Capítulo 7

Decisiones

Ya sabemos que una decisión, a nivel de lógica de programación, es la escogencia de uno de entre varios caminos lógicos dependientes todos de una condición. Normalmente algunos libros acotan en esta definición que se escoge uno de entre dos caminos lógicos y dado que todo el proceso del computador es binario podemos decir que esos libros también tienen la razón.

Por facilidades de representación se han esquematizado en los algoritmos (y así mismo en los Lenguajes de Programación) dos estructuras de decisión que son la *Estructura Si-Entonces-Sino* que es la que hemos estado utilizando en los ejemplos hechos hasta el momento y la *Estructura Casos* que nos permite realizar la escogencia de uno de entre varios ramales lógicos dependientes de una misma condición.

Estructura Si-Entonces-Sino

Esta es la estructura que hemos estado utilizando desde el comienzo de los algoritmos. Como ya se conoce podemos decir que su utilidad, fundamentalmente, es permitir que el computador escoja uno de dos ramales lógicos dependiendo de una determinada condición. Es importante anotar que tomar una decisión, por simple que ésta sea, le toma mucho tiempo al computador realizarla ya que aunque para nosotros es muy sencillo determinar si 9 es mayor que 5 para el computador no lo es pues debe realizar algunas operaciones para obtener la respuesta correcta.

De esta manera es útil saber que es mas eficiente un programa que tenga mas decisiones que otro que tenga menos toda vez que ambos busquen lograr el mismo objetivo. Teóricamente no hay mucho que decir acerca de las decisiones pero técnicamente es muy posible que usted llegue a encontrarse con algunos términos que desde ya es muy importante que conozca y que no son mas

que formas de reorganización del mismo esquema de decisión que hemos utilizado hasta el momento.

Decisiones Simples

Obedecen a la siguiente estructura

Si (Condición)
.
.
Instrucciones a ejecutar
En caso de que la condición sea Verdadera
.
.
Sino
.
.
Instrucciones a ejecutar
En caso de que la condición sea Falsa
.
.
Fin_Si

Como puede ver es la estructura mas sencilla para una toma de decisiones. Acerca de esta estructura podemos decir que no es obligatorio que cada que exista un condicional *Si* tenga que existir una alternativa *Sino* dado que no siempre es importante generar una determinada acción en el caso de que la condición sea Falsa. Normalmente es importante delimitar hasta donde llega toda la estructura de decisión y esa función la cumple el *Fin_Si* que aparece al final de ella. También vale la pena saber que en los Lenguajes de Programación estos delimitadores se pueden escribir con determinados signos establecidos por la sintaxis del mismo lenguaje.

No se olvide que en una estructura de decisión cuando se realizan las instrucciones por la parte Verdadera no se hacen las instrucciones por la parte Falsa y viceversa, es decir, cuando se realizan las instrucciones por la parte Falsa no se hacen las instrucciones por la parte verdadera.

Decisiones en Cascada

Este no es mas que un esquema en donde el *Sino* de cada *Si* condicional da inicio a un nuevo *Si* condicional y así sucesivamente. Su esquema general es el siguiente

Si Condición1
Instrucciones a ejecutar en caso de que
la condición1 sea Verdadera

```
Sino
        Si Condición2
                Instrucciones a ejecutar en caso de que
                la condición2 sea Verdadera
        Sino
                Si Condición3
                        Instrucciones a ejecutar en caso de que
                        la condición3 sea Verdadera
                Sino
                        Instrucciones a ejecutar en caso de que
                        la condición3 sea Falsa
```

Este es el esquema utilizado para el caso en el se dan 3 condiciones en cascada pero de acuerdo a las necesidades del algoritmo pueden ser más. Todo dependerá del objetivo que se quiera lograr. Para ilustrar un poco mejor la utilización de esta estructura veamos un ejemplo en donde sea necesaria.

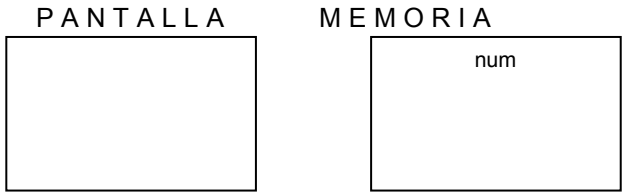
Ejemplo

Leer un número entero y determinar si es de uno o dos o tres o cuatro dígitos. Validar que el número no sea negativo.

```
Programa Decisión_en_Cascada
Var
        Entero  :       num
Inicio
        Escriba "Por favor, digite un número entero"
        Lea num

        Si num < 0
                num = num * ( -1 )

        Si num > = 1 y num < = 9
                Escriba " El número tiene 1 dígito "
        Sino
                Si num > = 10 y num < = 99
                        Escriba "El número tiene 2 dígitos"
                Sino
                        Si num > = 100 y num < = 999
                                Escriba "El número tiene 3 dígitos"
                        Sino
                                Si num > = 1000 y num < = 9999
                                        Escriba "El número tiene 4 dígitos"
                                Sino
                                        Escriba "El número tiene mas de 4 dígitos"

Fin
```

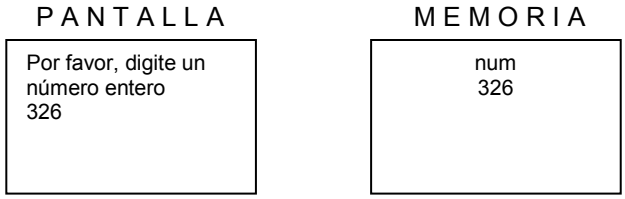
Vamos a realizarle una pequeña prueba de escritorio. En primera instancia la memoria inicia con una sola variable entera a la que hemos llamado *num*.

Programa Decisión_en_Cascada
Var
Entero : num



Continúa el algoritmo colocando un título en pantalla y leyendo un dato entero que será almacenado en la variable *num*. Vamos a asumir que el número leído es igual a 326.

Inicio
Escriba "Por favor, digite un número entero"
Lea num



Se realiza la primera validación y es verificar si el número es negativo. De ser así debe multiplicarse por el valor de (-1) para que se vuelva positivo. Como el valor recibido es 326 y por lo tanto la condición *Si num < 0* es falsa entonces se salta la orden *num = num * (-1)*

Si num < 0
*num = num * (-1)*

A continuación comienza a evaluar cuántos dígitos tiene el número leído. Para ello va realizando preguntas por rangos, sabiendo que el rango de los números de un dígito está entre 1 y 9, el rango de los números de dos dígitos está entre 10 y 99, el rango de los números de tres dígitos está entre 100 y 999, el rango de los números de 4 dígitos está entre 1000 y 9999 y sabiendo que por encima de 9999 por ahora solo nos interesa decir que hay números con mas de 4 dígitos (o sea 5 ó 6 ó 7 ó mas dígitos).

Ante la pregunta *Si num > = 1 y num < = 9* y sabiendo que la variable *num* contiene el número 326 puede decirse que se resuelve así

Si num > = 1 y num < = 9
V y F

Vemos pues que 326 es mayor que 1 pero no es menor que 9 por lo tanto la primera parte de la decisión es Verdadera y la segunda parte es Falsa por lo tanto acogiéndonos a la Tabla de Verdad del operador booleano **Y** vemos que *V y F* nos da Falso por lo tanto toda la decisión es Falsa. Por este motivo el computador se salta la orden *Escriba " El número tiene 1 dígito "* y continúa con el *Sino* de la primera decisión.

```
Si num > = 1 y num < = 9
        Escriba " El número tiene 1 dígito "
Sino
        Si num > = 10 y num < = 99
                Escriba "El número tiene 2 dígitos"
```

Claramente la orden dice *Si num > = 10 y num < = 99* y sabiendo que la variable num contiene 326 entonces la decisión se convierte en

```
Si 326 > = 10 y 326 < = 99
      V       y       F
```

Dado que 326 es mayor que 10 y que el mismo 326 no es menor que 99 y sabiendo que están unidas las dos condiciones por un operador booleano **Y** entonces toda la decisión es Falsa ya que así lo dice la tabla de verdad de este operador. Por lo tanto el computador se saltará la orden *Escriba "el número es de 3 dígitos"* y va a continuar con el *Sino* de la segunda decisión que da origen a una tercera condición.

```
Sino
        Si num > = 100 y num < = 999
                Escriba "El número tiene 3 dígitos"
```

En esta decisión se pregunta *Si num > = 100 y num < = 999* como sabemos que la variable num contiene 326 vemos que la decisión (internamente) se convierte en

```
Si num > = 100 y num < = 999
Si  326 > = 100 y  326 < = 999
      V       y       V
```

Con lo cual vemos que ambas condiciones son verdaderas por lo tanto toda la condición es Verdadera debido a que según la Tabla de Verdad del operador **Y** éste genera Verdadero solo si sus dos entradas son Verdaderas o sea solo si las condiciones que tiene a los lados son Verdaderas. Por lo tanto como esta condición es Verdadera entonces el computador ejecuta

Escriba "El número tiene 3 dígitos"

P A N T A L L A	M E M O R I A
Por favor, digite un número entero 326 El número tiene 3 dígitos	num 326

Como ejecuta esta instrucción por ser Verdadera la condición entonces no ejecuta ni evalúa lo que depende del Sino de esta condición. De tal manera que se “salta” lo siguiente

```
Sino
                    Si num > = 1000 y num < = 9999
                        Escriba “El número tiene 4 dígitos”
                    Sino
                        Escriba “El número tiene mas de 4 dígitos”
```

Llegando directamente hasta el finalizador del algoritmo que es lo último que ejecuta

Fin

Acogiéndonos a lo que en su momento habíamos indicado podemos mirar ahora el área de la pantalla y verificar si lo que aparece allí es conceptualmente cierto o no. Como vemos que es cierto entonces podemos decir que este algoritmo está bien al menos en lo que corresponde a la evaluación de números de tres dígitos. Realice ahora usted una prueba de escritorio con este algoritmo con cada uno de los siguiente números: - 5, 6498 y 32.

Usted puede notar que la utilización de Decisiones en cascada nos permite llegar de una manera sencilla y fácil a lograr un determinado objetivo en donde se involucren muchos condicionales inter-relacionados.

Decisiones en secuencia

Este es el tipo de estructura que se utiliza cuando se deben realizar varias preguntas en donde no es importante (normalmente) el *Sino* de cada decisión. Su esquema general es el siguiente

```
Si Condición1
    Instrucciones a ejecutar en caso de que
    La condición1 sea Verdadera
Si condición2
    Instrucciones a ejecutar en caso de que
    La condición2 sea Verdadera
Si condición3
    Instrucciones a ejecutar en caso de que
    La condición3 sea Verdadera
Si condición4
    Instrucciones a ejecutar en caso de que
    La condición4 sea Verdadera
```

No se olvide que pueden existir muchas mas condiciones. El esquema aquí presentado solamente muestra la estructura general para cuando sean 4 condiciones pero si se necesitan mas

simplemente se pueden utilizar y ya. Alguna de las condiciones puede tener su correspondiente *Sino*. Tenga en cuenta que si la ultima condición de un conjunto de Decisiones en Secuencia tiene Sino, éste solo se ejecutará en caso de que la última condición sea Falsa y no tendrá nada que ver con las demás condiciones. Al igual que el ejemplo anterior veamos un poco mas claro la utilización de la estructura con un ejemplo.

Ejemplo

Leer un número entero y determinar si es positivo o negativo o si es 0.

No olvide que para todos los enunciados aquí planteados, los algoritmos que se presentan son solo una versión de solución para dichos enunciados. Con esto quiero decirle una vez mas que si usted ha desarrollado por su cuenta los algoritmo aquí presentados y encuentra una solución diferente a lo que aparece en este libro entonces no se preocupe sencillamente realícele una prueba de escritorio a su versión y si cumple el objetivo entonces estará bien, no importa que sea diferente al algoritmo que aparezca en este libro. Esta es mi versión solución para este enunciado.

```
Programa Decisiones_en_Secuencia
Variables
        Entero  :       num
Inicio
        Escriba “Digite un número entero”
        Lea num

        Si num < 0
                Escriba “El número digitado es negativo”
        Si num > 0
                Escriba “El número digitado es positivo”
        Si num = 0
                Escriba “El número digitado es cero”
Fin
```

Sé que usted puede estar pensando que este algoritmo se pudo haber solucionado con Decisiones en Cascada pero tenga en cuenta que la razón por la que utilizamos aquí Decisiones en Secuencia es que sabemos con toda seguridad que un número cualquiera que sea solo podrá tener tres estados (si se le puede llamar así): Que sea menor que cero o sea negativo, que sea mayor que cero o sea positivo ó que sea igual a cero. Esa certeza es la que nos permite utilizar esta estructura de decisión y no la anterior.

También puede estar pensando que en vez de haber planteado el algoritmo así

```
        Si num < 0
                Escriba “El número digitado es negativo”
        Si num > 0
                Escriba “El número digitado es positivo”
        Si num = 0
```

Escriba "El número digitado es cero"

Se pudo haber planteado así

Si num < 0
Escriba "El número digitado es negativo"
Si num > 0
Escriba "El número digitado es positivo"
Sino
Escriba "El número digitado es cero"

Pero si realizáramos la prueba de escritorio con el valor –5 por ejemplo para la variable *num* veríamos que cuando pregunte *Si num < 0* la respuesta sería verdadera y por lo tanto ejecutaría *Escriba "El número digitado es negativo"* dejando en pantalla el título

El número digitado es negativo

Pero, siguiendo con la prueba de escritorio, cuando se realice la pregunta *Si num > 0* la respuesta tendrá que ser Falsa y entonces se ejecutará el *Sino* de esta decisión o sea *Escriba "El número digitado es cero"* lo cual nos dejaría en pantalla los títulos

El número digitado es negativo
El número digitado es cero

Que son títulos contradictorios pues un número no puede ser negativo y ser cero al mismo tiempo. Por esta razón la buena utilización de Decisiones en Secuencia nos va a permitir no solo lograr los objetivos sino, además, lograrlos de una buena forma técnica.

Decisiones anidadas

Éstas se presentan en aquellos casos en los cuales una estructura completa de decisión se encuentra dentro de otra. Su esquema general es el siguiente

Si Condicion_Externa
.
.
Si Condición_Interna
Instrucciones a ejecutar en caso de que
La condición interna sea Verdadera
Sino
Instrucciones a ejecutar en caso de que
La condición interna sea Falsa

```
        Fin_Si_Interno
        .
        .
Sino
        .
        .
        Instrucciones a ejecutar en caso de que
        La condición externa sea Falsa
        .
        .
Fin_Si_Externo
```

O también se puede encontrar de la siguiente forma

```
Si Condicion_Externa
        .
        .
        Instrucciones a ejecutar en caso de que
        La condición externa sea Verdadera
        .
        .
Sino
        .
        .
        .
        Si Condición_Interna
                Instrucciones a ejecutar en caso de que
                La condición interna sea Verdadera
        Sino
                Instrucciones a ejecutar en caso de que
                La condición interna sea Falsa
        Fin_Si_Interno
        .
        .
Fin_Si_Externo
```

En este caso podemos ver que en uno de los dos ramales lógicos de una estructura de decisión completa se encuentra otra estructura de decisión completa. Los puntos suspensivos colocados en la estructura representan que pueden existir mas instrucciones. Veamos su utilización con un ejemplo.

Ejemplo

Leer un número entero y determinar si es positivo o negativo. Si es positivo determinar si tiene dos dígitos y si es negativo determinar si tiene tres dígitos. Asumir que no puede entrar el número cero.

```
Programa Decisiones_Anidadas
Variables
        Entero  :       n
Inicio
        Escriba “Digite un número entero”
        Lea num

        Si num > 0
                Si num >= 10 y num <= 99
                        Escriba “El número es positivo y tiene dos dígitos”
                Sino
                        Escriba “El número es positivo y no tiene dos dígitos”
                Fin_Si
        Sino
                Si num >= - 999 y num <= -100
                        Escriba “El número es negativo y tiene tres dígitos”
                Sino
                        Escriba “El número es negativo y no tiene tres dígitos”
                Fin_Si
        Fin_Si
```

Realícele dos pruebas de escritorio a este algoritmo, la primera asumiendo que el número leído es –52 y la segunda asumiendo que es 1650. Tenga mucho cuidado con la ejecución de las decisiones que aquí se plantean.

Estructura Casos

Esta estructura permite abreviar una serie de Decisiones en cascada o en secuencia. La inmensa mayoría de lenguajes de programación tiene una instrucción equivalente para su implementación y precisamente lo que se busca con esta estructura es facilitar la toma de decisiones por parte del computador dado que éste es el proceso que mas tiempo le toma. La estructura casos toma el contenido de una variable y lo evalúa acorde con unos posibles valores ejecutando lo que se le indique en cada una de las opciones.

Estructura Casos Simple

Su estructura general es la siguiente:

```
Evalúe (variable)

        Si vale (valor_1) :     Instrucciones a ejecutar en caso de que la variable
                                sea igual a valor_1
```

```
        Si vale (valor_2) :     Instrucciones a ejecutar en caso de que la variable
                                sea igual a valor_2

        Si vale (valor_3) :     Instrucciones a ejecutar en caso de que la variable
                                sea igual a valor_3

        Si vale (valor_4) :     Instrucciones a ejecutar en caso de que la variable
                                sea igual a valor_4
              .                         .
              .                         .
              .                         .
        Si vale (valor_n) :     Instrucciones a ejecutar en caso de que la variable
                                Sea igual a valor_n

        Sino              :     Instrucciones a ejecutar en caso de que la variable no sea
                                Igual a ninguno de los valores posibles (o sea valor_1, valor_2,
                                valor_3, valor_4,....., valor_n)
Fin_Evalúe
```

Su forma de ejecución es muy sencilla. Al iniciar esta estructura el computador recibe la variable con un determinado contenido (o por lo menos dentro de nuestros algoritmos nos debemos asegurar de ello), verifica si el contenido de la variable es igual a uno de los posibles valores que se hayan considerado dentro de la estructura. De ser así, ejecuta las órdenes que acompañen el valor determinado. Si el contenido de la variable no es igual a ninguno de los contenidos, entonces ejecuta las instrucciones que acompañan el *Sino* de esta estructura.

Veámoslo con un ejemplo para una mejor clarificación de su uso y al tiempo haremos una comparación con una secuencia de Decisiones en cascada para que se vea su diferencia.

Ejemplo

Leer un número entero y si es de un dígito y además es menor que 5 escribir su nombre en pantalla (El nombre del 1 es UNO, el nombre del 2 es DOS, etc.).

Versión 1.0 Algoritmo solución con Decisiones

```
Programa Nom_Digito
Variables
        Entero  :       n
Inicio
        Escriba "Digite un número entero"
        Lea n

        Si n < 0
                n = n * ( -1 )

        Si n = 1
```

```
                Escriba “Uno”
        Sino
                Si n = 2
                        Escriba “Dos”
                Sino
                        Si n = 3
                                Escriba “Tres”
                        Sino
                                Si n = 4
                                        Escriba “Cuatro”
                                Sino
                                        Si n = 5
                                                Escriba “Cinco”
                                        Sino    Escriba “El número es mayor que cinco”
                                        Fin_Si
                                Fin_Si
                        Fin_Si
                Fin_Si
        Fin_Si
Fin
```

Algunas precisiones oportunas acerca de esta primera solución del problema planteado:

a. Los indicadores Fin_Si corresponden a cada una de las Decisiones que inicialmente se colocaron. No son obligatorios pero es muy útil cuando se escriben sobre todo al momento de codificar el algoritmo.

b. La indentación del algoritmo (o sea ese hecho de que cada conjunto o bloque de instrucciones comienza una columnas mas allá) es muy útil para la claridad del programa y de manera muy especial, al igual que el numeral anterior, cuando se va a codificar el programa.

c. Esta versión incluye una serie de decisiones en cascada y por lo tanto realizándole una prueba de escritorio podremos notar que logra el objetivo planteado. Sin embargo la utilización de la Estructura Casos nos permite escribir el mismo algoritmo de una manera mas entendible.

Versión 2.0 Algoritmo solución con Estructura Casos

```
Programa Nom_Digito
Variables
        Entero  :       n
Inicio
        Escriba “Digite un número entero”
        Lea n

        Si n < 0
                n = n * ( -1 )

        Evalúe ( n )
                Si vale 1 :     Escriba “Uno”
                Si vale 2 :     Escriba “Dos”
                Si vale 3 :     Escriba “Tres”
                Si vale 4 :     Escriba “Cuatro”
```

```
            Si vale 5 :     Escriba "Cinco"
            Sino    :       Escriba "El número es mayor que cinco"
        Fin_Evalúe
Fin
```

En esta solución, utilizando la Estructura Casos, usted puede notar que:

a. Este algoritmo logra el mismo objetivo que el algoritmo anterior solo que de una manera técnica mas apropiada.
b. La presentación del algoritmo es mucho mas entendible y precisamente eso lo hace mucho mas fácil de codificar que la anterior versión.
c. El *Sino* que aparece al final de la Estructura Casos se ejecuta en caso de que el contenido de la variable *n* no se igual ni a 1 ni a 2 ni a 3 ni a 4 ni a 5.
d. La indentación vuelve a ser útil y para ello quiero mostrarle dentro de este numeral cómo se verían ambas versiones sin indentación:

Versión 1.0 Algoritmo solución con Decisiones sin indentación

```
Programa Nom_Digito
Variables
        Entero  :       n
Inicio
Escriba "Digite un número entero"
Lea n

Si n < 0
n = n * ( -1 )

Si n = 1
Escriba "Uno"
Sino
Si n = 2
Escriba "Dos"
Sino
Si n = 3
Escriba "Tres"
Sino
Si n = 4
Escriba "Cuatro"
Sino
Si n = 5
Escriba "Cinco"
Sino    Escriba "El número es mayor que cinco"
Fin_Si
Fin_Si
Fin_Si
Fin_Si
Fin_Si
Fin
```

Versión 2.0 Algoritmo solución con Estructura Casos sin indentación

```
Programa Nom_Digito
Variables
        Entero  :       n
Inicio
Escriba "Digite un número entero"
Lea n

Si n < 0
n = n * ( -1 )

Evalúe ( n )
Si vale 1 :     Escriba "Uno"
Si vale 2 :     Escriba "Dos"
Si vale 3 :     Escriba "Tres"
Si vale 4 :     Escriba "Cuatro"
Si vale 5 :     Escriba "Cinco"
Sino      :     Escriba "El número es mayor que cinco"
Fin_Evalúe
Fin
```

No me podrá negar usted que los algoritmos vistos sin indentación (o sea sin estética) son mínimamente mas complejos de entenderlos que si se presentan de una manera técnica mas apropiada. Recuerde que la utilidad grande de la indentación se refleja al momento de codificar los algoritmos en algún lenguaje de programación pues allí el hecho de que su algoritmo esté bien y que la prueba de escritorio le haya arrojado unos resultados muy confiables puede llegar a truncarse por una mala codificación o peor aún por una mala agrupación de instrucciones. No está de más recordarle que cuando se agrupan las instrucciones como no son, los resultados del algoritmo pueden ser completamente diferentes.

Estructuras casos anidadas

Esta estructura se utiliza cuando una de las opciones de la estructura casos general da origen a otra estructura casos y otro conjunto de instrucciones. Veamos un ejemplo de esto:

Ejemplo

Leer un entero y si es igual a cualquier dígito comprendido entre 1 y 5 escribir su nombre. Si es igual a cinco además de escribir su nombre leer otro dígito y, si este último está entre 1 y 5, escribir

su componente decimal. Si entró un 3 entonces escribir “Cincuenta y Tres”, si entró un 1 entonces escribir “Cincuenta y Uno”.

```
Programa Casos_Anidados
Variables
        Entero  :       num, dig
Inicio
        Escriba “Digite un número entero”
        Lea num

        Si num < 0
                num = num * ( -1 )

        Evalúe ( num )
                Si vale 1:      Escriba “Uno”
                Si vale 2:      Escriba “Dos”
                Si vale 3:      Escriba “Tres”
                Si vale 4:      Escriba “Cuatro”
                Si vale 5:      Escriba “Cinco”
                                Escriba “Digite otro número entero”
                                Lea dig
                                Evalúe ( dig )
                                        Si vale 1:      Escriba “Cincuenta y Uno”
                                        Si vale 2:      Escriba “Cincuenta y Dos”
                                        Si vale 3:      Escriba “Cincuenta y Tres”
                                        Si vale 4:      Escriba “Cincuenta y Cuatro”
                                        Si vale 5:      Escriba “Cincuenta y Cinco”
                                        Sino    :       Escriba “El número es mayor que 5”
                                Fin_Evalúe
                Sino    :       Escriba “El número es mayor que cinco”
        Fin_Evalúe
Fin
```

Acerca de este algoritmo es poco lo que se puede aclarar pues como usted puede ver es bastante claro sin embargo debe tener en cuenta que cada *Evalúe* tiene su correspondiente *Sino* (y solo uno) y su correspondiente *Fin_Evalúe*. No se olvide que para la buena utilización de esta estructura es muy importante que usted sea una persona organizada y ordenada y verá como programar se vuelve un verdadero paseo.

Ejercicios

Algunas posibles soluciones a los siguientes ejercicios se pueden encontrar en el Libro *Algoritmos* del mismo autor.

1. Leer un número entero y determinar si es un número terminado en 4.

2. Leer un número entero y determinar si tiene 3 dígitos.

3. Leer un número entero y determinar si es negativo.

4. Leer un número entero de dos dígitos y determinar a cuánto es igual la suma de sus dígitos.

5. Leer un número entero de dos dígitos y determinar si ambos dígitos son pares.

6. Leer un número entero de dos dígitos menor que 20 y determinar si es primo.

7. Leer un número entero de dos dígitos y determinar si es primo y además si es negativo.

8. Leer un número entero de dos dígitos y determinar si sus dos dígitos son primos.

9. Leer un número entero de dos dígitos y determinar si un dígito es múltiplo del otro.

10. Leer un número entero de dos dígitos y determinar si los dos dígitos son iguales.

11. Leer dos números enteros y determinar cuál es el mayor.

12. Leer dos números enteros de dos dígitos y determinar si tienen dígitos comunes.

13. Leer dos números enteros de dos dígitos y determinar si la suma de los dos números origina un número par.

14. Leer dos números enteros de dos dígitos y determinar a cuánto es igual la suma de todos los dígitos.

15. Leer un número entero de tres dígitos y determinar a cuánto es igual la suma de sus dígitos.

16. Leer un número entero de tres dígitos y determinar si al menos dos de sus tres dígitos son iguales.

17. Leer un número entero de tres dígitos y determinar en qué posición está el mayor dígito.

18. Leer un número entero de tres dígitos y determinar si algún dígito es múltiplo de los otros.

19. Leer tres números enteros y determinar cuál es el mayor. Usar solamente dos variables.
20. Leer tres números enteros y mostrarlos ascendentemente.

21. Leer tres números enteros de dos dígitos cada uno y determinar en cuál de ellos se encuentra el mayor dígito.

22. Leer un número entero de tres dígitos y determinar si el primer dígito es igual al último.

23. Leer un número entero de tres dígitos y determinar cuántos dígitos primos tiene.

24. Leer un número entero de tres dígitos y determinar cuántos dígitos pares tiene.

25. Leer un número entero de tres dígitos y determinar si alguno de sus dígitos es igual a la suma de los otros dos.

26. Leer un número entero de cuatro dígitos y determinar a cuanto es igual la suma de sus dígitos.

27. Leer un número entero de cuatro dígitos y determinar cuántos dígitos pares tiene.

28. Leer un número entero menor que 50 y positivo y determinar si es un número primo.

29. Leer un número entero de cinco dígitos y determinar si es un número capicúo. Ej. 15651, 59895.

30. Leer un número entero de cuatro dígitos y determinar si el segundo dígito es igual al penúltimo.

31. Leer un número entero y determina si es igual a 10.

32. Leer un número entero y determinar si es múltiplo de 7.

33. Leer un número entero y determinar si termina en 7.

34. Leer un número entero menor que mil y determinar cuántos dígitos tiene.

35. Leer un número entero de dos dígitos, guardar cada dígito en una variable diferente y luego mostrarlas en pantalla.

36. Leer un número entero de 4 dígitos y determinar si tiene mas dígitos pares o impares.

37. Leer dos números enteros y determinar cuál es múltiplo de cuál.

38. Leer tres números enteros y determinar si el último dígito de los tres números es igual.

39. Leer tres números enteros y determina si el penúltimo dígito de los tres números es igual.

40. Leer dos números enteros y si la diferencia entre los dos es menor o igual a 10 entonces mostrar en pantalla todos los enteros comprendidos entre el menor y el mayor de los números leídos.

41. Leer dos números enteros y determinar si la diferencia entre los dos es un número primo.

42. Leer dos números enteros y determinar si la diferencia entre los dos es un número par.

43. Leer dos números enteros y determinar si la diferencia entre los dos es un número divisor exacto de alguno de los dos números.

44. Leer un número entero de 4 dígitos y determinar si el primer dígito es múltiplo de alguno de los otros dígitos.

45. Leer un número entero de 2 dígitos y si es par mostrar en pantalla la suma de sus dígitos, si es primo y menor que 10 mostrar en pantalla su último dígito y si es múltiplo de 5 y menor que 30 mostrar en pantalla el primer dígito.

46. Leer un número entero de 2 dígitos y si terminar en 1 mostrar en pantalla su primer dígito, si termina en 2 mostrar en pantalla la suma de sus dígitos y si termina en 3 mostrar en pantalla el producto de sus dos dígitos.

47. Leer dos números enteros y si la diferencia entre los dos números es par mostrar en pantalla la suma de los dígitos de los números, si dicha diferencia es un número primo menor que 10 entonces mostrar en pantalla el producto de los dos números y si la diferencia entre ellos terminar en 4 mostrar en pantalla todos los dígitos por separado.

48. Leer un número entero y si es menor que 100 determinar si es primo.

49. Leer un número entero y si es múltiplo de 4 determinar si su último dígito es primo.

50. Leer un número entero y si es múltiplo de 4 mostrar en pantalla su mitad, si es múltiplo de 5 mostrar en pantalla su cuadrado y si es múltiplo e 6 mostrar en pantalla su primer dígito. Asumir que el número no es mayor que 100.