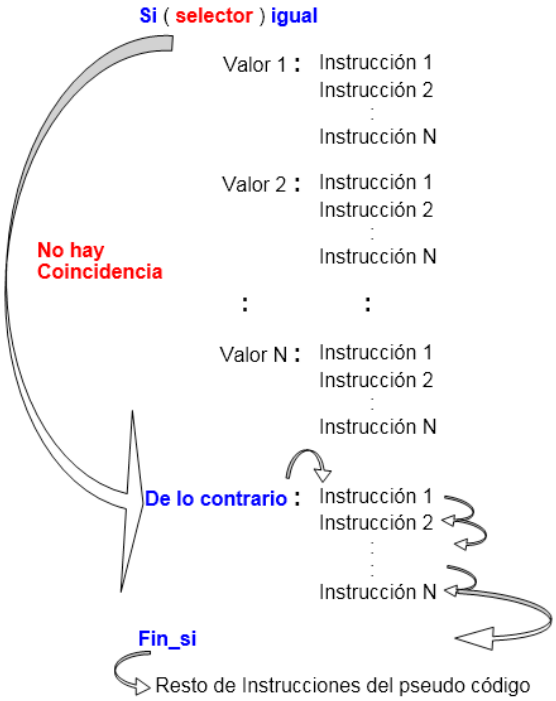


Ilustración 18. Funcionamiento cuando selector genera un valor que no coincide con ninguno de los valores dentro de la estructura de selección múltiple.

La siguiente Ilustración 19 muestra el funcionamiento cuando **selector** genera un dato que no coincide con ninguno de los valores, Valor 1, Valor 2, … , Valor N y no existe **De lo contrario**.

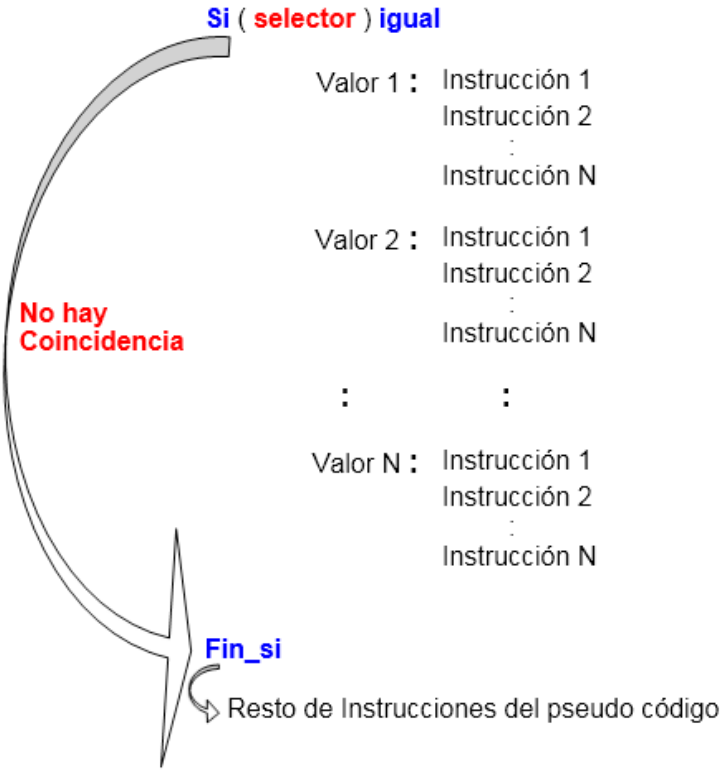


Ilustración 19. Funcionamiento cuando selector genera un valor que no coincide con ninguno de los valores y tampoco existe el De lo contario.

Al no existir coincidencia salta automáticamente al **Fin_si**, cerrándose la estructura, luego se sigue la ejecución del resto de instrucciones del pseudocódigo.

La siguiente Ilustración 20 muestra el formato general en el diagrama de flujo para la estructura de selección múltiple.

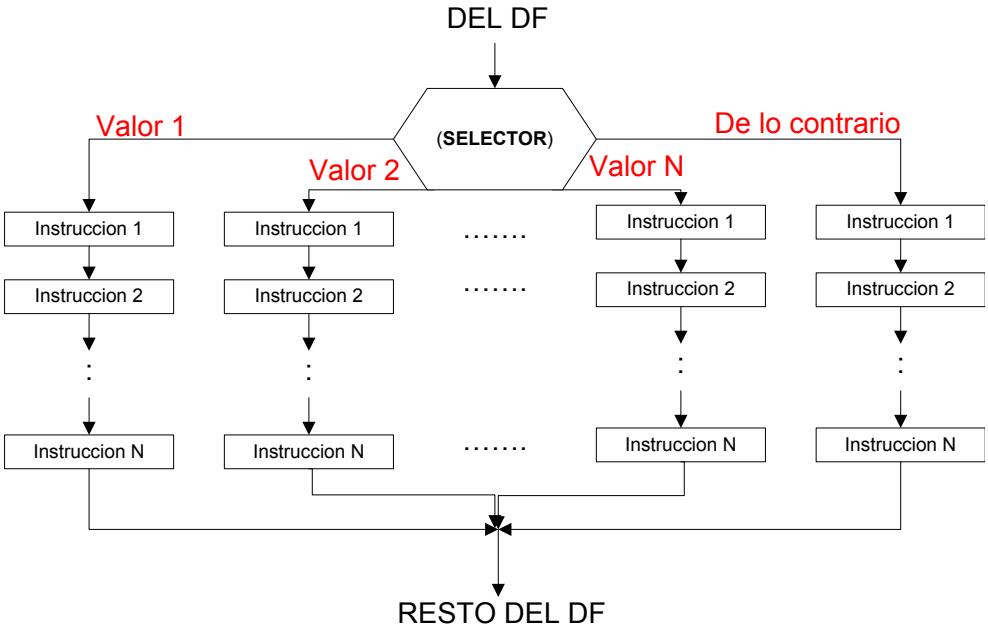


Ilustración 20. Formato general Diagrama de Flujo para la estructura de selección múltiple.

3.4 ESTRUCTURAS DE SELECCIÓN ANIDADAS

La estructura de selección anidada no tiene un formato general en el pseudocódigo, ni en el diagrama de flujo. Consiste en utilizar las estructuras anteriormente mencionadas dentro de alguna o varias de ellas mismas, por ejemplo, utilizar varias estructuras de selección simple dentro de otra estructura de selección simple, o dentro de una estructura de selección compuesta, o dentro de una estructura de selección múltiple, o viceversa. No existe una normal general en la construcción de las expresiones de selección anidadas ni en la cantidad de expresiones de selección que se pueden colocar.

Aspectos a tener en cuenta para construir una expresión de selección anidada:

- Deben existir tantos **Fin_si** como **Si** existan. Debe tenerse presente que un **Fin_si** nunca cierra más de un **Si**.
- El **Fin_si** más interno le corresponderá única y exclusivamente al último **Si** más interno.

- Claridad de dónde colocar cada **Si** y cada **Fin_si**, sino se ubican bien algunos **Si** podrían depender de otros **Si** y por tanto no ejecutarse apropiadamente o nunca ejecutarse.

Las siguientes ilustraciones muestran algunos ejemplos:

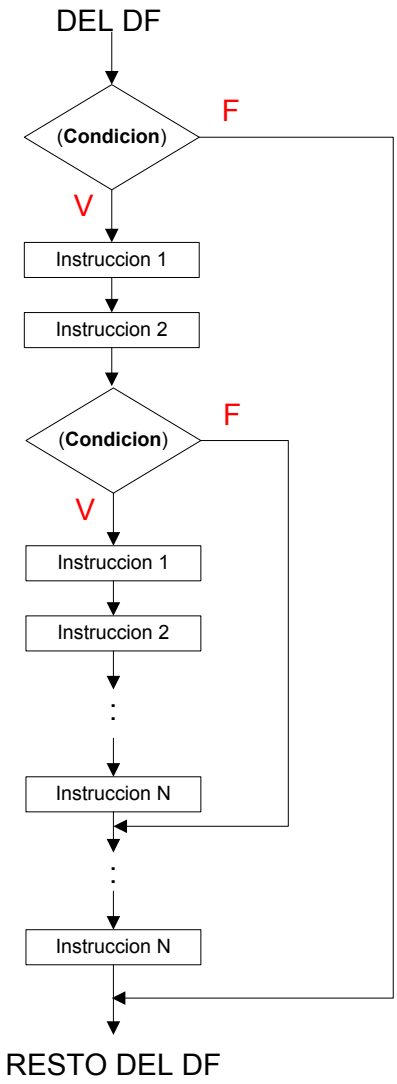


Ilustración 21. Estructura de selección anidada formada por una expresión de selección simple dentro de otra.

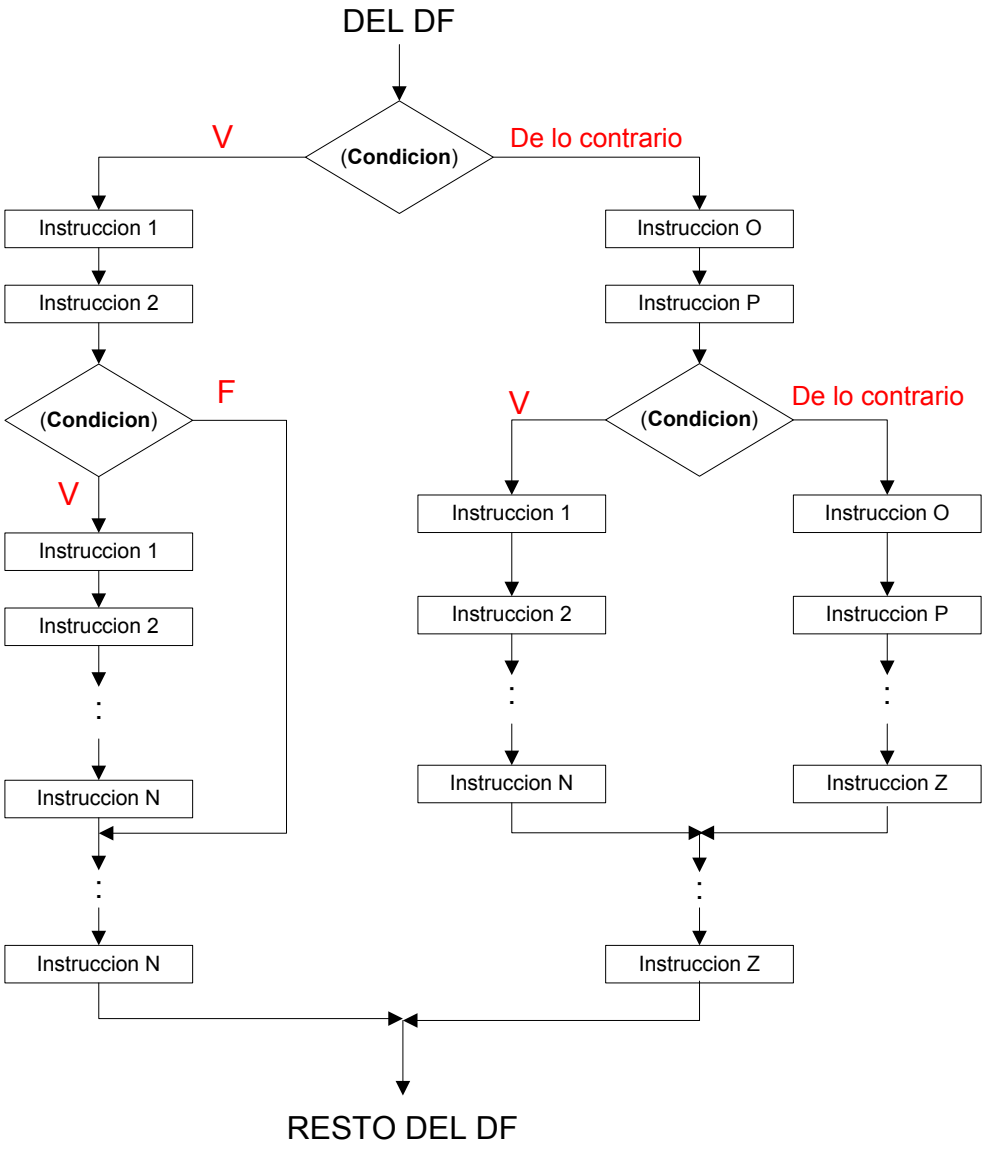


Ilustración 22. Estructura de selección anidada conformada por una estructura de selección simple y una estructura de selección compuesta dentro de una estructura de selección compuesta.

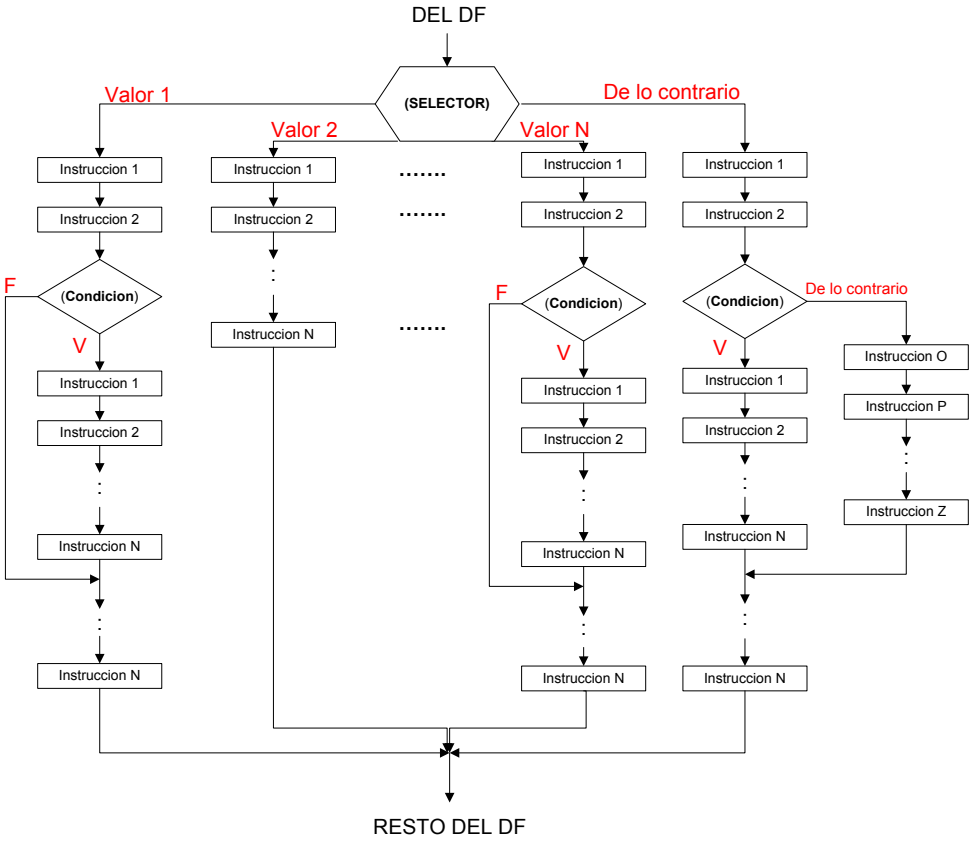


Ilustración 23. Estructura de selección anidada conformada por dos estructuras de selección simple, una estructura de selección compuesta dentro de una estructura de selección múltiple.

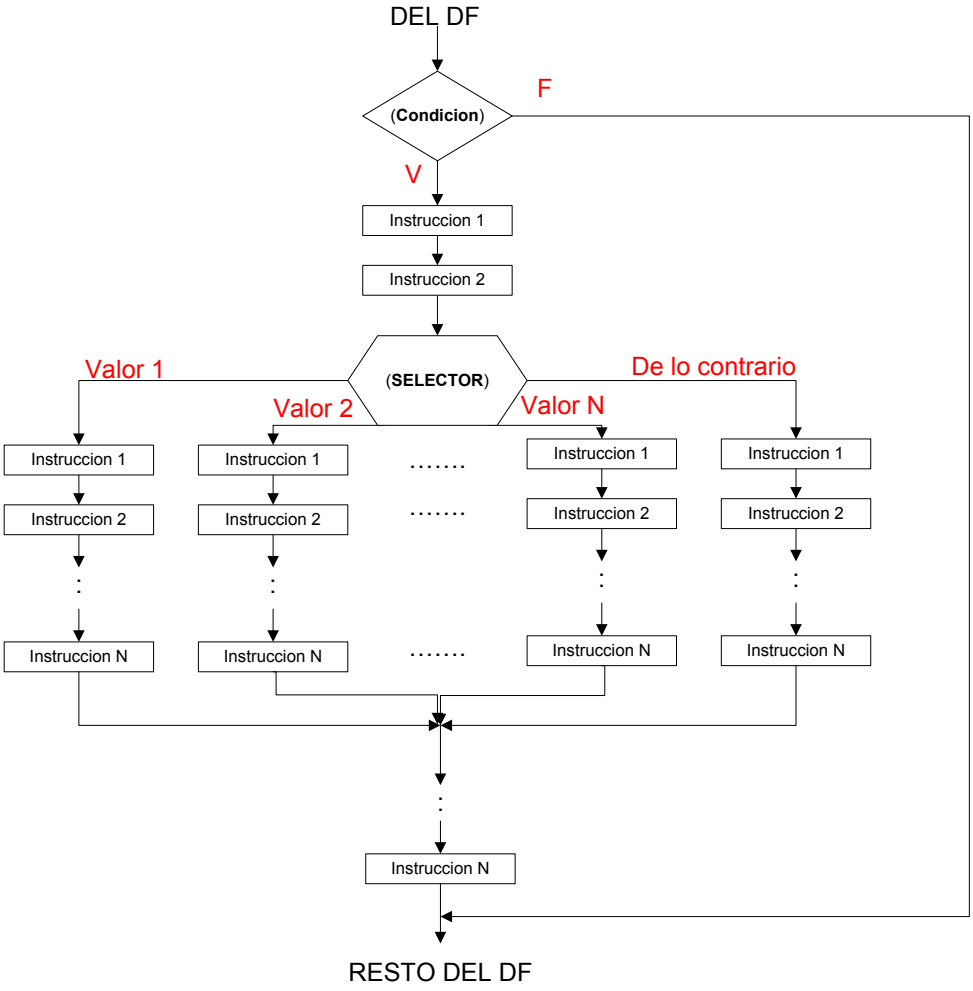


Ilustración 24. Estructura de selección anidada conformada por una estructura de selección múltiple dentro de una estructura de selección simple.

3.5 CONSTRUCCIÓN DE ESTRUCTURAS DE SELECCIÓN

A continuación se presentan una serie de ejemplos, a manera de guía, para ayudar a comprender al lector cuándo es necesario el uso de una estructura de selección y cómo construir su **condición**.

1. Si la palabra **SI** se encuentra en el enunciado y está vinculada directamente con el problema, inmediatamente se debe pensar en el uso de una estructura de selección. Algunos enunciados proveen de los elementos necesarios para construir la condición y éstos están vinculados directamente con la palabra **SI** y con el problema.

Construya un pseudocódigo tal que conocido el sueldo de un empleado, calcule el incremento del mismo basado en lo siguiente: **si** el *sueldo es inferior a $100000, aumente el 15%*; **si** el *sueldo está entre $100000 y $1.000.000, incluidos, aumente el 8%* y; **si** el *sueldo supera el $1.000.000, aumente el 3%*.

La frase donde se encuentra la palabra **SI** en el ejemplo nos muestra la **condición** y se puede construir teniendo en cuenta lo que dice la misma. Por ejemplo, "Si sueldo es inferior a 100000", indica que la **condición** es: Sueldo < 100000, y parte de la estructura de selección sería:

Si (Sueldo < 100000) **entonces**…

Las instrucciones que se colocan después del **entonces** son todas aquellas que deben ejecutar cuando la **condición** sea verdadera en este caso, el cálculo del aumento del sueldo.

El mismo procedimiento se puede aplicar con las otras frases para construir las demás condiciones.

2. Si en el enunciado no aparece la palabra **SI** y se deduce del mismo que el problema se resuelve haciendo comparaciones, entonces se debe hacer uso de una expresión de selección. En este caso, la **condición** contendrá los identificadores que contienen los números o valores a comparar.

Dados tres números enteros positivos diferentes construya un pseudocódigo que permita determinar cuál es el mayor de ellos.

Para resolver el problema se debe comparar cada número con los otros dos por lo cual se hace necesario el uso de una estructura de selección.

El enunciado comienza con "dados tres números…", lo cual significa que los genera el usuario y éstos deben guardarse en sus respectivas variables. Para construir la condición se usan los identificadores con el correspondiente operador relacional. Por ejemplo una **condición** podría ser:

Si (A > B Y A > C) **entonces**…

La anterior **condición** genera un resultado verdadero cuando el número almacenado en la variable A es mayor que los otros dos y la instrucción que se coloca después del **entonces** es la de mostrar en pantalla el número A como mayor.

A continuación se muestra el fragmento de pseudocódigo que contiene la estructura de selección completa:

```
Si ( A > B   Y   A > C) entonces
        Escribir (" El mayor es:", A)
De lo contrario
        Si ( B > A   Y  B > C) entonces
                Escribir (" El mayor es:", B)
        De lo contrario
                Escribir (" El mayor es:", C)
        Fin_si
Fin_si
```

3. Las estructuras de selección múltiple se emplean cuando el enunciado muestra que al evaluar una misma expresión o variable se generan diferentes valores y, a cada valor generado le corresponde la ejecución de una acción diferente. Sin embargo, cualquier estructura de selección podría ser empleada para el mismo fin, simplemente se obtendría una solución más extensa.

Construya un pseudocódigo que permita realizar operaciones aritméticas elementales, según sea la clave ingresada:

CLAVE	**OPERACIÓN**
+	SUMA
-	RESTA
*	MULTIPLICACION
/	DIVISION

Observe que en la variable llamada CLAVE se almacenará un dato tipo carácter y podrá contener cualquiera de los cuatro mostrados en la tabla. Por ejemplo, cuando CLAVE contenga símbolo **+** se debe realizar una suma. La **condición** para este ejemplo sería:

Si (CLAVE) **igual** …

Después del **igual** se colocan toda la gama de valores posibles que puede almacenar la variable CLAVE (+, -, *, /).

A continuación se muestran dos posibles soluciones para el mismo problema, usando estructuras de selección diferentes:

```
Si ( CLAVE) igual
 " + " :suma ← A + B
        Escribir ( " La operación", CLAVE,
        "da:", suma)
 " – " :   resta ← A - B
        Escribir ( " La operación", CLAVE,
        "da:", resta)
 " * " :   mult ← A * B
        Escribir("La operación", CLAVE,
        "da:", mult)
 " / " :   divi ← A / B
        Escribir("La operación", CLAVE,
        "da:", divi)
De lo contrario:Escribir("Clave no valida ")
Fin_si
```

```
Si ( CLAVE = " + ") entonces
  suma ← A + B
  Escribir ( " La operación", CLAVE, "da:",
             suma)
Fin_si
Si ( CLAVE = " – ") entonces
  resta ← A - B
  Escribir ( " La operación", CLAVE, "da:",
             resta)
Fin_si
Si ( CLAVE = " * ") entonces
  mult ← A * B
  Escribir ( " La operación", CLAVE, "da:",
             mult)
Fin_si
Si ( CLAVE = " / ") entonces
   divi ← A / B
   Escribir("La operación", CLAVE, "da:",
            divi)
De lo contrario
   Escribir  ( "Clave no valida ")
Fin_si
```

4. Si el problema se resuelve con la ayuda de expresiones aritméticas, se debe identificar qué valores pueden generar errores aritméticos que hagan que la expresión no pueda calcularse. Si se encuentra al menos un valor que genere error se debe incluir una estructura de selección.

- En el siguiente ejemplo, si la variable C toma el valor de cero, la operación no tiene solución, existe un error de división por cero.

$$\frac{A+B}{C}$$

Los valores de C para los cuales la operación aritmética no genera error son todos aquellos para los cuales se satisface C < > 0, la **condición** sería entonces:

Si (C < > 0) **entonces**

A continuación se muestra el fragmento de pseudocódigo correspondiente a la estructura de selección simple, que soluciona este problema.

Si (C < > 0) **entonces**
    Resultado ← (A + B) / C
    Escribir (" El valor de la expresión es:", Resultado)
Fin_si

En caso de querer mostrar un mensaje de error, se puede emplear una estructura de selección compuesta como muestra el siguiente fragmento de pseudocódigo:

Si (C < > 0) **entonces**
    Resultado ← (A + B) / C
    Escribir (" El valor de la expresión es:", Resultado)
De lo contrario
    Escribir (" El valor ingresado para C es incorrecto")
Fin_si

- Otro error son las raíces cuadradas de números negativos. La siguiente expresión aritmética posee variables dentro de una raíz cuadrada, si $B+3*C<0$, la raíz no tiene solución por tanto, se debe usar una estructura de selección.

$$\sqrt{(B+3*C)}$$

Los valores para los cuales la operación aritmética no genera error son todos aquellos que satisfacen $B+3*C>0$, parte de la estructura de selección sería:

Si ($B+3*C>0$) **entonces**…

Las instrucciones que se colocan después del **entonces**, son todas aquellas que se deben ejecutar cuando la **condición** genera un resultado **verdadero**, en los casos anteriores, la expresión matemática a resolver.
A continuación se muestra el fragmento de pseudocódigo que representa la estructura de selección completa:

Si ($B+3*C>0$) **entonces**
    Resultado ← ($B+3*C$) ** 0.5

Escribir (“ El valor de la expresión es:”, Resultado)
Fin_si

5. Consiste en encontrar todos los valores no deseados que puedan generar resultados con errores de interpretación. Son los más difíciles de encontrar y normalmente no se hallan explícitamente en los enunciados. Los valores que crean los casos críticos son generados por errores de digitación del usuario, por suposición de datos que nunca deben ser ingresados pero que pueden ser digitados, etc.

Los casos críticos implican que el programador debe pensar en lo “obvio” y en las “suposiciones” para así evitar que ellas aparezcan en algún momento dado de la ejecución del programa y generen error en la interpretación de los resultados obtenidos.

Si se encuentra al menos un caso crítico, se debe usar una estructura de selección y la **condición** estará formada por una expresión que contenga todos los valores posibles que generen error.

Conocido el sueldo de un empleado, construya un pseudocódigo que calcule e imprima su nuevo sueldo, sabiendo que si el sueldo del empleado es inferior a $100000 se le debe aumentar el 15%.

Dentro del enunciado se observa la **SI**, se debe construir una estructura de selección cuya condición sería la siguiente:

Sueldo < 100000

Al construir las expresiones aritméticas se comprueba que ninguna generaría error independientemente del valor del Sueldo.

NSueldo = Sueldo + Aumento
Aumento = Sueldo x 0.15

Sin embargo, un sueldo negativo generaría un error de interpretación. Se construye una estructura de selección cuya condición sería:

Si (Sueldo > = 0) **entonces**

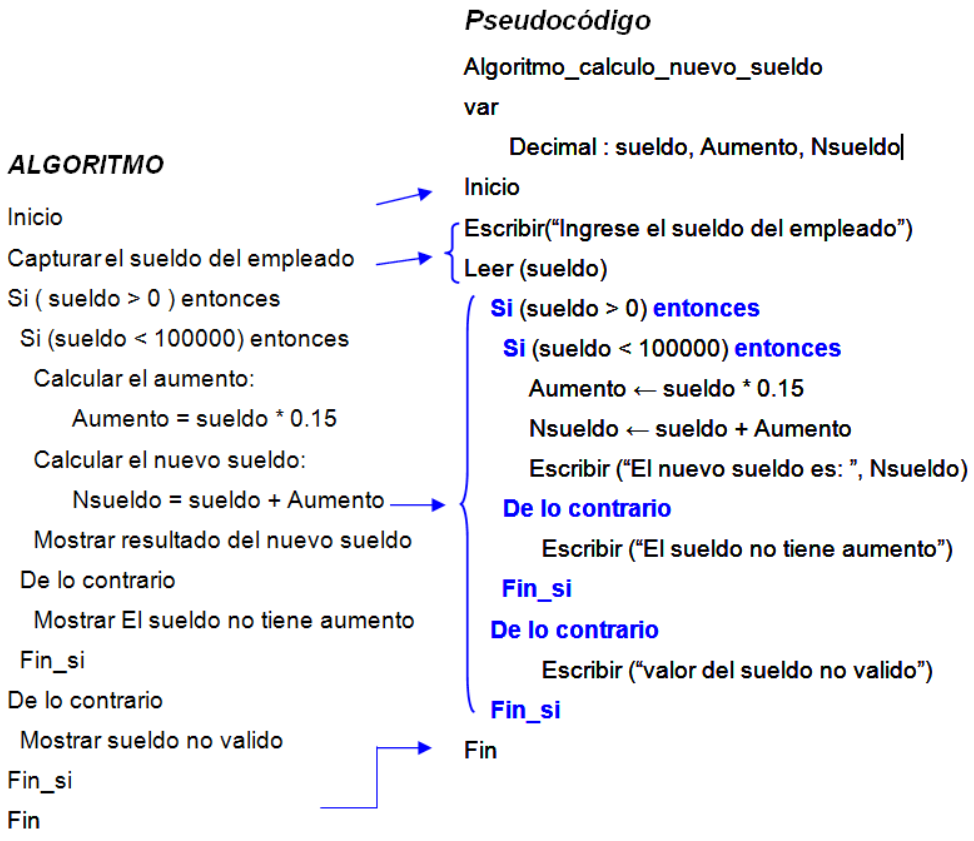
Normalmente si se encuentran casos críticos, las estructuras de selección deben colocarse inmediatamente después de haber sido capturado el dato que se

evaluará en la **condición**; en el ejemplo anterior, inmediatamente después de que el usuario ingrese el Sueldo.

A continuación se muestra el fragmento de pseudocódigo que representa la estructura de selección completa:

```
Si (Sueldo > = 0) entonces
        Si (Sueldo < 100000) entonces
                Aumento ← Sueldo * 0.15
                NSueldo ← Sueldo + Aumento
                Escribir (" El nuevo sueldo es:", NSueldo)
        Fin_si
Fin_si
```

El algoritmo completo y su equivalente en pseudocódigo quedarían como sigue:



La siguiente solución muestra un error común en la aplicación de las estructuras de selección.

Pseudocódigo –versión dos–

```
Algoritmo_calculo_nuevo_sueldo
var
    Decimal : sueldo, Aumento, Nsueldo
Inicio
Escribir("Ingrese el sueldo del empleado")
Leer (sueldo)
Si(sueldo>0 Y sueldo<100000) entonces
    Aumento ← sueldo * 0.15
    Nsueldo ← sueldo + Aumento
De lo contario
    Escribir("El sueldo no tiene aumento")
Fin_si
Escribir ("El nuevo sueldo es: ", Nsueldo)
Fin
```

La primera versión del pseudocódigo cumple con los requerimientos establecidos en el algoritmo, en la segunda versión se reúnen las dos condiciones en una sola. Cuando no se cumple esta condición, no se calcula Nsueldo por lo tanto, no habrá valor alguno para visualizar al ejecutarse la penúltima instrucción, lo cual genera un error.

En la siguiente Ilustración 25 se muestra el diagrama de flujo para este problema relacionándolo con el pseudocódigo por medio de flechas.

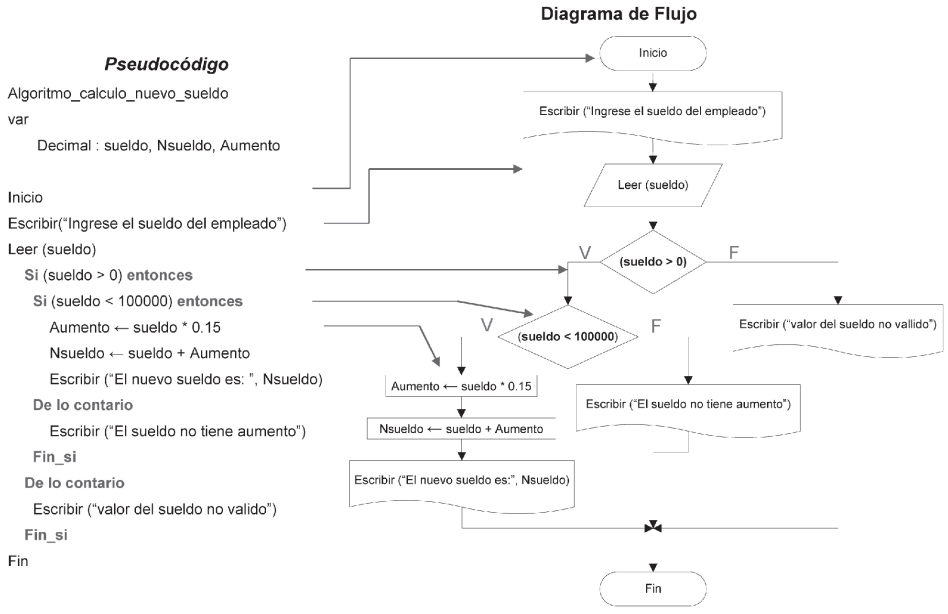


Ilustración 25. Relación entre pseudocódigo y diagrama de flujo usando estructuras de control.

3.6 EJERCICIOS

3.6.1 Ejercicios con Respuesta

1. Dados tres números enteros positivos diferentes construya un pseudocódigo que permita determinar cuál es el mayor de ellos.

2. Construya un pseudocódigo que permita realizar operaciones aritméticas elementales, según sea la clave ingresada:

CLAVE	OPERACIÓN
+	SUMA
-	RESTA
*	MULTIPLICACION
/	DIVISION

Cada operación requiere de dos operandos –números–. Muestre la clave ingresada y el resultado de la operación.

3. Una compañía dedicada al alquiler de automóviles cobra un monto fijo de $300 para los primeros 300 Km. de recorrido. Para más de 300 Km. y hasta 1000 Km., cobra un monto adicional de $15 por cada kilómetro en exceso sobre 300. Para más de 1000 Km. cobra un monto adicional de $10 por cada kilómetro en exceso sobre 1000. Los precios no incluyen el 16% del impuesto al valor agregado, IVA. Diseñe un pseudocódigo que determine el monto a pagar por el alquiler de un vehículo y el monto incluido del impuesto.

3.6.2 Ejercicios sin Respuesta

4. Dado cualquier par de números enteros construya un pseudocódigo que reste siempre el menor del mayor.

5. Construya un pseudocódigo que indique si un número es par o impar.

6. Si representamos los días de la semana con dígitos tendremos que el 1 corresponde al lunes, el 2 al martes y así sucesivamente. Dado un entero cualquiera, construya un pseudocódigo que muestre el nombre del día de la semana al que corresponde.

7. Construya un pseudocódigo que dada una letra cualquiera indique si se trata de una vocal o no.

8. Construya un pseudocódigo que redondee un número es decir, que convierta un número decimal a su equivalente entero.

9. Construya un pseudocódigo que dado dos valores **X** y **Y**, verifique si las siguientes ecuaciones arrojan el mismo resultado: a) **X** + 3**Y**
   b) 2**X** – 5**Y**

3.6.3 Respuesta a los Ejercicios

1. Dados tres números enteros positivos diferentes construya un pseudocódigo que permita determinar cuál es el mayor de ellos.

   Primero se determina si es necesario usar alguna estructura de selección. Aunque el enunciado no contiene la palabra SI, pide encontrar el mayor de tres números, para ello es necesario comparar un número con los dos restantes y, determinar si éste es o no mayor. Como se requiere el uso de comparaciones para resolver el problema entonces se debe usar una estructura de selección. Las **condiciones** de cada una de las estructuras es:

   Si (A > B Y A > C) **entonces**
   Si (B > A Y B > C) **entonces**
   Si (C > A Y C > B) **entonces**

   Se deben usar a lo máximo tres comparaciones (tres condiciones) una para cada número. En las variables A, B y C se guardaran los números ingresados por el usuario.

   No se necesitan estructuras de selección múltiple porque no existen más de dos caminos a escoger con una misma expresión. Y el ejercicio no exige el uso de operaciones aritméticas por lo no habrán errores de este tipo.

   El ejercicio específica "números enteros positivos" por lo que no son válidos los números negativos, la ***condición*** de la estructura quedaría:

   Si (A >= 0 Y B >= 0 Y C >=0) **entonces**

Además, el ejercicio exige hallar el mayor de tres números diferentes por lo que los números no son válidos si son iguales, se requiere de la siguiente estructura de selección:

Si (A <> B Y A <> C Y B <> C) **entonces**

Estas estructuras se colocan justo después de capturarse los tres números.

Para este ejercicio se mostraran dos versiones del algoritmo y de pseudocódigo, la primera usa estructuras de selección simples.

ALGORITMO usando ESS

```
Inicio
Capturar tres números enteros positivos.
Si (A>=0 Y B >= 0 Y C>=0) entonces
 Si (A<> B Y A<>C Y B<>C) entonces
  Si (A > B Y A > C) entonces
   Mostrar el mayor es A
  Fin_si
  Si (B > A Y B > C) entonces
   Mostrar el mayor es B
  Fin_si
  Si (C > A Y C > B) entonces
   Mostrar el mayor es C
  Fin_si
 Fin_si
 Si (A = B O A = C O B = C)entonces
  Mostrar existen números iguales
 Fin_si
Fin_si
Si (A < 0 O B < 0 O C < 0) entonces
    Mostrar numero no valido
Fin_si
Fin
```

Pseudocódigo usando ESS

```
Algoritmo_encuentra_mayor
var
     Entero : A,B,C
Inicio
 Escribir("Ingrese tres números enteros positivos
          diferentes")
 Leer (A,B,C)
 Si (A >= 0 Y B >= 0 Y C >= 0) entonces
    Si (A<>B Y A<>C Y B<>C) entonces
       Si (A > B Y A > C) entonces
          Escribir ("El numero mayor es:",A)
       Fin_si
       Si (B > A Y B > C) entonces
          Escribir ("El numero mayor es:",B)
       Fin_si
       Si (C > A Y C > B) entonces
          Escribir ("El numero mayor es:",C)
       Fin_si
    Fin_si
    Si ( A = B O A = C O B = C) entonces
       Escribir ("Existen números iguales")
    Fin_si
 Fin_si
 Si ( A < 0 O B < 0 O C < 0) entonces
       Escribir ("numero no valido")
 Fin_si
Fin
```

Se requieren de 7 estructuras de selección para resolver el problema, dos de ellas son utilizadas para informar al usuario que ocurre cuando ingresa un dato que no es valido (números negativos) ó cuando digita dos o tres números iguales. La ausencia de estas dos estructuras no afecta la solución del problema pero es importante incluirlas en la solución.

La segunda versión utiliza estructuras de selección compuesta. En la construcción del pseudocódigo el programador es libre de escoger la solución que utilizará, generalmente se usa aquella que tenga menos líneas de código.

ALGORITMO usando ESC

```
Inicio
Capturar tres números enteros positivos.
Si ( A>=0 Y B >= 0 Y C>=0 ) entonces
 Si (A<> B Y A<>C Y B<>C) entonces
  Si (A > B Y A > C) entonces
      Mostrar el mayor es A
  De lo contrario
    Si (B > A Y B > C) entonces
        Mostrar el mayor es B
    De lo contrario
        Mostrar el mayor es C
    Fin_si
  Fin_si
 De lo contrario
  Mostrar existen números iguales
 Fin_si
De lo contrario
   Mostrar numero no valido
Fin_si
Fin
```

Pseudocódigo usando ESC

```
Algoritmo_encuentra_mayor
var
     Entero : A,B,C
Inicio
 Escribir("Ingrese tres números enteros positivos
          diferentes")
 Leer (A,B,C)
   Si (A>=0 Y B>=0 Y C>=0) entonces
    Si (A<>B Y A<>C Y B<>C) entonces
     Si (A > B Y A > C) entonces
        Escribir ("El numero mayor es:",A)
     De lo contrario
      Si (B > A Y B > C) entonces
        Escribir ("El numero mayor es:",B)
      De lo contrario
        Escribir ("El numero mayor es:",C)
      Fin_si
     Fin_si
    De lo contario
        Escribir ("Existen números iguales")
    Fin_si
   De lo contario
        Escribir ("numero no valido")
   Fin_si
Fin
```

Comparando las dos soluciones se puede apreciar que aquella que utiliza estructuras de selección compuesta tiene ventajas por que usa menos líneas de código y, contiene tan sólo 4 estructuras de selección. Se puede aprovechar la zona del **Falso** para determinar cuándo el número mayor es el almacenado en la variable C, obsérvese que se llega a este **De lo contrario** cuando se han descartado las opciones anteriores.

Es importante recordar que un **Si** sólo puede tener un **De lo contrario** y debe tener un **Fin_si**, si esto no se tiene en cuenta se podría construir la solución de un problema utilizando más de un **De lo contrario** dentro de un solo **Si**.

A continuación la Ilustración 26 y la Ilustración 27 muestra la solución del problema usando diagramas de flujo.

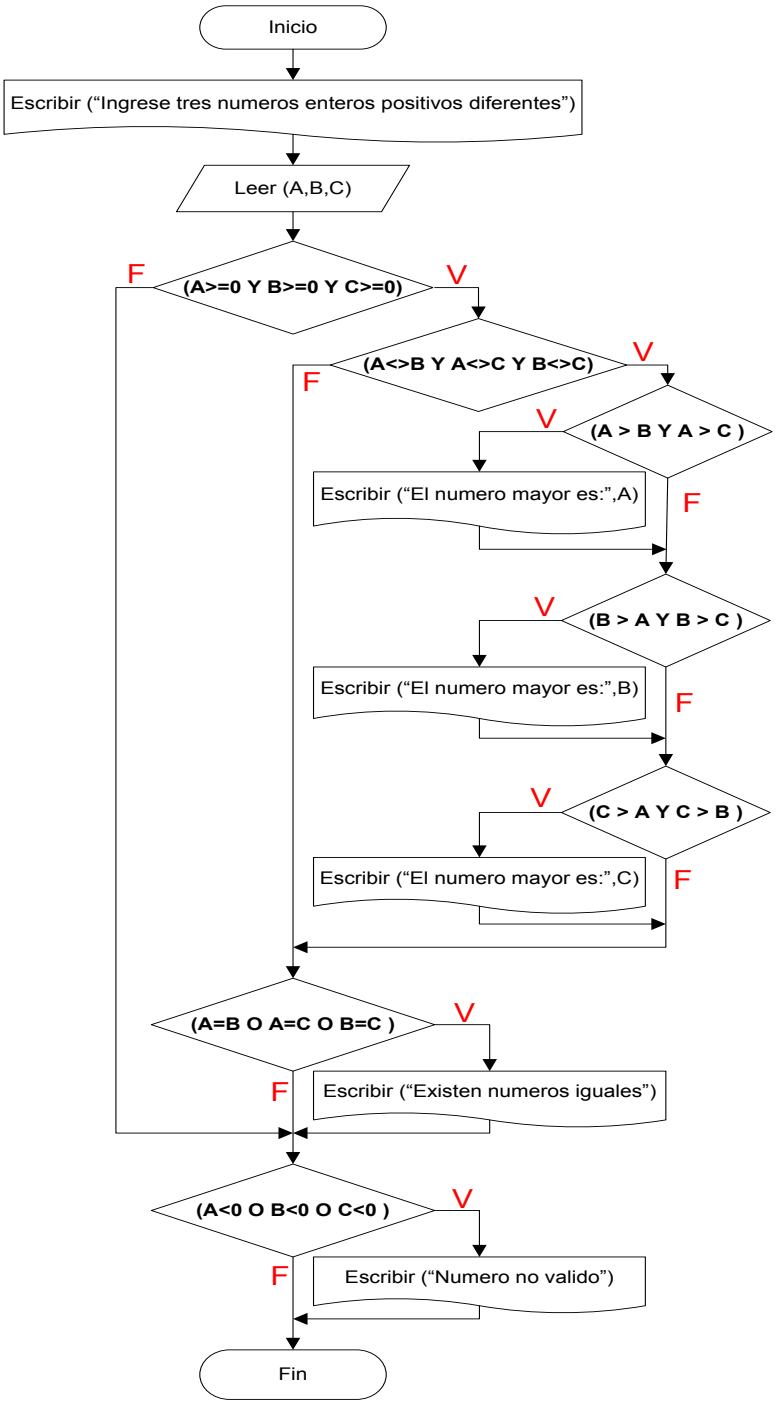


Ilustración 26. Diagrama de flujo usando estructuras de selección simples.

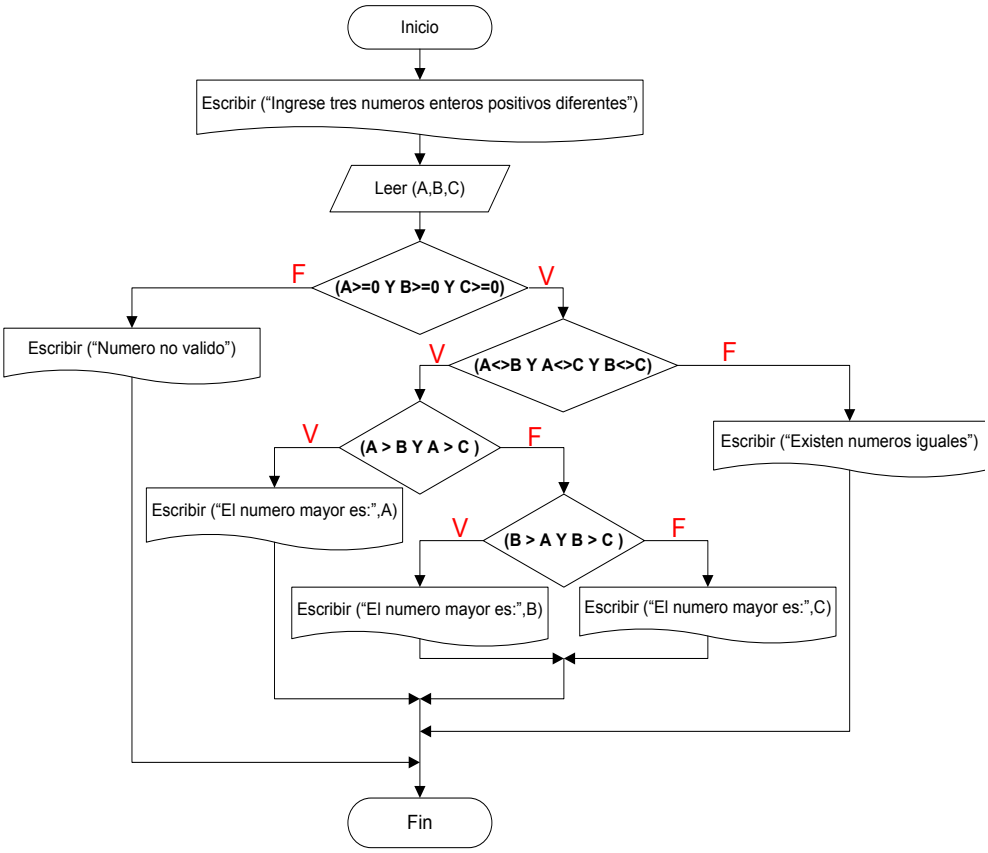


Ilustración 27. Diagrama de flujo estructuras de selección compuestas.

2. Construya un pseudocódigo que permita realizar operaciones aritméticas elementales, según sea la clave ingresada:

CLAVE	OPERACIÓN
+	SUMA
-	RESTA
*	MULTIPLICACION
/	DIVISION

Cada operación requiere de dos operandos –números–. Muestre la clave ingresada y el resultado de la operación.

El enunciado carece de la palabra SI sin embargo, para poder llevar a cabo la operación SUMA, es necesario determinar que la clave ingresada es el símbolo + y, para ello, se debe comparar el dato dado por el usuario con el símbolo correspondiente. Como se deben hacer comparaciones se escribe una condición, es decir:

Si (clave = "+") **entonces**
Si (clave = "–") **entonces**
Si (clave = "*") **entonces**
Si (clave = "/") **entonces**

Note que dependiendo de un solo dato ingresado por el usuario se pueden escoger uno de entre cuatro caminos (las cuatro operaciones) por lo que una estructura de selección múltiple podría ser más apropiada:

Si (clave) **igual**

"+" **:** operación suma
"–" **:** operación resta
"*" **:** operación multiplicación
"/" **:** operación división

Las operaciones para resolver el ejercicio son:

sum = A + B
res = A – B
mul = A * B
divi = A / B

Sólo en la operación de división se puede generar un error aritmético, si el usuario ingresa un cero en la variable B, la condición queda:

Si (B <> 0) **entonces**

En la ejecución del pseudocódigo el usuario puede ingresar cualquier número sea entero o decimal, además, las condiciones limitaran la solución a sólo cuatro claves conocidas por lo que una clave diferente no genera resultados que no se pueden interpretar.

Según el análisis anterior, se necesitan dos estructuras de selección para resolver el problema, una de ellas para impedir que se ingrese el número cero y otra, para permitir realizar una operación dependiendo de la clave ingresada por el usuario.

El problema se resolverá usando las dos opciones, primero con las estructuras de selección simple o compuestas y, luego con la estructura de selección múltiple. El lector apreciará ambas soluciones y podrá observar las diferencias en las mismas.

ALGORITMO usando ESS

```
Inicio
Capturar clave.
Si (clave = " + ") entonces
  Mostrar ingrese dos números
  Capturar dos números
  Calcular suma
    suma = A + B
  Mostrar el resultado de la operación
            y la clave
Fin_si
Si (clave = "– ") entonces
  Mostrar ingrese dos números
  Capturar dos números
  Calcular resta
      resta = A – B
  Mostrar el resultado de la operación
            y la clave
Fin_si
Si (clave = " * ") entonces
  Mostrar ingrese dos números
```

Pseudocódigo usando ESS

```
Algoritmo_encuentra_mayor
var
      Decimal : A,B, sum, res, mult, divi
      Caracter : clave
Inicio
 Escribir("Ingrese clave")
 Leer (clave)
 Si (clave = "+") entonces
     Escribir ("Ingrese dos números")
     Leer (A,B)
     sum ← A + B
     Escribir ("La clave", clave, "genera",sum)
 Fin_si
 Si (clave = "–") entonces
     Escribir ("Ingrese dos números")
     Leer (A,B)
     res ← A – B
     Escribir ("La clave", clave, "genera",res)
 Fin_si
 Si (clave = "*") entonces
     Escribir ("Ingrese dos números")
     Leer (A,B)
     mult ← A * B
```

```
  Mostrar ingrese dos números
  Capturar dos números
  Calcular multiplicación
     mult = A * B
  Mostrar el resultado de la operación
           y la clave
Fin_si
Si (clave = " / ") entonces
  Mostrar ingrese dos números
      Capturar dos números
      Si ( B < > 0) entonces
        Calcular división
           divi = A / B
           Mostrar el resultado de la
                    operación y la clave
      Fin_si
      Si (B = 0) entonces
         Mostrar Clave no valida
      Fin_si
    Fin_si
    Si(clave<>"+" O clave<>"–" 0 clave<>"*"
       O clave<>"/") entonces
       Mostrar clave no valida
    Fin_si
    Fin
      Escribir ("La clave", clave, "genera",mult)
Fin_si
Si (clave = "/") entonces
    Escribir ("Ingrese dos números")
    Leer (A,B)
    Si (B <> 0) entonces
       divi ← A / B
       Escribir ("La clave", clave, "genera",divi)
       Fin_si
       Si (B = 0) entonces
          Escribir ("Numero no valido")
       Fin_si
    Fin_si
    Si (clave<>" + " O clave<>" – " O clave<>" * "
       clave<>" / ") entonces
       Escribir ("Clave no valida")
    Fin_si
    Fin
```

A continuación la Ilustración 28 se muestra el Diagrama de Flujo respectivo.

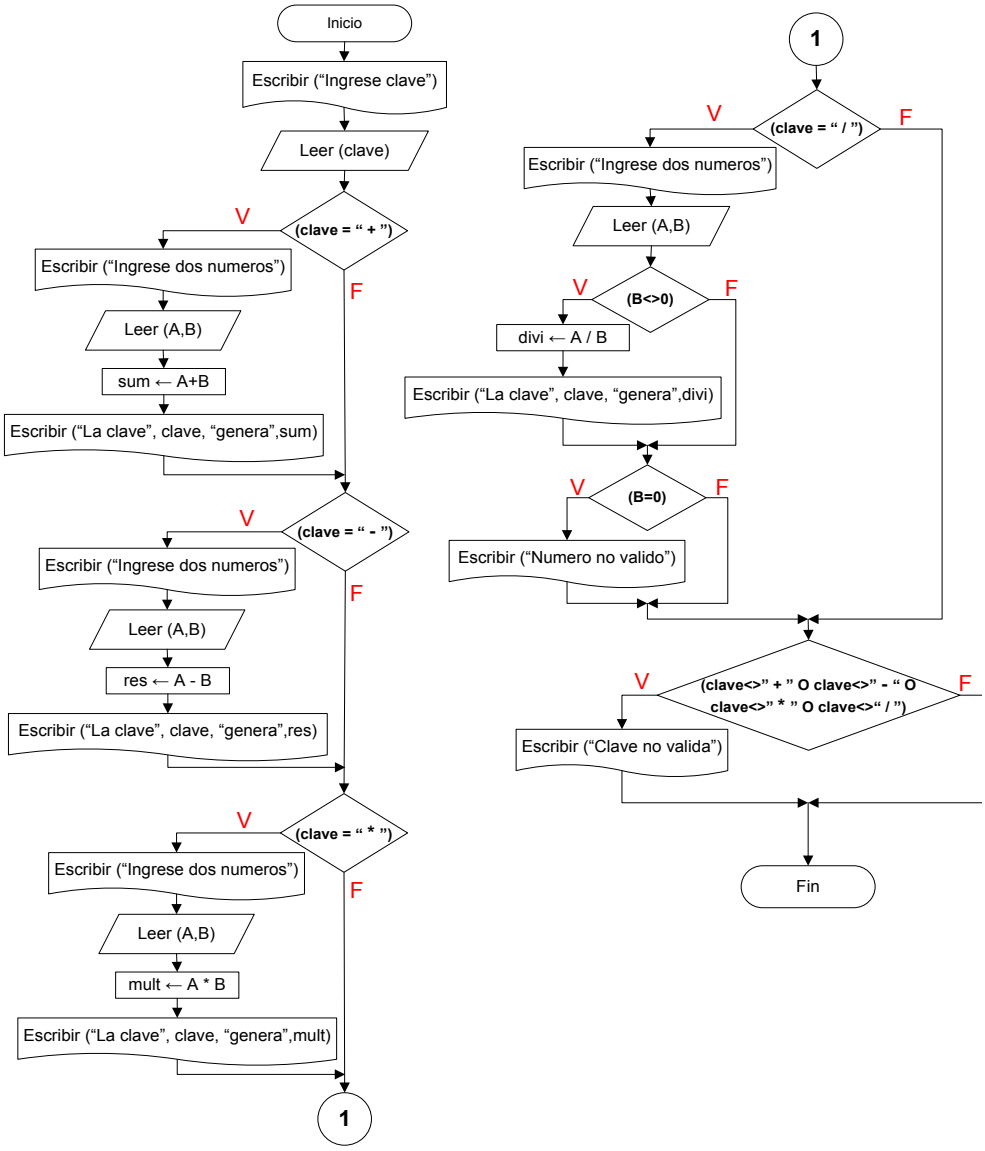


Ilustración 28. Solución utilizando estructuras de selección simple.

ALGORITMO usando ESC

```
Inicio
Capturar clave.
Si (clave = " + ") entonces
  Mostrar ingrese dos números
  Capturar dos números
  Calcular suma
    suma = A + B
  Mostrar el resultado de la operación
          y la clave
De lo contrario
  Si (clave = "– ") entonces
    Mostrar ingrese dos números
    Capturar dos números
    Calcular resta
      resta = A – B
    Mostrar el resultado de la operación
          y la clave
  De lo contrario
    Si (clave = " * ") entonces
      Mostrar ingrese dos números
      Capturar dos números
      Calcular multiplicación
        mult = A * B
      Mostrar el resultado de la
            operación y la clave
    De lo contrario
      Si (clave = " / ") entonces
        Mostrar ingrese dos números
```

Pseudocódigo usando ESC

```
Algoritmo_encuentra_mayor
var
    Decimal : A,B, sum, res, mult, divi
    Caracter : clave
Inicio
 Escribir("Ingrese clave")
 Leer (clave)
 Si (clave = "+") entonces
    Escribir ("Ingrese dos números")
    Leer (A,B)
    sum ← A + B
    Escribir ("La clave", clave, "genera",sum)
 De lo contrario
   Si (clave = "–") entonces
      Escribir ("Ingrese dos números")
      Leer (A,B)
      res ← A – B
      Escribir ("La clave", clave, "genera",res)
   De lo contrario
     Si (clave = "*") entonces
       Escribir ("Ingrese dos números")
       Leer (A,B)
       mult ← A * B
       Escribir ("La clave", clave, "genera",mult)
     De lo contrario
       Si (clave = "/") entonces
          Escribir ("Ingrese dos números")
          Leer (A,B)
          Si (B <> 0) entonces
             divi ← A / B
             Escribir ("La clave", clave, "genera",divi)
```

```
        Capturar dos números
        Si ( B < > 0) entonces
          Calcular división
          divi = A / B
          Mostrar el resultado de la
              operación y la clave
        De lo contrario
          Mostrar numero no valido
        Fin_si
      De lo contrario
        Mostrar clave no valida
      Fin_si
    Fin_si
  Fin_si
Fin_si
Fin
```

```
        De lo contrario
          Escribir ("Numero no valido")
        Fin_si
      De lo contrario
        Escribir ("clave no valida")
      Fin_si
    Fin_si
  Fin_si
Fin_si
Fin
```

A continuación la Ilustración 29 muestra el Diagrama de Flujo respectivo.

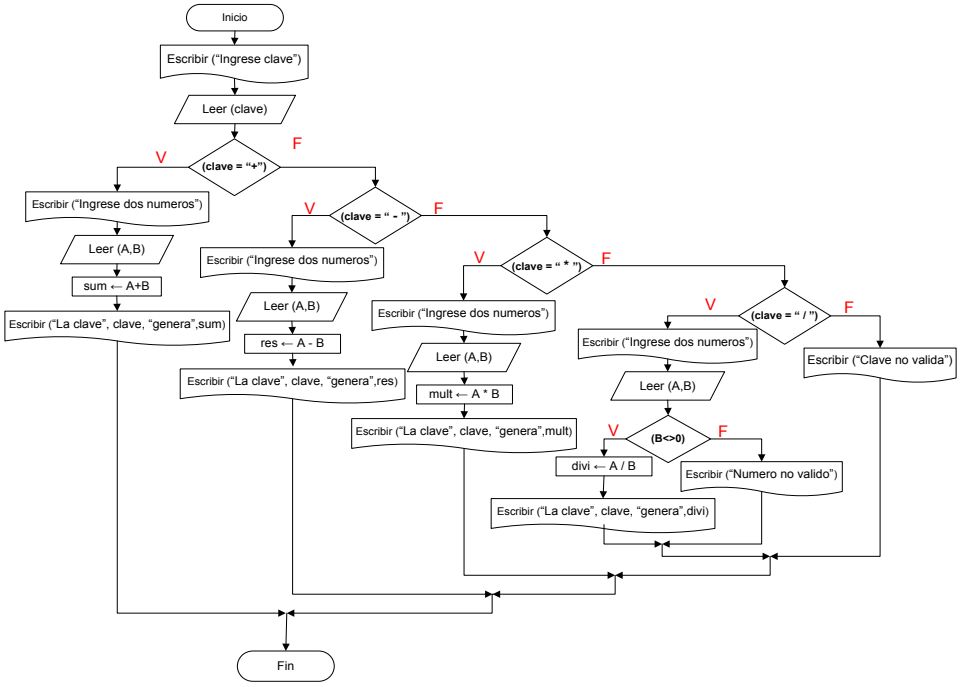


Ilustración 29. Solución utilizando estructuras de selección compuesta.

ALGORITMO usando ESM

```
Inicio
Capturar clave.
Si (clave ) igual
 " + " : Mostrar ingrese dos números
        Capturar dos números
        Calcular suma
   suma = A + B
        Mostrar el resultado de la
        operación y la clave
 "– " : Mostrar ingrese dos números
        Capturar dos números
        Calcular resta
        resta = A – B
        Mostrar el resultado de la
        operación y la clave
" * " : Mostrar ingrese dos números
        Capturar dos números
        Calcular multiplicación
          mult = A * B
        Mostrar el resultado de la
              operación y la clave
 " / " : Mostrar ingrese dos números
        Capturar dos números
        Si ( B < > 0) entonces
          Calcular división
          divi = A / B
          Mostrar el resultado de la
               operación y la clave
         De lo contrario
           Mostrar numero no valido
         Fin_si
  De lo contrario : Mostrar clave no valida
  Fin_si
  Fin
```

Pseudocódigo usando ESM

```
Algoritmo_encuentra_mayor
var
     Decimal : A,B, sum, res, mult, divi
     Caracter : clave
Inicio
 Escribir("Ingrese clave")
 Leer (clave)
 Si (clave) igual
          " + " :  Escribir ("Ingrese dos números")
                   Leer (A,B)
                   sum ← A + B
                   Escribir ("La clave", clave, "genera",sum)
          " – " :  Escribir ("Ingrese dos números")
                   Leer (A,B)
                   res ← A – B
                   Escribir ("La clave", clave, "genera",res)
          " * " :  Escribir ("Ingrese dos números")
                   Leer (A,B)
                   mult ← A * B
                   Escribir ("La clave", clave, "genera",mult)
          " / " :  Escribir ("Ingrese dos números")
                   Leer (A,B)
                   Si (B <> 0) entonces
                    divi ← A / B
                    Escribir ("La clave", clave, "genera",divi)
                   De lo contrario
                    Escribir ("Numero no valido")
                   Fin_si
De lo contrario : Escribir ("clave no valida")
Fin_si
Fin
```

El uso de una estructura de selección múltiple permite construir un programa más corto. La siguiente Ilustración 30 muestra el diagrama de flujo respectivo.

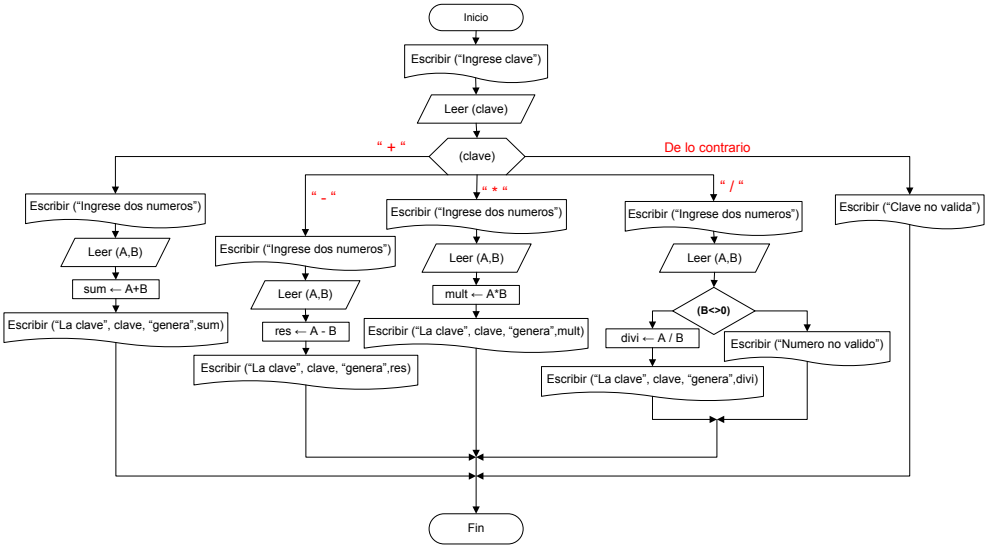


Ilustración 30. Diagrama de flujo usando estructura de selección múltiple.

3. Una compañía dedicada al alquiler de automóviles cobra un monto fijo de $300 para los primeros 300 Km. de recorrido. Para más de 300 Km. y hasta 1000 Km., cobra un monto adicional de $15 por cada kilómetro en exceso sobre 300. Para más de 1000 Km. cobra un monto adicional de $10 por cada kilómetro en exceso sobre 1000. Los precios no incluyen el 16% del impuesto al valor agregado, IVA. Diseñe un pseudocódigo que determine el monto a pagar por el alquiler de un vehículo y el monto incluido del impuesto.

El ejercicio exige determinar el monto a pagar por kilómetro recorrido dependiendo además del rango en que se encuentre según la distancia recorrida, es decir, si se utilizó el automóvil por 315 km se debe pagar por esta cantidad $300 por los primeros 300 km y $15 por cada kilómetro por encima de 300, o sea, $15 x 15 = $225, para dar un total de $300 + $225 = $525.

La siguiente Ilustración 31 muestra las ecuaciones que se deben usar para resolver el problema, las mismas dependen de la distancia recorrida.

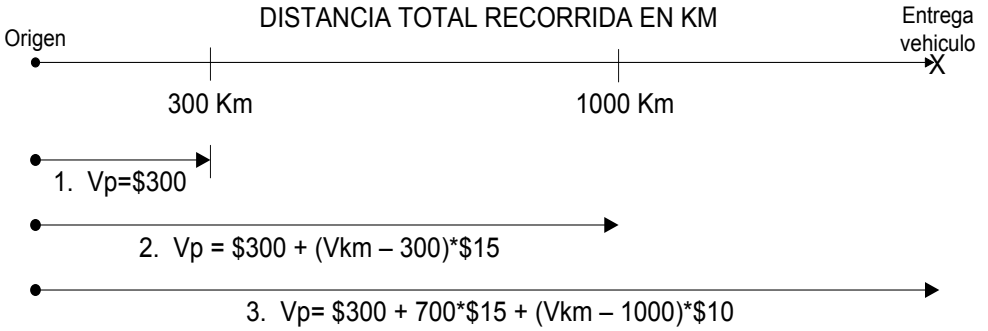


Ilustración 31. Análisis del ejercicio 3.

En la figura 3.24 **Vp** es la variable donde se almacena el valor a pagar y, **Vkm** es la variable que contiene el valor de los kilómetros recorridos. Note de la figura anterior que $525 se obtuvo de la ecuación 2, es decir:

$$Vp = 300 + (315 - 300) * 15$$
$$Vp = 525$$

Vkm se reemplazó por 315 que es la distancia del ejemplo.
La ecuación 1 muestra que un recorrido entre 0 y 300Km debe pagar un valor de $300. Cuando se supera la distancia de 300Km, en el análisis del valor total a pagar se tiene en cuenta los $300 por los primeros 300Km ya que este valor es fijo para este recorrido; el valor a pagar por encima de los 300Km (pero inferior a los 1000Km) dependerá del trayecto recorrido y para obtener ese valor se resta de la distancia los primeros 300Km pues éstos ya están incluidos en los $300, es decir:

$$Vp = 300 + (Vkm - 300) * 15$$

El primer número 300 representan el monto a pagar por la distancia recorrida de 300Km; el paréntesis refleja la distancia restante, por encima de los primeros 300Km, cuyo resultado se multiplica por 15 que es el valor a pagar por cada kilómetro en exceso sobre 300.

Utilizando un análisis similar se obtiene la ecuación 3 de la Figura 3.24, en donde 300 + 700*15 representa el monto a pagar por los primeros 1000Km y, (Vkm–1000)*10 simboliza el dinero a pagar por los kilómetros en exceso sobre 1000.

En la solución del problema, el programador deberá solicitar al usuario la distancia recorrida.

De la Figura 3.24 se puede apreciar que existen 3 maneras de obtener el monto a pagar por la distancia recorrida, el primero de ellos es usado cuando el vehículo ha recorrido una distancia entre 0 y 300Km, el segundo para recorridos entre 300 y 1000Km y el último para trayectos superiores a los 1000Km; para escoger cuál de ellas se debe utilizar es necesario construir estructuras de selección, quedando:

Si (Vkm <= 300) **entonces**
  Vp = 300

Si (Vkm >300 Y Vkm <= 1000) **entonces**
  Vp = 300 + (Vkm – 300)*15

Si (Vkm > 1000) **entonces**
  Vp = 300 + 700*15 + (Vkm – 1000)*10

Al ser Vkm la distancia recorrida por el automóvil, la misma no puede ser negativa por tanto es necesario una estructura de selección quedando:

Si (Vkm > 0) **entonces**

Ingresar un valor mayor a 0Km implica que el programa deba generar un mensaje de texto informando el monto a pagar por el alquiler del vehículo. 0Km no es un dato inválido pero tampoco se debe cancelar dinero alguno por el alquiler del automóvil. Para los valores que no cumplen la condición el programa debe informar al usuario del ingreso de datos inválidos.

Según el análisis anterior, se necesitan cuatro estructuras de selección para resolver el problema, una de ellas para impedir que se ingrese un número negativo y, las otras, para escoger la ecuación pertinente que permita obtener el valor a pagar por el usuario según la distancia recorrida. En cada condición el uso de operadores relacionales crea un rango de valores para la variable **Vkm** representando una gran cantidad de posibilidades para cada estructura de selección.

A continuación se elabora el algoritmo y el pseudocódigo. El problema se resuelve usando las dos opciones, primero con las estructuras de selección simples y luego con las estructuras de selección compuestas.

ALGORITMO usando ESS

```
Inicio
Capturar distancia recorrida.
Si (distancia recorrida>0) entonces
  Si (distancia recorrida < 300) entonces
      Vp = 300
  Fin_si
  Si (distancia recorrida >= 300 Y
    distancia recorrida < 1000) entonces
      Vp = 300 + (distancia recorrida –
        300)*15
  Fin_si
  Si (distancia recorrida >1000) entonces
      Vp = 300 + 700 *15 + (distancia
        recorrida –  300)*15
  Fin_si
  Mostrar el valor a pagar por el alquiler
Fin_si
Si(distancia recorrida < 0) entonces
    Mostrar dato no valido
Fin_si
Fin
```

Pseudocódigo usando ESS

```
Algoritmo_encuentra_mayor
var
     Decimal : Vkm, Vp
Inicio
 Escribir("Ingrese distancia recorrida")
 Leer (Vkm)
 Si (Vkm >0) entonces
    Si (Vkm <= 300) entonces
      Vp = 300
    Fin_si
    Si (Vkm > 300 Y Vkm <= 1000) entonces
      Vp = 300 + (Vkm – 300)*15
    Fin_si
    Si (Vkm > 1000) entonces
       Vp = 300 + 700 * 15 + (Vkm – 1000)*10
    Fin_si
    Escribir ("El valor a pagar por el alquiler es", Vp)
 Fin_si
 Si (Vkm <=0) entonces
    Escribir ("Dato no valido")
 Fin_si
Fin
```

ALGORITMO usando ESC

```
Inicio
Capturar distancia recorrida.
Si (distancia recorrida>0) entonces
 Si (distancia recorrida < 300) entonces
    Vp = 300
 De lo contrario
   Si (distancia recorrida >= 300 Y
    distancia recorrida < 1000) entonces
     Vp = 300 + (distancia recorrida –
      300)*15
   De lo contrario
     Vp = 300 + 700 *15 + (distancia
      recorrida –  300)*15
   Fin_si
 Fin_si
 Mostrar el valor a pagar por el alquiler
De lo contrario
 Mostrar dato no valido
Fin_si
Fin
```

Pseudocódigo usando ESC

```
Algoritmo_encuentra_mayor
var
     Decimal : Vkm, Vp
Inicio
 Escribir("Ingrese distancia recorrida")
 Leer (Vkm)
 Si (Vkm >0) entonces
    Si (Vkm <= 300) entonces
       Vp = 300
    De lo contrario
       Si (Vkm > 300 Y Vkm <= 1000) entonces
          Vp = 300 + (Vkm – 300)*15
       De lo contrario
          Vp = 300 + 700 * 15 + (Vkm – 1000)*10
       Fin_si
    Fin_si
    Escribir ("El valor a pagar por el alquiler es", Vp)
 De lo contrario
    Escribir ("Dato no valido")
 Fin_si
Fin
```

Las siguientes ilustraciones muestran los diagramas de flujo respectivos.

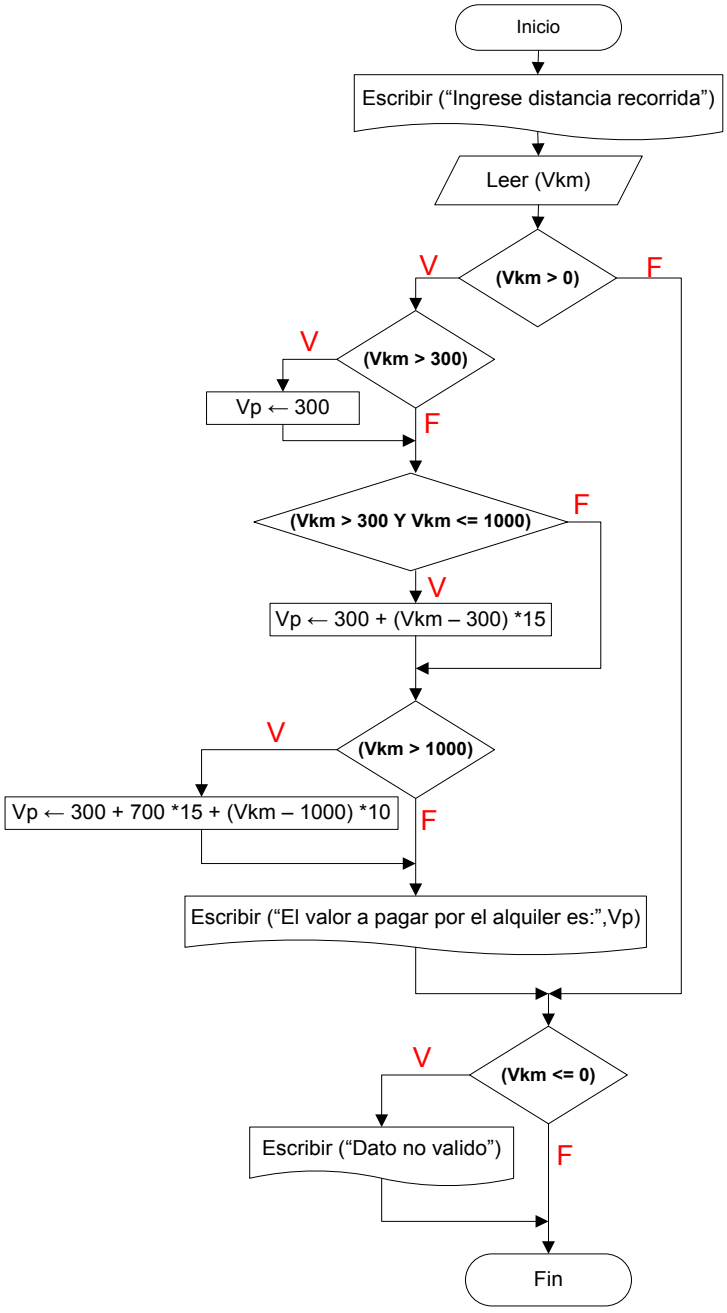


Ilustración 32. Diagrama de flujo usando estructuras de selección simples.

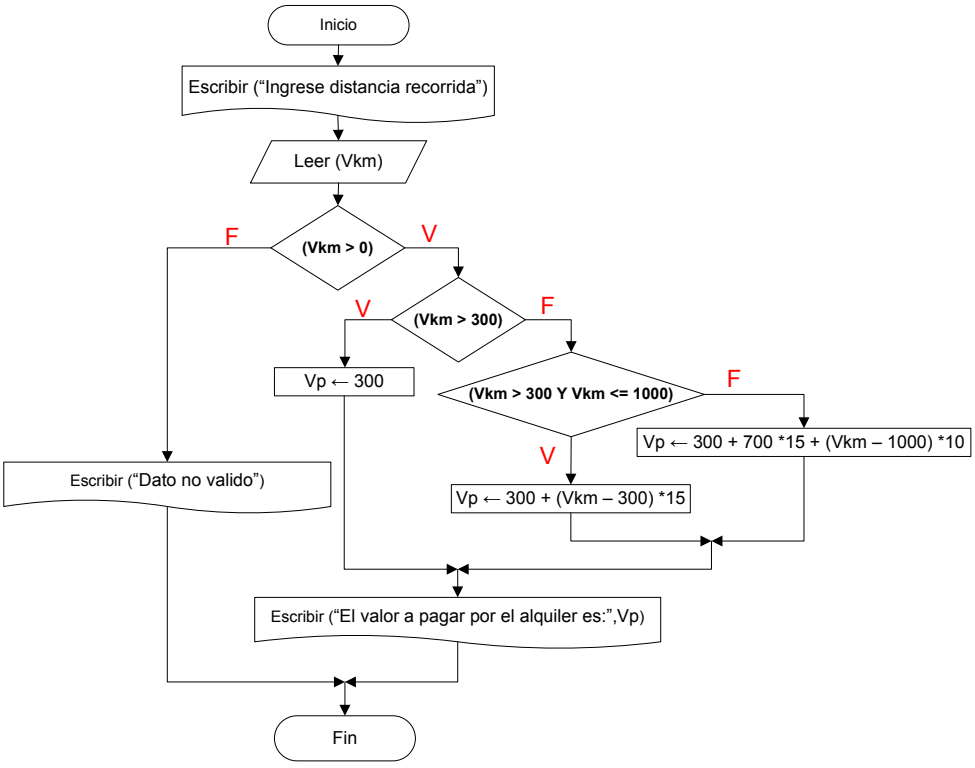


Ilustración 33. Diagrama de flujo usando estructuras de selección compuestas.

Sólo se requieren 3 símbolos de decisión para resolver el problema esto debido a la ventaja del uso de las estructuras de selección compuestas, la construcción del diagrama de flujo resulta más sencilla y corta.