Capítulo 8

Ciclos

Ya hemos utilizado no solo esta palabra sino una estructura asociada que nos permite representar un conjunto de instrucciones que debe repetirse una cantidad determinada de veces, normalmente, dependiente de una condición. Los ciclos nos van a permitir iterar todo un proceso tantas veces como nosotros (ó el usuario) lo determinemos.

Concepto General

Un ciclo puede definirse como una estructura que nos permite repetir o iterar un conjunto de instrucciones y que tiene las siguientes características:

a. El conjunto de instrucciones debe ser finito
b. La cantidad de veces que se repita dicho conjunto de instrucciones también debe ser finita. En algunos casos esta cantidad de veces va a depender de una condición explícita y en otros casos va a depender de una condición implícita. Una condición es explícita cuando depende solamente de la misma ejecución del programa sin que sea importante la participación del usuario. Asimismo una condición es implícita cuando depende solamente de la voluntad del usuario y por lo tanto la cantidad de iteraciones o repeticiones del ciclo podría llegar a ser diferente cada vez pues sería posible que cambiara con cada usuario.
c. Deben estar claramente demarcados el inicio y el fin del ciclo. En los casos en los cuales solo exista una instrucción a iterar, no serán necesarios dichas marcas.
d. Dentro de un ciclo podrá ir cualquiera de las otras estructuras que se han estudiado incluyendo otros ciclos.

Vamos a desarrollar un ejemplo sin ciclos para notar la gran utilidad de estructurar nuestros algoritmos con ciclos.

Ejemplo

Mostrar los números del 1 al 100 de 1 en 1.

Versión Ineficiente No.1

```
Programa Vers_Inef_1
Inicio
        Escriba “   1 “
        Escriba “   2 “
        Escriba “   3 “
        Escriba “   4 “
        Escriba “   5 “
        Escriba “   6 “
        Escriba “   7 “
        Escriba “   8 “
        Escriba “   9 “
        Escriba “ 10 “
        Escriba “ 11 “
        Escriba “ 12“
        Escriba “ 13 “
        Escriba “ 14 “
        Escriba “ 15 “
        Escriba “ 16 “
        Escriba “ 17 “
                .
                .
                .
                .
                .
        Escriba “ 98 “
        Escriba “ 99 “
        Escriba “100“
Fin
```

Como puede ver en esta versión no se han utilizado variables y los puntos suspensivos representan toda esa cantidad de instrucciones que hacen falta pues en total serían 100 instrucciones *Escriba*. Es evidente que se logra el objetivo planteado pero que tal que en vez de ir el enunciado hasta 100 fuera hasta 1000 o fuera hasta 10000. Nuestro algoritmo se convertiría no solo en una cantidad ineficiente de instrucciones sino que además por cada vez que existiera una modificación prácticamente tendría que existir un algoritmo diferente pues tendríamos que adicionarle mas y mas líneas de órdenes. Veamos a continuación otra forma ineficiente de solucionar este mismo problema sin utilizar ciclos.

Versión Ineficiente No. 2

```
Programa Vers_Inef_2
Variables
        Entero  :       N
Inicio
        N = 1
        Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
                .
                .
                .
                .
        Si  N < = 100
                N = N + 1
                Escriba N
        Si  N < = 100
                N = N + 1
                Escriba N
Fin
```

Como puede ver, tendríamos que escribir 99 veces el esquema

```
Si  N < = 100
        N = N + 1
        Escriba N
```

Para poder lograr el objetivo con lo cual esta segunda versión ineficiente resultaría ser mucho mas larga que la anterior y, dada la gran cantidad de decisiones que debe tomar el computador, sería a su vez mas ineficiente. Lo que si podemos hacer es tomar el esquema repetitivo de esta última versión y escribirlo dentro de un ciclo que controle que se repita dicho esquema hasta cuando se hayan escrito todos los números enteros de 1 a 100 de 1 en 1,

Cómo lo haremos...? Pues muy sencillo. Simplemente note que tenemos que decirle al computador que inicie una variable en 1 y que mientras el contenido de esta variable sea menor o igual que 100 que escriba su contenido y que lo incremente en 1. De esta manera el contenido de la variable irá de 1 en 1 desde 1 hasta 100 escribiendo cada vez que cambie de número es decir cumpliendo el objetivo planteado. Note usted que acabo de decirle el algoritmo informal ahora todo lo que tenemos que hacer es llevarlo a un algoritmo técnico para que posteriormente sea fácil codificarlo en un Lenguaje de Programación. De manera que la siguiente es la versión técnica eficiente de la solución al enunciado planteado

Versión Eficiente con Ciclos

```
Programa Nums_1_100
Variables
        Entero :N
Inicio
        N = 1
        Mientras N < = 100
                Escriba N
                N = N + 1
        Fin_mientras
Fin
```

Evidentemente el algoritmo así presentado es mucho mas claro. No se puede negar que tanto ésta solución como las demás soluciones, a pesar de ser ineficientes, también cumplen con el objetivo planteado. Acerca de esta versión podemos hacer algunas reflexiones

a. Es mucho mas fácil de codificar en un determinado Lenguaje de Programación pues es una solución mucho mas compacta
b. Es mucho mas entendible y por lo tanto es mucho mas fácil de concebir como solución
c. No necesita muchas líneas de código para lograr el objetivo
d. Cuando necesiten *mostrar los números del 1 al 10000* todo lo que tenemos que hacer es cambiar el número *100* que aparece al inicio del ciclo por el número *10000* y el mismo algoritmo funcionará bien. No tendremos necesidad de adicionar líneas de código.
e. Cuando necesiten *mostrar los números del 45 al 951* simplemente tendremos que reemplazar la línea que dice *N = 1* por *N = 45* y en donde aparece el número *100* cambiarlo por el número *951* y el algoritmo logrará bien este nuevo objetivo.
f. Finalmente la solución planteado con ciclos al enunciado *Mostrar los números del 1 al 100 de 1 en 1* se ha convertido en una solución mucho mas genérica que cambiando los valores tope

del mismo ciclo será *Mostrar todos los enteros comprendidos entre dos números asumiendo que el primer número es el menor y el segundo es el mayor*.

Con estas reflexiones podemos justificar plenamente la utilización de ciclos dentro de un algoritmo y, en lo posible, buscar hacer un uso muy eficiente de esta estructura.

Tipos de Ciclos

Solo para facilitar la escritura de algunos algoritmos y con el ánimo de que desde el balcón de la lógica de programación se puedan tener mas herramientas que faciliten la estructuración de los ciclos, la mayoría de los lenguajes de programación tienen tres formas de presentación de los ciclos, ellas son:

a. Ciclo Mientras
b. Ciclo Para
c. Ciclo Haga Hasta

Precisamente vamos a revisar la estructura de construcción de cada uno de los ciclos tal como son concebidos por la mayoría de lenguajes de programación y posteriormente los utilizaremos para representar el mismo algoritmo con cada una de las estructuras.

a.- Ciclo Mientras

Este es el ciclo que hemos utilizado desde que comenzamos a hablar de algoritmos. Es el esquema general de trabajo para todos los ciclos, esto quiere decir que si usted entiende claramente la lógica de funcionamiento de este ciclo se le va a facilitar entender no solo los otros ciclos que aquí se explican sino cualquier otro ciclo que encuentre en algún otro libro de Lógica de Programación. Es útil que sepa que este ciclo también es llamado en algunos libros el *Ciclo Mientras Que* pero su filosofía es la misma del *Ciclo Mientras* que vamos a explicar aquí y que hemos venido utilizando.

Su estructura general es la siguiente

```
Mientras Condición
        .
        .
        Cuerpo del Ciclo
        .
        .
Fin_Mientras
```

Su forma de ejecución (textualmente explicada) es muy sencilla*: Mientras se cumpla que la condición sea Verdadera entonces se ejecutará el Cuerpo del Ciclo*. De manera que también podríamos decir que el Cuerpo del Ciclo se repetirá tantas veces como lo permita la condición o mientras dicha condición sea Verdadera. En condiciones normales la cantidad de veces que se repita el cuerpo del ciclo será siempre una cantidad finita y deberá existir, dentro del mismo cuerpo del ciclo, una o mas instrucciones que nos permitan aproximarnos a la condición o sea que propendan porque en algún momento la condición sea Falsa.

b.- Ciclo Para

El ciclo Para tiene la siguiente estructura

```
Para Var = tope_inicial hasta tope_final Paso Valor
        .
        .
        Cuerpo del Ciclo
        .
        .
Fin_Para
```

En este ciclo su forma de ejecución es la siguiente : *Var* representa una variable que va a tomar valores iniciando en *tope_inicial* y terminando en *tope_final* avanzando con un *Paso de Valor*. En los casos en los que no se especifica el valor del paso la mayoría de los lenguajes de programación asume el incremente de 1. El *Cuerpo del Ciclo* se ejecutará una vez por cada valor que tome la variable *Var.* Veamos con un ejemplo cuál sería la aplicación de este ciclo.

Ejemplo

Escribir los números impares comprendidos entre 1 y 20.

Es evidente que este ejemplo lo podemos desarrollar usando el ciclo mientras pero para efectos didácticos lo vamos a desarrollar usando el ciclo Para.

```
Programa Ejem_Ciclo_Para
Variables
        Entero :Num
Inicio
        Para Num = 1 hasta 20 Paso 2
                Escriba Num
        Fin_Para
Fin
```

Puede usted notar que este algoritmo es muy breve gracias a la presencia del *ciclo Para* en su contexto ya que si se hubiera estructurado utilizando el ciclo Mientras, una versión de solución hubiera podido ser la siguiente

```
Programa Ejem_Ciclo_Mientras
Variables
        Entero :Num
Inicio
        Num = 1
        Mientras Num < = 20
                Escriba Num
                Num = Num + 2
        Fin_Mientras
Fin
```

Ambas versiones logran el mismo objetivo lo cual significa que ambas versiones son correctas. Es importante anotar que dentro de lo normal cada ciclo siempre va a tener una variable que es la que almacena el valor de inicio del ciclo, es la que va a estar presente en la evaluación de la condición y es la que se incrementa para que en algún momento la condición sea Falsa. Es evidente que esta variable es muy importante por ello éste tipo de variables de ha caracterizado con el nombre de Indice del Ciclo. Podríamos decir que el índice del ciclo es la variable que permite la ejecución del cuerpo del ciclo. Un ciclo puede llegar a tener varios índices al tiempo.

Como los índices no son mas que variables entonces varios ciclos pueden tener el mismo índice siempre que se utilice éste en un ciclo solo hasta cuando haya terminado la ejecución del ciclo anterior.

c.- Ciclo Haga Hasta

Esta es otra de las otra de las formas que traen algunos de los lenguajes de programación para expresar un ciclo. Su estructura general es la siguiente

```
Haga
        .
        .
        .
        Cuerpo del Ciclo
        .
        .
        .
Hasta Condición
```

En este ciclo el Cuerpo del mismo se va a ejecutar hasta cuando se cumpla una condición esto quiere decir que el conjunto de instrucciones que conforman el cuerpo del ciclo se va a repetir mientras la evaluación de la condición sea Falsa. Es un ciclo muy parecido al Ciclo Mientras con la diferencia de que en éste la instrucciones se repiten Mientras la condición sea Falsa y no verdadera como sería en el Ciclo Mientras.

d. Ciclo Haga Mientras

Muy parecido al esquema anterior, algunos lenguajes de programación cuentan con esta otra estructura para representar un ciclo

Haga
.
.
.
Cuerpo del Ciclo
.
.
.
Mientras Condición

Podría decirse que esta es una inversión de la estructura del ciclo mientras. En este ciclo el cuerpo del mismo se repite mientras la condición sea Verdadera y su única diferencia con el *ciclo Mientras* es que en el *Ciclo Haga Mientras* primero se ejecuta el cuerpo del ciclo y luego se evalúa la condición en cambio en el *ciclo Mientras* primero se evalúa la condición y luego se ejecuta el cuerpo del ciclo.

Ejemplos Usando Todas Las Estructuras De Ciclos

1. Leer un número entero y determinar cuántos dígitos tiene

Es útil recordar que los datos enteros se manejan con aritmética entera, concepto que será muy útil para la concepción de este algoritmo ya que la cantidad de dígitos que tiene un número entero es igual a la cantidad de veces que se pueda dividir el número entre 10 sin que el cociente sea cero. Entonces lo que vamos a hacer en este algoritmo es leer un número, inicializar una variable (que actuará como contador_de_dígitos), dividir progresivamente el número entre 10 hasta cuando sea igual a cero y por cada vez que se divida entre 10 vamos a incrementar el contenido de la variable contador_de_dígitos en 1. De esta manera cuando el número haya llegado a cero tendremos en la variable contador_de_dígitos la cantidad de dígitos que originalmente tenía el número.

En caso de que el número original sea negativo también funciona esta lógica. No se olvide que siempre que vaya desarrollar un algoritmo primero debe **clarificar el objetivo** para que sepa hacia donde va y hasta donde debe llegar.

a. Usando Ciclo Mientras

```
Programa Ejemplo_1
Variables
        Entero :  Numero, Cuenta_Digitos                // Declara Variables
Inicio
        Escriba " Digite un número entero "             // Solicita un dato entero
        Lea Numero                                              // Lee un entero y lo almacena
                                                                // en la variable Numero

        Cuenta_Digitos = 0                              // Inicializa el contador en ceros

        Mientras Numero < > 0                           // Mientras Numero sea diferente de 0
                Numero = Numero / 10                            //Divida entre 10
                Cuenta_Digitos = Cuenta_Digitos + 1             // y cuente
        Fin_Mientras

        Escriba " Tiene ", Cuenta_Digitos, " dígitos "  // Escriba la cantidad de dígitos
Fin
```

Note usted que al lado derecho del algoritmo técnico está prácticamente escrito el algoritmo informal para mayor claridad al momento que usted le desarrolle una prueba de escritorio. Igualmente la doble barra inclinada (//) representa el inicio de un comentario. Los comentarios son textos explicativos de las órdenes o de bloque de órdenes de los algoritmos. No tienen incidencia en su ejecución. Todos los lenguajes tienen un equivalente para escribir comentarios. En nuestro caso vamos a utilizar la doble barra inclinada que es utilizada en Lenguaje C.

b. Usando Ciclo Para

Este algoritmo no es "fácilmente" implementable con un ciclo Para. Tenga en cuenta que la utilización de uno y otro ciclo es precisamente para que se haga mucho mas sencilla la implementación de su algoritmo. La razón por la cual no es fácilmente implementable con este ciclo se remite a su estructura ya que este ciclo necesita un valor exacto de inicio, un valor exacto de finalización y un incremento y en este caso no es fácil encontrar estos tres valores.

c. Usando Ciclo Haga Hasta

```
Programa Ejemplo_1
Variables
        Entero : Numero, Cuenta_Digitos                 // Declaración de Variables
Inicio
        Escriba " Digite un número "                    // Título de Aviso
```

```
        Lea Numero                                  // Lea un entero y guárdelo en Numero

        Cuenta_Digitos = 0                          // Inicializa Cuenta_Digitos en 0

        Haga                                        // Inicia el ciclo
                Numero = Numero / 10                // Divida entre 10
                Cuenta_Digitos = Cuenta_Digitos + 1 // y cuente

        Hasta que Numero = 0                        // Hasta que Numero sea igual a 0

        Escriba " Tiene ", Cuenta_Digitos, " Dígitos "   // Escriba el resultado solicitado
Fin
```

Cabe anotar que la estructuración de un algoritmo utilizando un Ciclo Haga Hasta implica razonar muy bien la condición del ciclo ya que como puede verse es diferente a la condición utilizada en el ciclo mientras.

Ciclo Mientras	*Mientras Numero < > 0*
Ciclo Haga Hasta	*Hasta que Numero = 0*

Normalmente si realizamos dos soluciones para un mismo algoritmo pero en una de ellas utilizamos un *Ciclo Mientras* y en otro utilizamos un *Ciclo Haga Hasta*, podríamos decir que las condiciones deben ser contrarias (a nivel lógico).

d. Usando Ciclo Haga Mientras

```
Programa Ejemplo_1
Variables
        Entero : Numero, Cuenta_Digitos             // Declaración de Variables
Inicio
        Escriba " Digite un número "                // Título de Aviso
        Lea Numero                                  // Lea un entero y guárdelo en Numero

        Cuenta_Digitos = 0                          // Inicializa Cuenta_Digitos en 0

        Haga                                        // Inicia el ciclo
                Numero = Numero / 10                // Divida entre 10
                Cuenta_Digitos = Cuenta_Digitos + 1 // y cuente

        Mientras Numero > 0                         // Mientras el Número sea mayor que 0

        Escriba " Tiene ", Cuenta_Digitos, " Dígitos "   // Escriba el resultado solicitado
Fin
```

Esta versión es muy aproximada a la anterior pero la condición esta vez es diferente. Tenga en cuenta que utilizar cualquier estructura de ciclos requiere razonar muy bien la condición que se ha de usar para que, a nivel lógico, el algoritmo entregue los resultados esperados.

2. Leer dos números enteros y mostrar todos los enteros comprendidos entre el menor y el mayor

En este ejemplo vamos a generar todos los números enteros comprendidos entre dos números leídos. Nótese que el algoritmo en ningún momento nos indica cuál es el mayor y cuál es el menor, lo cual significa que tenemos que averiguarlo antes de generar los números porque lo que el algoritmo sí dice es que deben escribirse **ascendentemente** (ya que reza *mostrar los enteros comprendidos entre el menor y el mayor*). Verifique si el objetivo es completamente claro para usted y de esta forma ahora sí puede revisar las soluciones aquí planteadas a este ejemplo.

Conceptualmente, primero vamos a detectar cuál de los dos números es el menor y cuál es el mayor. Luego haremos una variable Auxiliar igual al número menor y a medida que la vayamos incrementando en 1 vamos a ir escribiendo su contenido hasta cuando esta variable alcance el valor del número mayor.

a. Usando Ciclo Mientras

```
Programa Ejemplo_2
Variables
        Entero : Numero1, Numero2, Auxiliar          // Declaración de Variables
Inicio
        Escriba " Digite un Entero "                 // Solicita el primero número
        Lea Numero1                                  // Lo lee y lo almacena en Numero1

        Escriba " Digite otro Entero "               // Solicita el segundo número
        Lea Numero2                                  // Lo lee y lo almacena en Numero2

        Si Numero1 <  Numero2                        // Si el primer número es el menor

                Auxiliar = Numero1                   // Haga Auxiliar igual a Numero1

                Mientras Auxiliar < = Numero2        // Mientras Auxiliar sea <=  Numero2
                        Escriba Auxiliar             // Escriba el contenido de Auxiliar
                        Auxiliar = Auxiliar + 1      // e incremente Auxiliar
                Fin_Mientras
        Fin_Si

        Si Numero2 > Numero1                         // Si el segundo número es el menor
                Auxiliar = Numero2                   // Haga Auxiliar igual a Numero2
                Mientras Auxiliar < = Numero1        // Mientras Auxiliar sea <= Numero2
                        Escriba Auxiliar             // Escriba el contenido de Auxiliar
                        Auxiliar = Auxiliar + 1      // e incremente Auxiliar
                Fin_Mientras
```

```
        Fin_Si

        Si Numero1 = Numero2                    // Si los dos números son iguales
                Escriba "Los números son iguales "      // Avise que son iguales

Fin
```

b. Usando Ciclo Para

```
Programa Ejemplo_2
Variables
        Entero : Num1, Num2, Aux                        // Declaración de variables
Inicio
        Escriba " Digite un Entero "                    // Solicita el primero número
        Lea Num1                                        // Lo lee y lo almacena en Num1

        Escriba " Digite otro Entero "                  // Solicita el segundo número
        Lea Num2                                        // Lo lee y lo almacena en Num2

        Si Num1 < Num2                                  // Si Num1 es el menor
                Para Aux = Num1 hasta Num2 Paso 1       // Haga que Aux vaya de Num1 a
                                                        // Num2 de 1 en 1
                        Escriba Auxiliar                // y escriba cada valor de Aux
        Fin_Si

        Si Num2 < Num1                                  // Si Num2 es el menor
                Para Aux = Num2 hasta Num1 Paso 1       // Haga que Aux vaya de Num2 a
                                                        // Num1 de 1 en 1
                        Escriba Auxiliar                // y escriba cada valor de Aux
        Fin_Si

        Si Num1 = Num2                                  // Si los dos números son iguales
                Escriba " Los números son iguales "     // Avise que son iguales
        Fin_Si

Fin
```

c. Usando Ciclo Haga Hasta

```
Programa Ejemplo_2
Variables
        Entero : Num1, Num2, Aux
Inicio
```

```
        Escriba " Digite un Entero "            // Solicita el primero número
        Lea Num1                                // Lo lee y lo almacena en Num1

        Escriba " Digite otro Entero "          // Solicita el segundo número
        Lea Num2                                // Lo lee y lo almacena en Num2

        Si Num1 < Num2                          // Si Num1 es el menor
                Aux = Num1                      // Inicie Aux con el valor de Num1
                Haga
                        Escriba Aux             // Escriba el contenido de Aux
                        Aux = Aux + 1           // e incremente dicho contenido
                Hasta que Aux = Num2            // Hasta que Aux alcance a Num2
        Fin_Si

        Si Num2 < Num1                          // Si Num2 es el menor
                Aux = Num2                      // Inicie Aux con el valor de Num2
                Haga
                        Escriba Aux             // Escriba el contenido de Aux
                        Aux = Aux + 1           // e incremente dicho contenido
                Hasta que Aux = Num1            // Hasta que Aux alcance a Num1

        Si Num1 = Num2                          // Si los dos números son iguales
                Escriba "Los números son iguales"   // Avise que son iguales

Fin
```

d. Usando Ciclo Haga Mientras

```
Programa Ejemplo_2
Variables
        Entero : Num1, Num2, Aux
Inicio
        Escriba " Digite un Entero "            // Solicita el primero número
        Lea Num1                                // Lo lee y lo almacena en Num1

        Escriba " Digite otro Entero "          // Solicita el segundo número
        Lea Num2                                // Lo lee y lo almacena en Num2

        Si Num1 < Num2                          // Si Num1 es el menor
                Aux = Num1                      // Inicie Aux con el valor de Num1
                Haga
                        Escriba Aux             // Escriba el contenido de Aux
                        Aux = Aux + 1           // e incremente dicho contenido
                Mientras Aux <= Num2            // Mientras que Aux sea <= Num2
        Fin_Si

        Si Num2 < Num1                          // Si Num2 es el menor
                Aux = Num2                      // Inicie Aux con el valor de Num2
                Haga
                        Escriba Aux             // Escriba el contenido de Aux
                        Aux = Aux + 1           // e incremente dicho contenido
```

```
        Mientras Aux <= Num1                // Mientras que Aux sea <= Num1

    Si Num1 = Num2                          // Si los dos números son iguales
        Escriba "Los números son iguales"   // Avise que son iguales

Fin
```

3. Leer dos números enteros y determinar cual de los dos tiene mas dígitos

Para la solución de este problema podemos remitirnos al algoritmo que nos permitía saber cuántos dígitos tenía un número ya que en el fondo éste es casi lo mismo. Fundamentalmente el objetivo de este es contar los dígitos que tiene un número, luego contar los dígitos que tiene otro número y comparar ambos resultados para decir cuál de los dos tiene mas dígitos.

a. Usando Ciclo Mientras

```
Programa Ejemplo_3
Variables
    Entero :    Num1,       // Almacenará el primer número
                Num2,       // Almacenará el segundo número
                Aux1,       // Almacenará provisionalmente a Num1
                Aux2,       // Almacenará provisionalmente a Num2
                ContDig1,   // Almacenará la cantidad de dígitos de Num1
                ContDig2    // Almacenará la cantidad de dígitos de Num2
Inicio
    Escriba "Digite un entero"      // Solicita un dato entero
    Lea Num1                        // Lo lee y lo almacena en Num1

    Escriba "Digite otro entero"    // Solicita otro dato entero
    Lea Num2                        // Lo lee y lo almacena en Num2


    Aux1 = Num1                     // Almacene en Aux1 el contenido de Num1
    ContDig1 = 0                    // Inicialice ContDig1 en 0

    Mientras Aux1 < > 0             // Mientras Aux1 sea difte. de cero
        Aux1 = Aux1 / 10            // Divida Aux1 entre 10
        ContDig1 = ContDig1 + 1     // y cuente
    Fin_Mientras


    Aux2 = Num2                     // Almacene en Aux2 el contenido de Num2
    ContDig2 = 0                    // Inicialice ContDig2 en 0

    Mientras Aux2 < > 0             // Mientras Aux2 sea difte. de cero
        Aux2 = Aux2 / 10            // Divida Aux2 entre 10
```

```
            ContDig2 = ContDig2 + 1          // y cuente
        Fin_Mientras

        Si ContDig1 > ContDig2               // Si el primer número tiene mas dígitos
            Escriba Num1, " tiene mas dígitos que ", Num2          // Avise

        Si ContDig1 < ContDig2               // Si el segundo número tiene mas dígitos
            Escriba Num2, " tiene mas dígitos que ", Num1          // Avise

        Si ContDig1 = ContDig2               // Si los dos números tienen la misma cantidad
                                             // de dígitos
            Escriba Num1, " tiene la misma cantidad de dígitos que ", Num2          // Avise

Fin
```

Como puede ver nos hemos basado en el algoritmo que determinaba cuántos dígitos tenía un número para desarrollar este algoritmo.

b. Usando Ciclo Para

Este algoritmo no es implementable con un Ciclo Para por las mismas razones que no era implementable el algoritmo en el cual nos basamos para desarrollar esta solución. Debo anotar que en casos muy particulares como el Lenguaje C este algoritmo es implementable pero debido a que la filosofía de este lenguaje en cuanto al Ciclo Para es un poquito diferente y además porque las "herramientas" de programación que brinda este lenguaje son un poco mas flexible que en otros lenguajes.

c. Usando Ciclo Haga Hasta

```
Programa Ejemplo_3
Variables
        Entero :    Num1,           // Almacenará el primer número
                    Num2,           // Almacenará el segundo número
                    Aux1,           // Almacenará provisionalmente a Num1
                    Aux2,           // Almacenará provisionalmente a Num2
                    ContDig1,       // Almacenará la cantidad de dígitos de Num1
                    ContDig2        // Almacenará la cantidad de dígitos de Num2
Inicio
        Escriba "Digite un entero"          // Solicita un dato entero
        Lea Num1                            // Lo lee y lo almacena en Num1

        Escriba "Digite otro entero"        // Solicita otro dato entero
        Lea Num2                            // Lo lee y lo almacena en Num2


        Aux1 = Num1                         // Almacena en Aux1 el contenido de Num1
```

```
        ContDig1 = 0                    // Inicialice ContDig1 en 0

        Haga
                Aux1 = Aux1 / 10        // Divida Aux1 entre 10
                ContDig1 = ContDig1 +1  // y cuente
        Hasta que Aux1 = 0              // Hasta que Aux1 sea igual a 0


        Aux2 = Num2                     // Almacena en Aux2 el contenido de Num2
        ContDig2 = 0                    // Inicialice ContDig2 en 0

        Haga
                Aux2 = Aux2 / 10        // Divida Aux2 entre 10
                ContDig2 = ContDig2 +1  // y cuente
        Hasta que Aux2 = 0              // Hasta que Aux2 sea igual a 0


        Si ContDig1 > ContDig2          // Si el primer número tiene mas dígitos
                Escriba Num1, " tiene mas dígitos que ", Num2       // Avise

        Si ContDig1 < ContDig2          // Si el segundo número tiene mas dígitos
                Escriba Num2, " tiene mas dígitos que ", Num1       // Avise

        Si ContDig1 = ContDig2          // Si los dos números tienen la misma cantidad
                                        // de dígitos
                Escriba Num1, " tiene la misma cantidad de dígitos que ", Num2      // Avise

Fin
```

d. Usando Ciclo Haga Mientras

```
Programa Ejemplo_3
Variables
        Entero :        Num1,           // Almacenará el primer número
                        Num2,           // Almacenará el segundo número
                        Aux1,           // Almacenará provisionalmente a Num1
                        Aux2,           // Almacenará provisionalmente a Num2
                        ContDig1,       // Almacenará la cantidad de dígitos de Num1
                        ContDig2        // Almacenará la cantidad de dígitos de Num2
Inicio
        Escriba "Digite un entero"      // Solicita un dato entero
        Lea Num1                        // Lo lee y lo almacena en Num1

        Escriba "Digite otro entero"    // Solicita otro dato entero
        Lea Num2                        // Lo lee y lo almacena en Num2


        Aux1 = Num1                     // Almacena en Aux1 el contenido de Num1
        ContDig1 = 0                    // Inicialice ContDig1 en 0

        Haga
                Aux1 = Aux1 / 10        // Divida Aux1 entre 10
```

```
        ContDig1 = ContDig1 +1          // y cuente
    Mientras  Aux1 < > 0                // Mientras Aux1 sea diferente de cero

    Aux2 = Num2                         // Almacena en Aux2 el contenido de Num2
    ContDig2 = 0                        // Inicialice ContDig2 en cero

    Haga
        Aux2 = Aux2 / 10                // Divida Aux2 entre 10
        ContDig2 = ContDig2 +1          // y cuente
    Mientras Aux2 < > 0                 // mientras Aux2 sea diferente de cero

    Si ContDig1 > ContDig2              // Si el primer número tiene mas dígitos
        Escriba Num1, " tiene mas dígitos que ", Num2          // Avise

    Si ContDig1 < ContDig2              // Si el segundo número tiene mas dígitos
        Escriba Num2, " tiene mas dígitos que ", Num1          // Avise

    Si ContDig1 = ContDig2              // Si los dos números tienen la misma cantidad
                                        // de dígitos
        Escriba Num1, " tiene la misma cantidad de dígitos que ", Num2          // Avise

Fin
```

4. Leer números enteros hasta que digiten 0 y determinar a cuánto es igual el promedio de los números leídos que hayan sido positivos.

De nuevo y al igual que en todos los ejercicios vamos a clarificar el objetivo. Necesitamos leer números enteros y a medida que vamos leyendo estos números los vamos contando y los vamos acumulando en sendas variables. Cuando digiten el número 0 eso significará que en ese momento debemos obtener el resultado entero de dividir el resultado de haber acumulado todos los números entre el resultado de haberlos contado es decir lo que vamos a dividir es la sumatoria de números entre la cantidad de números y ese resultado es lo que debemos mostrar en pantalla porque es el dato que nos están solicitando.

a. Usando Ciclo Mientras

```
Programa  Ejemplo_4
Variables
    Entero :Num,                    // Almacenará cada uno de los
                                        // números leídos
                Acum,               // Almacenará la suma de los números
                                    // leídos diferentes de cero
                Cont                // Almacenará la cantidad de números
                                    // leídos diferentes de cero
    Real :      Promedio            // Almacenará el resultado de dividir la
                                    // sumatoria de números entre la
                                    // cantidad de números
```

```
Inicio
        Acum = 0                                    // Inicialice el acumulador en cero
        Cont = 0                                    // Inicialice el contador en cero

        Escriba " Digite enteros y finalice con  0 "    // Aviso para solicitar los números
        Lea Num                                     // Lea el primer número

        Mientras Num < > 0                          // Mientras los números que entren
                                                    // sean diferentes de cero
                Si Num > 0                          // Si el último número leído es positivo
                        Acum = Acum + Num           // Acumule el último número leído
                        Cont = Cont + 1         // y cuéntelo
                Fin_Si

                Lea Num                             // Lea un nuevo número
        Fin_Mientras

        Promedio = Acum / Cont                      // Calcule el promedio

        Escriba "El promedio es ", Promedio         // Muestre el promedio en pantalla
Fin
```

Hemos utilizado la primera orden *Lea Num* para recibir el primer número y con esto poder entrar al ciclo.

b. Usando Ciclo Para

Si se supiera con exactitud cuántos números se van a leer entonces si sería implementable este algoritmo con el ciclo Para pero como el algoritmo dice que *Hasta que digiten 0* entonces este algoritmo no es implementable con un ciclo Para.

c. Usando Ciclo Haga Hasta

```
Programa  Ejemplo_4
Variables
        Entero : Num,                       // Almacenará cada uno de los
                                            // números leídos
                 Acum,                      // Almacenará la suma de los números
                                            // leídos diferentes de cero
                 Cont                       // Almacenará la cantidad de números
                                            // leídos diferentes de cero
        Real :   Promedio                   // Almacenará el resultado de dividir la
                                            // sumatoria de números entre la
                                            // cantidad de números
Inicio
        Acum = 0                            // Inicialice el acumulador en cero
        Cont = 0                            // Inicialice el contador en cero
```

```
        Escriba " Digite enteros y finalice con  0 "      // Aviso para solicitar los números
        Lea Num                                           // Lea el primer número

        Haga
                Acum = Acum + Num                         // Acumule el último número leído
                Cont = Cont + 1                   // Cuéntelo
                Lea Num                                   // Lea el siguiente número
        Hasta que Num = 0                                 // Hasta que el último número leído sea
                                                          // igual a 0

        Promedio = Acum / Cont                            // Calcule el promedio

        Escriba "El promedio es ", Promedio               // Muestre el promedio en pantalla
Fin
```

d. Usando Ciclo Haga Mientras

```
Programa  Ejemplo_4
Variables
        Entero :Num,                              // Almacenará cada uno de los
                                                  // números leídos
                        Acum,                     // Almacenará la suma de los números
                                                  // leídos diferentes de cero
                        Cont                      // Almacenará la cantidad de números
                                                  // leídos diferentes de cero
        Real :          Promedio                  // Almacenará el resultado de dividir la
                                                  // sumatoria de números entre la
                                                  // cantidad de números
Inicio
        Acum = 0                                  // Inicialice el acumulador en cero
        Cont = 0                                  // Inicialice el contador en cero

        Escriba " Digite enteros y finalice con  0 "      // Aviso para solicitar los números
        Lea Num                                           // Lea el primer número

        Haga
                Acum = Acum + Num                         // Acumule el último número leído
                Cont = Cont + 1                   // Cuéntelo
                Lea Num                                   // Lea el siguiente número
        Mientras Num < > 0                                // Mientras el último número leído sea
                                                          // diferente de 0

        Promedio = Acum / Cont                            // Calcule el promedio

        Escriba "El promedio es ", Promedio               // Muestre el promedio en pantalla
Fin
```

5. Leer un número entero y calcular su factorial.

Primero que nada vamos a ver Qué es factorial de un número. Se define como Factorial de un número N cualquiera, el resultado de multiplicar sucesivamente todos los enteros comprendidos entre 1 y ese número N. No se aplica esta definición cuando dicho número N es negativo y en caso de que ese número N sea 0 se asume que el factorial de 0 es 1. De esta forma el factorial de 6 es el resultado de multiplicar 1 * 2 * 3 * 4 * 5 * 6 lo cual nos da como resultado 720, igualmente el factorial de 3 es igual a multiplicar 1 * 2 * 3 que es igual a 6, el factorial de 0 es 1 y el factorial de –8 no está definido o mejor no está incluido en la definición.

Ya con esta definición lo que tendremos que hacer es implementarla a nivel de un algoritmo. Para ello primero que nada vamos a leer dicho número N y validando de que dicho número sea positivo, utilizando otra variable a manera de contador, vamos a generar la secuencia de números enteros comprendida entre 1 y N solo que a medida que la vayamos generando vamos a ir multiplicando los números generados, resultado que será almacenado en una variable *Facto* que será la que al final se mostrará en pantalla para cumplir con el objetivo del enunciado.

a. Usando Ciclo Mientras

```
Programa Ejemplo_5
Variables
        Entero :N,                      // Almacenará el número leído que es el
                                        // número al cual se le quiere calcular el
                                        // factorial
                Cont,                   // Esta variable es la que nos va a permitir
                                        // generar la secuencia de todos los enteros
                                        // comprendidos entre 1 y el número leído (a
                                        // manera de contador)
                Facto                   // Almacenará el resultado del factorial
Inicio
        Escriba "Digite un número entero"   // Solicita un número entero
        Lea N                               // Lo lee y lo almacena en N

        Si N < 0                            // Si el número es negativo entonces avisa
                Escriba " El factorial no está definido para números negativos "

        Facto = 1                           // Inicializa el factorial en 1
        Cont = 1                            // Inicializa el contador en 1

        Mientras Cont < = N                 // Mientras el contador sea menor que el
                                            // número leído
                Facto = Facto * Cont        // Multiplique Facto por cada uno de los valores
                                            // que tome Cont
                Cont = Cont + 1         // Incremente el valor de Cont
        Fin_Mientras

        Escriba " El factorial de ", N, " es ", Facto       // Escriba el resultado
Fin
```

b. Usando Ciclo Para

```
Programa Ejemplo_5
Variables
        Entero :N,                      // Almacenará el número leído que es el
                                        // número al cual se le quiere calcular el
                                        // factorial
                Cont,                   // Esta variable es la que nos va a permitir
                                        // generar la secuencia de todos los enteros
                                        // comprendidos entre 1 y el número leído (a
                                        // manera de contador)
                Facto                   // Almacenará el resultado del factorial
Inicio
        Escriba "Digite un número entero"   // Solicita un número entero
        Lea N                               // Lo lee y lo almacena en N

        Si N < 0                            // Si el número es negativo entonces avisa
                Escriba " El factorial no está definido para números negativos "

        Facto = 1                           // Inicializa el factorial en 1

        Para Cont = 1 hasta N (Paso 1)      // Genera la secuencia de números enteros
                                            // desde 1 hasta el número leído de 1 en 1 y
                                            // cada valor lo va almacenando en la  Cont
                Facto = Facto * Cont        // Multiplique Facto por cada uno de los valores
                                            // que tome Cont
        Fin_Para

        Escriba " El factorial de ", N, " es ", Facto       // Escriba el resultado del factorial
Fin
```

c. Usando Ciclo Haga Hasta

```
Programa Ejemplo_5
Variables
        Entero :N,                      // Almacenará el número leído que es el
                                        // número al cual se le quiere calcular el
                                        // factorial
                Cont,                   // Esta variable es la que nos va a permitir
                                        // generar la secuencia de todos los enteros
                                        // comprendidos entre 1 y el número leído (a
                                        // manera de contador)
                Facto                   // Almacenará el resultado del factorial
Inicio
        Escriba "Digite un número entero"   // Solicita un número entero
        Lea N                               // Lo lee y lo almacena en N

        Si N < 0                            // Si el número es negativo entonces avisa
```

```
        Escriba " El factorial no está definido para números negativos "

    Facto = 1                           // Inicializa el factorial en 1
    Cont = 1                            // Inicializa el contador en 1

    Haga
        Facto = Facto * Cont            // Multiplique Facto por cada uno de los valores
                                        // que vaya tomando Cont
        Cont = Cont + 1         // Incremente el valor de Cont
    Hasta que Cont > N                  // Hasta que el valor almacenado en Cont sea
                                        // mayor que el valor almacenado en N

    Escriba " El factorial de ", N, " es ", Facto       // Escriba el resultado
Fin
```

d. Usando Ciclo Haga Mientras

```
Programa Ejemplo_5
Variables
    Entero :N,                  // Almacenará el número leído que es el
                                // número al cual se le quiere calcular el
                                // factorial
            Cont,               // Esta variable es la que nos va a permitir
                                // generar la secuencia de todos los enteros
                                // comprendidos entre 1 y el número leído (a
                                // manera de contador)
            Facto               // Almacenará el resultado del factorial
Inicio
    Escriba "Digite un número entero"   // Solicita un número entero
    Lea N                               // Lo lee y lo almacena en N

    Si N < 0                            // Si el número es negativo entonces avisa
        Escriba " El factorial no está definido para números negativos "

    Facto = 1                           // Inicializa el factorial en 1
    Cont = 1                            // Inicializa el contador en 1

    Haga
        Facto = Facto * Cont            // Multiplique Facto por cada uno de los valores
                                        // que vaya tomando Cont
        Cont = Cont + 1         // Incremente el valor de Cont
    Mientras Cont < = N                 // Mientras que el valor almacenado en Cont
                                        // sea menor o igual al valor almacenado en N

    Escriba " El factorial de ", N, " es ", Facto       // Escriba el resultado
Fin
```

6. Leer un número (asumir que es una base) y leer otro número (asumir que es un exponente) y elevar dicha base a dicho exponente.

Al igual que en todos los casos anteriores lo que necesitamos esta vez es leer dos números, asumiremos que el primero es una base y que el segundo es un exponente. Se trata de elevar dicha base a dicho exponente. Recordemos pues que si la base fuera 5 y el exponente fuera 4 entonces tendríamos que calcular a cuánto es igual 5^4 o sea 5 * 5 * 5 * 5 que quiere decir multiplicar el número 5 cuatro veces.

Por lo tanto necesitaremos que una variable actúe como contador para que nos permita controlar las veces que la base se va a multiplicar. La restricción fundamental que hemos de tener en cuenta es que el exponente sea positivo ya que b^n es diferente de b^{-n}.

a. Usando Ciclo Mientras

```
Programa Ejemplo_6
Variables
        Entero :Base,                   // Almacenará el número que se tomará como
                                        // base
                Exp,                    // Almacenará el número que se tomará como
                                        // exponente
                Aux,                    // Variable que va a almacenar
                                        // Progresivamente todos los valores
                                        // comprendidos entre 1 y el valor del
                                        // exponente
                Resul                   // Variable que almacenará el resultado final de
                                        // haber multiplicado la Base tantas veces como
                                        // lo diga el Exponente
Inicio

        Escriba " Digite una base "     // Solicita una Base
        Lea Base                        // y la lee

        Escriba " Digite un exponente " // Solicita un exponente
        Lea Exp                         // y lo lee

        Si Exp < 0                      // En caso de que el exponente sea negativo
                Escriba " Exponente Errado "    // Avisa
        Sino                            // Si el exponente es cero o es positivo
                Resul = 1               // Inicializa Resul en 1
                Aux = 1                 // Inicializa la auxiliar en 1

                Mientras Aux <= Exp     // Mientras el contenido de la variable auxiliar
                                        // sea menor o igual que el contenido de la
                                        // variable que almacena el exponente
                        Resul = Resul * Base    // Multiplica
                        Aux = Aux + 1           // e incremente el valor de la auxiliar
                Fin_Mientras
```

```
        Escriba " Resultado = ", Resul   // Muestra el resultado
Fin
```

b. Usando Ciclo Para

```
Programa Ejemplo_6
Variables
    Entero :Base,                  // Almacenará el número que se tomará como
                                   // base
            Exp,                   // Almacenará el número que se tomará como
                                   // exponente
            Aux,                   // Variable que va a almacenar
                                   // Progresivamente todos los valores
                                   // comprendidos entre 1 y el valor del
                                   // exponente
            Resul                  // Variable que almacenará el resultado final de
                                   // haber multiplicado la Base tantas veces como
                                   // lo diga el Exponente
Inicio

    Escriba " Digite una base "    // Solicita una Base
    Lea Base                       // y la lee

    Escriba " Digite un exponente "  // Solicita un exponente
    Lea Exp                        // y lo lee

    Si Exp < 0                     // En caso de que el exponente sea negativo
        Escriba " Exponente Errado "  // Avisa
    Sino                           // Si el exponente es cero o es positivo
        Resul = 1                  // Inicializa Resul en 1

        Para Aux = 1 hasta Exp ( Paso 1)   // Inicia un ciclo desde 1 hasta el valor
                                           // que tiene almacenado la variable Exp
                                           // guardando cada valor en la variable
                                           // Aux
            Resul = Resul * Base           // Por cada valor de Aux multiplica lo
                                           // que contenga la variable Resul por lo
                                           // que almacena la variable Base
        Fin_Para

        Escriba " Resultado = ", Resul     // Muestra en pantalla el resultado
                                           // solicitado
Fin
```

c. Usando Ciclo Haga Hasta

```
Programa Ejemplo_6
```

```
Variables
        Entero :Base,                   // Almacenará el número que se tomará como
                                        // base
                Exp,                    // Almacenará el número que se tomará como
                                        // exponente
                Aux,                    // Variable que va a almacenar
                                        // Progresivamente todos los valores
                                        // comprendidos entre 1 y el valor del
                                        // exponente
                Resul                   // Variable que almacenará el resultado final de
                                        // haber multiplicado la Base tantas veces como
                                        // lo diga el Exponente
Inicio

        Escriba " Digite una base "     // Solicita una Base
        Lea Base                        // y la lee

        Escriba " Digite un exponente " // Solicita un exponente
        Lea Exp                         // y lo lee

        Si Exp < 0                      // En caso de que el exponente sea negativo
                Escriba " Exponente Errado "  // Avisa
        Sino                            // Si el exponente es cero o es positivo
                Resul = 1               // Inicializa Resul en 1
                Aux = 1                 // Inicializa la auxiliar en 1

                Haga
                        Resul = Resul * Base  // Multiplica
                        Aux = Aux + 1         // e incrementa el valor de Aux
                Hasta que Aux > Exp     // Hasta que Aux sea mayor que Exp

                Escriba " Resultado ", Resul  // Escriba el resultado final

Fin
```

d. Usando Ciclo Haga Mientras

```
Programa Ejemplo_6
Variables
        Entero :Base,                   // Almacenará el número que se tomará como
                                        // base
                Exp,                    // Almacenará el número que se tomará como
                                        // exponente
                Aux,                    // Variable que va a almacenar
                                        // Progresivamente todos los valores
                                        // comprendidos entre 1 y el valor del
                                        // exponente
                Resul                   // Variable que almacenará el resultado final de
                                        // haber multiplicado la Base tantas veces como
                                        // lo diga el Exponente
Inicio
```

```
Escriba " Digite una base "          // Solicita una Base
Lea Base                             // y la lee

Escriba " Digite un exponente "      // Solicita un exponente
Lea Exp                              // y lo lee

Si Exp < 0                           // En caso de que el exponente sea negativo
        Escriba " Exponente Errado " // Avisa
Sino                                 // Si el exponente es cero o es positivo
        Resul = 1                    // Inicializa Resul en 1
        Aux = 1                      // Inicializa la auxiliar en 1

        Haga
                Resul = Resul * Base // Multiplica
                Aux = Aux + 1        // e incrementa el valor de Aux
        Mientras Aux <= Exp          // Mientras que Aux sea menor o igual que Exp

        Escriba " Resultado ", Resul // Escriba el resultado final
```

Fin

A esta altura del libro usted podrá ver la gran utilidad que tiene la implementación de ciclos en nuestros algoritmos dado que ellos facilitan no solo la comprensión de la solución sino que además nos permiten obtener unos algoritmo muy flexibles a los cambios pues con solo cambiar el valor de inicio del ciclo o el valor final del ciclo o el incremento del ciclo, con eso es suficiente para que ese mismo ciclo pueda cumplir de una manera eficiente otros objetivos.

Ciclos anidados

Una variante muy interesante en cuanto a la concepción de ciclos corresponde a los ciclos anidados. Se define como ciclos anidados la estructura en la cual un ciclo está dentro de otro (completamente). Esto significa que si se tratara de una estructura mientras anidada su esquema sería el siguiente

```
Mientras Condición1
        .
        .
        Mientras Condición2
                .
                .
                Cuerpo del ciclo mas interno
                .
                .
        Fin_Mientras Interno
        .
        .
Fin_Mientras Externo
```

Su forma de ejecución es muy sencilla: Mientras sea Verdadera la condición1 el ciclo externo va a ejecutar el cuerpo del ciclo dentro del cual se encuentra un ciclo interno que está regido por la condición2 de tal manera que se ejecutará completamente el ciclo interno mientras la condición2 sea Verdadera. Cuando ésta condición2 sea Falsa se ejecutarán el resto de instrucciones del ciclo externo y se volverá a la condición1 a evaluar si aún es verdadera para volver a entrar en el cuerpo de dicho ciclo. Veamos esta teoría aplicada con un ejemplo al cual le haremos su respectiva prueba de escritorio :

Ejemplo 1

Leer números hasta que digiten 0 y a cada valor leído calcularle su factorial.

Clarificación del objetivo

No se olvide que la solución presentada en este libro es solo una de las posibles soluciones a cada uno de los enunciados planteados como objetivos. Su usted ha desarrollado soluciones para estos algoritmos y encuentra que son diferentes no se preocupe. Realícele pruebas de escritorio y si cumplen con el objetivo entonces su algoritmo estará tan bien como el mío.

Para solucionar este algoritmo vamos a tener en cuenta la solución ya presentada para calcular el factorial de un número. En esencia nuestro algoritmo va a leer números hasta que digiten cero y, mientras esta condición sea verdadera, se calculará el factorial (o sea se aplicará el algoritmo ya resuelto de cálculo del factorial de un número) a cada uno de los números leídos.

No se olvide que el factorial de un entero es el resultado de multiplicar sucesivamente todos los enteros comprendidos entre 1 y el número dado. No está definido el factorial para números negativos y el factorial de 0 es 1.

Algoritmo

```
Programa Ciclos_Anidados_1
Variables
        Entero :Num,                    // Almacenará cada uno de los números
                                        // diferentes de cero que sean digitados
                Facto,                  // Almacenará el factorial de cada uno de los
                                        // números digitados
                Cont                    // Almacenará la secuencia de enteros
                                        // comprendidos entre 1 y el último número
                                        // leído, toda vez que este último número sea
                                        // positivo
Inicio
        Escriba " Digite números y finalice con 0 "     // Solicita números y avisa el finalizador
        Lea Num                                         // Lea el primer dato

        Mientras Num < > 0              // Mientras el dato leído sea diferente de 0

                Si  Num < 0             // Si el dato leído es negativo avise
```

```
                    Escriba " No está definido el factorial para números negativos "

            Sino                        // Sino
                    Facto = 1           // Inicialice la variable Facto en 1
                    Para Cont = 1 hasta Num (paso 1)    // Con un ciclo Para almacene
                                                        // sucesivamente en la variable
                                                        // Cont todos los enteros
                                                        // comprendidos entre 1 y el
                                                        // número leído
                            Facto = Facto * Cont        // Multiplique progresivamente
                                                        // cada uno de los valores
                                                        // enteros comprendidos entre 1
                                                        // y el número leído
                    Escriba " El factorial de ", Num, " es ", Facto   // Escriba el resultado
                                                                      // del factorial de cada
                                                                      // número leído
            Fin_Si
            Lea Num                         // Lea el siguiente número
    Fin_Mientras
Fin
```

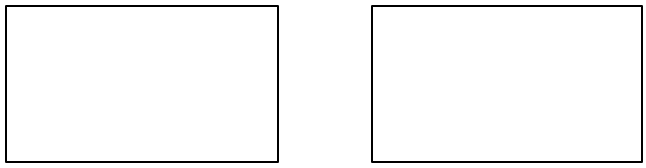
Note usted que la utilización de ciclos anidados no tiene restricciones en cuanto a los ciclos que se utilicen. En este ejemplo vemos como dentro de un ciclo Mientras puede ir sin ningún problema un ciclo Para. En general usted en un algoritmo puede combinar cualquier tipo de ciclos.

Prueba de Escritorio

De nuevo y tal como lo hicimos en ejemplos anteriores, vamos a desarrollar una prueba de escritorio (paso a paso) con el objetivo entender claramente el concepto de los ciclos anidados.

Inicialmente tendremos las mismas dos partes para la prueba de escritorio que hemos tenido en todas ellas: La pantalla y la Memoria.

P A N T A L L A M E M O R I A

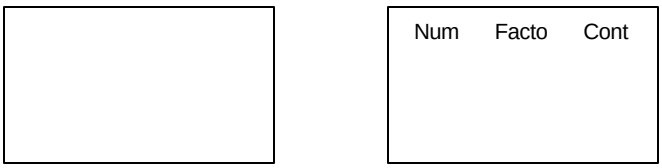


Comenzamos la prueba de escritorio con la declaración de las variables en memoria.

```
Programa Ciclos_Anidados_1
Variables
    Entero :Num,            // Almacenará cada uno de los números
                            // diferentes de cero que sean digitados
            Facto,          // Almacenará el factorial de cada uno de los
```

```
                                    // números digitados
Cont                                // Almacenará la secuencia de enteros
                                    // comprendidos entre 1 y el último número
                                    // leído, toda vez que este último número sea
                                    // positivo
```

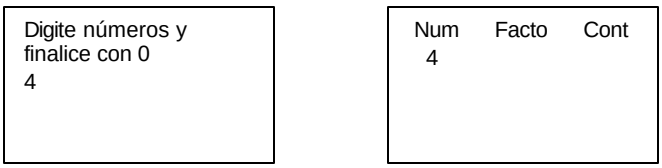
P A N T A L L A M E M O R I A



En la parte de ejecución nuestro algoritmo comienza avisándole al usuario para que ingrese unos datos y sepa como finalizar. A continuación lee el primer entero y lo almacena en la variable Num. Vamos a asumir que el primer entero digitado es un 4.

```
Inicio
    Escriba " Digite números y finalice con 0 "     // Solicita números y avisa el finalizador
    Lea Num                                         // Lea el primer dato
```

P A N T A L L A M E M O R I A



Como primera medida, con el número leído se verifica que sea diferente de cero y, como esta proposición es verdadera, se procede a entrar en el ciclo.

```
    Mientras Num < > 0                  // Mientras el dato leído sea diferente de 0
```

Se verifica igualmente si el número leído es menor que cero (o sea si es negativo). Como el número 4 no es negativo entonces se procede a realizar el conjunto de instrucciones que se encuentran por el lado del *Sino*.

```
        Si  Num < 0
            Escriba " No está definido el factorial para números negativos "

        Sino
            Facto = 1
```

De acuerdo a esto inicializamos la variable *Facto* con el valor de 1 para iniciar el ciclo *Para* que es el que realmente nos va a permitir calcular el factorial

P A N T A L L A

```
Digite números y
finalice con 0
4
```

M E M O R I A

Num	Facto	Cont
4	1	

El ciclo *Para* planteado tiene como índice a la variable Cont y le asigna inicialmente el valor de 1. El contenido de esta variable deberá llegar hasta el valor actual de Num (o sea 4) incrementándola cada vez de 1 en 1.

P A N T A L L A

```
Digite números y
finalice con 0
4
```

M E M O R I A

Num	Facto	Cont
4	1	1

Para Cont = 1 hasta Num (paso 1)
*Facto = Facto * Cont*

Y por cada valor que vaya tomando la variable *Cont* vamos realizando la operación asignación *Facto = Facto * Cont.* De esta manera las asignaciones sucesivas serán

Cuando *Cont* vale 1 entonces *Facto* vale 1

P A N T A L L A

```
Digite números y
finalice con 0
4
```

M E M O R I A

Num	Facto	Cont
4	~~1~~	1
	1	

Al incrementar el valor de *Cont* en 1 entonces *Cont* queda valiendo 2 y por lo tanto *Facto* es igual a 2.

P A N T A L L A

```
Digite números y
finalice con 0
4
```

M E M O R I A

Num	Facto	Cont
4	~~1~~	~~1~~
4	~~1~~	2
	2	

Al incrementar el valor de *Cont* en 1 entonces *Cont* queda ahora valiendo 3 y por lo tanto *Facto* es igual a 6 puesto que *Facto = Facto * Cont*

P A N T A L L A

Digite números y
finalice con 0
4

M E M O R I A

Num	Facto	Cont
4	~~1~~	~~1~~
4	~~1~~	~~2~~
4	~~2~~	3
4	6	

Al incrementar el valor de *Cont* (de nuevo) en 1 su actual contenido pasa a ser 4. Recuerda usted que el ciclo decía inicialmente *Para Cont = 1 hasta Num*, eso significa que como *Num* vale 4 y en este momento *Cont* también vale 4 entonces esta es la última iteración del ciclo *Para* planteado o sea que es la última vez que hacemos *Facto = Facto * Cont*. Con lo que acabo de decir pasaríamos a la siguiente instrucción dentro del ciclo *Mientras* externo. Es importante que note que hemos desarrollado el ciclo *Para* y que, aunque ya casi terminamos de ejecutarlo, no hemos terminado la ejecución del ciclo externo *Mientras*.

P A N T A L L A

Digite números y
finalice con 0
4

M E M O R I A

Num	Facto	Cont
4	~~1~~	~~1~~
4	~~1~~	~~2~~
4	~~2~~	~~3~~
4	~~6~~	4
4	24	

De esta forma se ejecutaría la siguiente orden que se encuentra después del ciclo. En las ordenes Escriba no se olvide que en donde aparezcan nombres de variables se coloca el contenido actual de ellas

```
        Escriba " El factorial de ", Num, " es ", Facto
Fin_Si
```

P A N T A L L A

Digite números y
finalice con 0
4
El factorial de 4 es 24

M E M O R I A

Num	Facto	Cont
4	~~1~~	~~1~~
4	~~1~~	~~2~~
4	~~2~~	~~3~~
4	~~6~~	4
4	24	

Resultado que efectivamente es cierto con lo cual terminamos la decisión pasando a la siguiente instrucción que se encuentra después de ella y que corresponde a leer otro entero. Luego de lo cual volveríamos a evaluar la condición del ciclo Mientras para saber si ya se ha terminado o no.

Lea Num
Fin_Mientras

Y para el caso de la prueba de escritorio que estamos desarrollando supongamos que el nuevo valor leído es –5 que, por la orden dada, quedaría almacenado en la variable *Num*.

P A N T A L L A

```
Digite números y
finalice con 0
4
El factorial de 4 es 24
-5
```

M E M O R I A

```
Num     Facto     Cont
 -5
```

He borrado los contenidos sucesivos de las variables Facto y Cont solo por hacer mas clara la prueba de escritorio a partir del momento en que entra un nuevo valor. Esto no tiene ninguna implicación en el desarrollo de ESTA prueba de escritorio ya que al leer un nuevo valor (según el algoritmo) se repite todo el proceso con él. Tenga usted mucho cuidado cuando vaya a borrar deliberadamente contenidos de variables. Le sugiero que mejor nunca lo haga. En este libro se hace por cuestión de espacio, no por otra razón pero admito que no es lo mas conveniente.

Con este valor volvemos a evaluar la condición que aparece en el inicio del ciclo o sea

Mientras Num < > 0

Como vemos que el valor almacenado en la variable Num es diferente de Cero (porque es igual a –5) entonces volvemos a ejecutar el cuerpo del ciclo. Lo primero que encontramos es una decisión acerca de Si el número es negativo (o sea menor que cero). Como puede verse el contenido de la variable Num en este momento es –5 lo cual indica que la decisión es Verdadera.

```
Si  Num < 0
        Escriba " No está definido el factorial para números negativos "
```

Por lo tanto se ejecuta la orden *Escriba* que se encuentra por el lado Verdadero de la Decisión y se ignora el correspondiente *Sino*.

P A N T A L L A

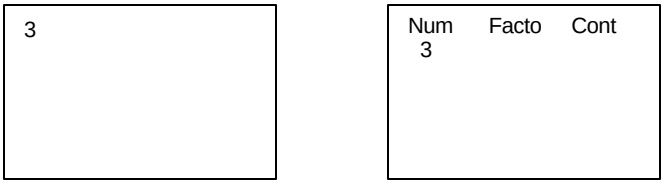
```
-5
No está definido el
factorial para números
negativos
```

M E M O R I A

```
Num     Facto     Cont
 -5
```

De esta manera la ejecución del algoritmo pasará a la siguiente instrucción que se encuentra después del final de la Decisión (o sea inmediatamente después del *Fin_Si*) y que es Lea Num o sea que lea otro número entero. Esta vez supongamos que el número leído es 3.

P A N T A L L A M E M O R I A



De esta forma, con el nuevo valor almacenado en la variable *Num*, volvemos a evaluar la condición del ciclo o sea

Mientras Num < > 0

Como el valor actual de Num es 3 entonces volvemos a entrar al ciclo (Externo). Note usted que aún no nos hemos salido del ciclo externo y que si el número leído es positivo tendremos que entrar también al ciclo interno.

```
Si  Num < 0
        Escriba " No está definido el factorial para números negativos "
```

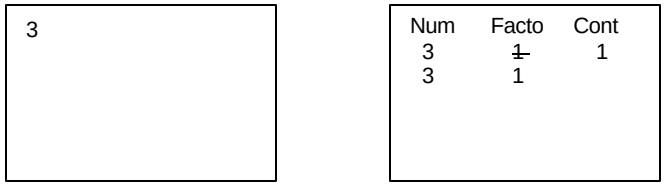
La primera decisión que se encuentra es si el número es negativo. Como vemos que Num almacena un número 3 y el 3 no es negativo entonces se ejecutarán las órdenes que se encuentran por el Sino de la decisión.

```
Sino
        Facto = 1
```

Ya sabemos que inicializamos la variable *Facto* con el valor de 1 y así mismo hacemos con la variable *Cont* que según el ciclo *Para* planteado irá desde 1 hasta el valor actual de *Num* (o sea hasta 3) de 1 en 1 para realizar progresivamente la operación *Facto = Facto * Cont*.

```
Para Cont = 1 hasta Num (paso 1)
        Facto = Facto * Cont
```

P A N T A L L A M E M O R I A



Nos quedamos en este ciclo *Para* realizando las iteraciones que sean necesarias hasta cuando *Cont* llegue al valor actual de *Num*. De esta manera incrementamos el valor de *Cont* en 1 y queda valiendo 2, razón por la cual la operación *Facto = Facto * Cont* es igual a 2.

P A N T A L L A

3

M E M O R I A

Num	Facto	Cont
3	~~1~~	~~1~~
3	~~1~~	2
3	2	

Al incrementar el valor de *Cont* en 1 (continuando con la ejecución del ciclo *Para*) obtenemos en la variable *Cont* el valor 3 que es el tope al cual debía llegar puesto que el ciclo estaba estructurado para que *Cont* fuera desde 1 hasta el valor actual de *Num* que es 3. Por lo tanto esta es la última iteración del ciclo *Para* (interno).

P A N T A L L A

3

M E M O R I A

Num	Facto	Cont
3	~~1~~	~~1~~
3	~~1~~	~~2~~
3	~~2~~	3
3	6	

Como ya se llegó al tope planteado por el ciclo para entonces pasamos a la siguiente instrucción o sea

Escriba " El factorial de ", Num, " es ", Facto

Con lo cual aparecería en pantalla

P A N T A L L A

3
El factorial de 3 es 6

M E M O R I A

Num	Facto	Cont
3	~~1~~	~~1~~
3	~~1~~	~~2~~
3	~~2~~	3
3	6	

Lo cual vemos que es cierto pues el factorial de 3 realmente es 6. Con esta orden llegamos al final de la Decisión o sea al *Fin_Si* pero no hemos terminado la prueba de escritorio pues después de

este *Fin_Si* ejecutamos la orden *Lea Num* para lo cual vamos a asumir que esta vez digitan el número 0.

P A N T A L L A

```
3
El factorial de 3 es 6
0
```

M E M O R I A

Num	Facto	Cont
3	~~1~~	~~1~~
3	~~1~~	~~2~~
3	~~2~~	3
3	6	
0		

Con lo cual volvemos a evaluar la condición que se encuentra en el ciclo Mientras y que dice

Mientras Num < > 0

Como vemos que esta vez el valor almacenado en *Num* es 0 (o sea que no es diferente de cero) entonces esto quiere decir que la condición es Falsa y por lo tanto nos salimos hasta la siguiente instrucción después del *Fin_Mientras.* Que es

Fin

Y que representa que hemos llegado al final de esta prueba de escritorio. Vemos que los resultados obtenidos en pantalla en las diferentes ejecuciones fueron

Digite números y finalice con 0
4
El factorial de 4 es 24
-5
No está definido el factorial para números negativos
3
El factorial de 3 es 6
0

Realizando manualmente las operaciones podemos ver que el factorial de 4 realmente es 24, que no está definido el factorial para números negativos (como por ejemplo el –5), que el factorial de 3 es 6 y que cuando se digitó cero se finalizó tal como se indicó al inicio del algoritmo. Con esto podemos decir que el algoritmo planteado aquí es correcto. No se olvide que cuando un ciclo está dentro de otro su orden de ejecución es muy importante para que realmente se obtengan los resultados esperados.

Ejemplo 2

Mostrar las tablas de multiplicar del 1 al 3.

Clarificación del Objetivo

Todos hemos conocido desde nuestra primaria las tablas de multiplicar. El objetivo de este algoritmo es mostrar en pantalla la tabla del 1, la tabla del 2 y la tabla del 3 tal como nos las enseñaron o sea

```
1 x 1 =  1
1 x 2 =  2
1 x 3 =  3
1 x 4 =  4
1 x 5 =  5
1 x 6 =  6
1 x 7 =  7
1 x 8 =  8
1 x 9 =  9
1 x 10 = 10

2 x 1 =  2
2 x 2 =  4
2 x 3 =  6
2 x 4 =  8
2 x 5 =  10
2 x 6 =  12
2 x 7 =  14
2 x 8 =  16
2 x 9 =  18
2 x 10 = 20

3 x 1 =  3
3 x 2 =  6
3 x 3 =  9
3 x 4 =  12
3 x 5 =  15
3 x 6 =  18
3 x 7 =  21
3 x 8 =  24
3 x 9 =  27
3 x 10 = 30
```

El objetivo es, básicamente, es mostrar en pantalla las tablas tal como se han mostrado aquí. Para nos vamos a valer de dos ciclos anidados. El ciclo externo nos va a permitir controlar que solo salga hasta la tabla del 3 es decir que el índice del ciclo externo va a ir de 1 a 3 y el ciclo interno es el que nos va a permitir que se generen los números del 1 al 10 para que al realizar la multiplicación entre el índice del ciclo externo y el índice del ciclo interno obtengamos los resultados esperados. Para facilidad de una posterior prueba de escritorio vamos a utilizar ciclos para dado que en este caso todos los topes son concretos.

De nuevo se trata solo de tener un ciclo externo con una variable que va a ir desde 1 hasta 3 y otro ciclo interno con una variable que va a ir desde 1 hasta 10. Dentro del ciclo interno estará ubicada una orden que permita mostrar en pantalla lo solicitado.

Algoritmo

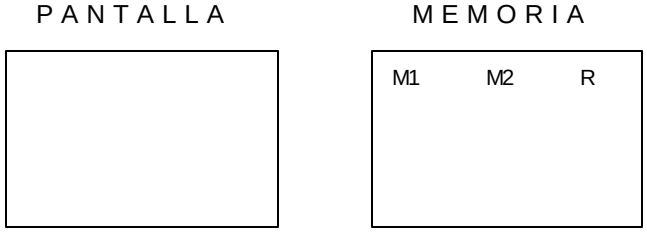
```
Programa Ciclos_Anidados_2
Variables
        Entero :M1,             // Indice el ciclo externo, almacenará el valor del
                                //multiplicando
                M2,             // Indice del ciclo interno, almacenará el valor del
                                // multiplicador
                R               // Almacena el resultado de multiplicar cada vez el
                                // Multiplicando por el Multiplicador
Inicio
        Escriba " Tablas de Multiplicar del 1 al 3 "    // Avisa qué es lo que va a escribir

        Para M1 = 1 hasta 3 (Paso 1)                    // Ciclo externo cuya variable va desde
                                                        // 1 hasta 3
                Para M2 = 1 hasta 10  (Paso 1)          // Ciclo interno cuya variable  va desde
                                                        // 1 hasta 10
                        R = M1 * M2                     // Resultado de cada Multiplicación
                        Escriba M1, " x ", M2, " = ", R // Muestra el resultado en la forma en
                                                        // que se conocen las tablas de
                                                        // multiplicar
                Fin_Para                                // Fin del ciclo interno
        Fin_Para                                        // Fin del ciclo externo
Fin                                                     // Fin del Algoritmo
```

Prueba de Escritorio

Teniendo nuestro algoritmo ya planteado vamos a desarrollarle una prueba de escritorio completa para acercarnos un poco mas al concepto de ciclos anidados. Primero que nada ubicamos en memoria tres variables dado que así es como comienza el algoritmo

```
Programa Ciclos_Anidados_2
Variables
        Entero :M1, M2, R
```



Aparece en pantalla un aviso que anuncia qué es lo que se va a mostrar a continuación

```
Inicio
```

Escriba " Tablas de Multiplicar del 1 al 3 "

P A N T A L L A

Tablas de Multiplicar del 1 al 3

M E M O R I A

M1	M2	R

El ciclo externo involucra una variable *M1* que va a tomar valores comenzando en 1 y terminando en 3 (inclusive) de 1 en 1. Así mismo el ciclo interno incluye una variable *M2* que va a tomar valores comenzando en 1 y terminando en 10 (inclusive) de 1 en 1. Dentro del ciclo mas interno ejecutará el resultado de multiplicar *M1* por *M2* y lo almacenará en la variable *R* mostrando todo en pantalla.

```
Para M1 = 1 hasta 3 (Paso 1)
    Para M2 = 1 hasta 10  (Paso 1)

        R = M1 * M2
        Escriba M1, " x ", M2, " = ", R
```

De esta manera mientras *M1* toma el valor de 1, *M2* tomará valores entre 1 y 10 (sin que cambie para nada el valor de *M1*). Entonces cuando *M2* valga 1 la ejecución será

P A N T A L L A

Tablas de Multiplicar del 1 al 3 1 x 1 = 1

M E M O R I A

M1	M2	R
1	1	1

Cuando *M2* tome el valor de 2 (manteniéndose el valor de 1 en *M1*)

P A N T A L L A

Tablas de Multiplicar del 1 al 3 1 x 1 = 1 1 x 2 = 2

M E M O R I A

M1	M2	R
1	~~1~~	~~1~~
1	2	2

Cuando *M2* tome el valor de 3 (manteniéndose el valor de 1 en *M1*)

P A N T A L L A

Tablas de Multiplicar del 1 al 3 1 x 1 = 1 1 x 2 = 2 1 x 3 = 3

M E M O R I A

M1	M2	R
1	~~1~~	~~1~~
1	~~2~~	~~2~~
1	3	3

Cuando *M2* tome el valor de 4 (manteniéndose el valor de 1 en *M1*)

P A N T A L L A

Tablas de Multiplicar
del 1 al 3

1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4

M E M O R I A

M1	M2	R
1	~~1~~	~~1~~
1	~~2~~	~~2~~
1	~~3~~	~~3~~
1	4	4

Cuando *M2* tome el valor de 5 (manteniéndose el valor de 1 en *M1*)

P A N T A L L A

Tablas de Multiplicar
del 1 al 3

1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5

M E M O R I A

M1	M2	R
1	~~1~~	~~1~~
1	~~2~~	~~2~~
1	~~3~~	~~3~~
1	~~4~~	~~4~~
1	5	5

Cuando *M2* tome el valor de 6 (manteniéndose el valor de 1 en *M1*)

P A N T A L L A

Tablas de Multiplicar
del 1 al 3

1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6

M E M O R I A

M1	M2	R
1	~~1~~	~~1~~
1	~~2~~	~~2~~
1	~~3~~	~~3~~
1	~~4~~	~~4~~
1	~~5~~	~~5~~
1	6	6

Cuando *M2* tome el valor de 7 (manteniéndose el valor de 1 en *M1*)

P A N T A L L A

Tablas de Multiplicar
del 1 al 3
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7

M E M O R I A

M1	M2	R
1	~~1~~	~~1~~
1	~~2~~	~~2~~
1	~~3~~	~~3~~
1	~~4~~	~~4~~
1	~~5~~	~~5~~
1	~~6~~	~~6~~
1	7	7

Cuando *M2* tome el valor de 8 (manteniéndose el valor de 1 en *M1*)

P A N T A L L A

```
Tablas de Multiplicar
del 1 al 3

1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
```

M E M O R I A

M1	M2	R
1	~~1~~	~~1~~
1	~~2~~	~~2~~
1	~~3~~	~~3~~
1	~~4~~	~~4~~
1	~~5~~	~~5~~
1	~~6~~	~~6~~
1	~~7~~	~~7~~
1	8	8

Cuando *M2* tome el valor de 9 (manteniéndose el valor de 1 en *M1*)

P A N T A L L A

```
Tablas de Multiplicar
del 1 al 3

1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
```

M E M O R I A

M1	M2	R
1	~~1~~	~~1~~
1	~~2~~	~~2~~
1	~~3~~	~~3~~
1	~~4~~	~~4~~
1	~~5~~	~~5~~
1	~~6~~	~~6~~
1	~~7~~	~~7~~
1	~~8~~	~~8~~
1	9	9

Cuando *M2* tome el valor de 10 (manteniéndose el valor de 1 en *M1*)

P A N T A L L A

```
Tablas de Multiplicar
del 1 al 3

1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
```

M E M O R I A

M1	M2	R
1	~~1~~	~~1~~
1	~~2~~	~~2~~
1	~~3~~	~~3~~
1	~~4~~	~~4~~
1	~~5~~	~~5~~
1	~~6~~	~~6~~
1	~~7~~	~~7~~
1	~~8~~	~~8~~
1	~~9~~	~~9~~
1	10	10

Como la variable *M2* ya llegó a su tope (que era 10) entonces nos salimos del ciclo interno (que incluye sus instrucciones) y volvemos al ciclo externo o sea a incrementar la variable *M1* en 1 de manera que quedaría con el valor de 2. Como esta variable aún no ha llegado a su tope entonces volvemos a entrar al cuerpo del ciclo externo que incluye un ciclo interno en donde una variable *M1* va a almacenar enteros empezando en 1 y terminando en 10 y ejecutándose (dentro de este ciclo interno) las órdenes de multiplicar y mostrar los resultados en pantalla. De esta forma *M1* toma el valor de 2 mientras *M2* va desde 1 hasta 10.

Cuando *M2* tome el valor de 1 (manteniéndose el valor de 2 en *M1*)

P A N T A L L A

```
.
.
.
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

2 x 1 = 1
```

M E M O R I A

M1	M2	R
.	.	.
.	.	.
1	~~7~~	~~7~~
1	~~8~~	~~8~~
1	~~9~~	~~9~~
~~1~~	~~10~~	~~10~~
2	1	1

Cuando *M2* tome el valor de 2 (manteniéndose el valor de 2 en *M1*)

P A N T A L L A

```
.
.
.
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10

2 x 1 = 2
2 x 2 = 4
```

M E M O R I A

M1	M2	R
.	.	.
.	.	.
1	~~7~~	~~7~~
1	~~8~~	~~8~~
1	~~9~~	~~9~~
~~1~~	~~10~~	~~10~~
2	~~1~~	~~2~~
2	2	4

Así iremos sucesivamente manteniendo el valor que almacena *M1* pero incrementando el valor de *M2* progresivamente a medida que vamos haciendo la multiplicación y vamos mostrando los resultados. De esta forma al finalizar la ejecución del ciclo interno tendremos tanto en memoria como en pantalla

P A N T A L L A

```
.
.
.
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

M E M O R I A

M1	M2	R
~~1~~	~~10~~	~~10~~
2	~~1~~	~~2~~
2	~~2~~	~~4~~
2	~~3~~	~~6~~
2	~~4~~	~~8~~
2	~~5~~	~~10~~
2	~~6~~	~~12~~
2	~~7~~	~~14~~
2	~~8~~	~~16~~
2	~~9~~	~~18~~
2	10	20

Igualmente puede usted realizar la prueba de escritorio en donde *M1* se incrementa en 1 quedando con el valor de 3 y *M2* vuelve a tomar valores desde 1 hasta 10 generando la tabla de multiplicar del 10. Luego de esto *M2* llega a su tope tal como sucedió en dos oportunidades mas solo que esta vez *M1* también llega a su tope por lo cual nos salimos del ciclo externo y ejecutamos la orden que

se encuentra inmediatamente después del *Fin_Ciclo* correspondiente. Esa orden no es mas que el Fin del algoritmo. Al final obtendremos el resultado esperado o sea tener en pantalla las tablas de multiplicar del 1 al 3. Si hubiéramos querido tener todas las tablas de multiplicar todo lo que tendríamos que hacer es cambiar la orden

Para M1 = 1 hasta 3 (Paso 1)

Por

Para M1 = 1 hasta 10 (Paso 1)

Y nuestro algoritmo quedaría generando todas las tablas de multiplicar.

Ejemplo 3

Leer un número entero y mostrar todos los enteros comprendidos entre 1 y cada uno de los dígitos del número leído

Clarificación del Objetivo

Antes de plantearle una solución a este enunciado no se olvide que los algoritmos presentados en este libro son solo una versión de solución a los problemas aquí mismo planteados. Esto quiere decir que si usted desarrolla un algoritmo que no es igual a los presentados en este libro pero al realizarle la correspondiente prueba de escritorio usted ve que también cumple con el objetivo entonces esto quiere decir que su algoritmo está bien, sin importar que sea diferente al presentado en este libro. Es muy importante que usted tenga esto en cuenta dado que muchas personas me preguntan lo mismo. Por ahora no se olvide que lo que importa es lograr el objetivo planteado utilizando un camino algorítmico.

En este ejercicio vamos a leer un número entero. Si es negativo lo multiplicamos por (-1) para facilitar las operaciones y las validaciones. Recordemos que la fórmula

*Dig = num – num / 10 * 10*

Va a almacenar en la variable Dig el último dígito de un número entero. Si por ejemplo num es igual a 1543 entonces

*Dig = num – num / 10 * 10*
*Dig = 1543 – 1543 / 10 * 10*
*Dig = 1543 - 154 * 10*
Dig = 1543 - 1540
Dig = 3

Que corresponde al último dígito del valor tomado como ejemplo. Entonces lo que vamos a hacer es que dentro de un ciclo vamos a ir obteniendo el último dígito del número y vamos a ir dividiendo

progresivamente el número entre 10. De esta manera habremos obtenido todos los dígitos por separado. Al tiempo que obtenemos cada dígito, a través de otro ciclo, vamos generando los números comprendidos entre 1 y el dígito obtenido. Todos los resultados se irán escribiendo en pantalla.

Algoritmo

```
Programa Ciclos_Anidados_3
Variables
        Entero :Num,                    // Almacenará el número leído
                 Dig,                   // Almacenará cada uno de los dígitos
                                        // que tenga el número leído
                 Aux                    // Almacenará cada uno de los enteros
                                        // comprendidos entre 1 y cada uno de
                                        // los dígitos del número leído
Inicio
        Escriba "Digite un número entero "  // Solicita un número entero, lo lee
        Lea Num                         // y lo almacena en la variable Num

        Si Num < 0                      // Si el número es negativo
                Num = Num * ( -1 )      // lo volvemos positivo para facilitar los
                                        // cálculos

        Mientras Num < > 0              // Mientras el contenido del número
                                        // leído sea diferente de cero
                Dig = Num – Num / 10 * 10   // Almacene el último dígito en la
                                        // variable Dig
                Para Aux = 1 hasta Dig (Paso 1)  // Genere la secuencia de enteros
                                        // comprendidos entre 1 y el dígito
                                        // obtenido y cada valor almacénelo en
                                        // la variable Aux
                        Escriba Aux     // Muestre cada uno de los valores que
                                        // tome la variable Aux
                Fin_Para

                Num = Num / 10          // Divida el número original entre 10
                                        // (para "quitarle" aritméticamente el
                                        // último dígito)
        Fin_Mientras                    // Fin del ciclo inicial
Fin                                     // Fin del Algoritmo
```

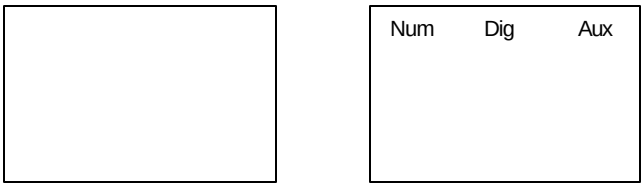
Prueba de Escritorio

Nuestra prueba de escritorio comienza con la declaración de tres variables en pantalla tal como se ha hecho en los otros ejercicios

```
Programa Ciclos_Anidados_3
Variables
        Entero :Num,
                 Dig,
```

Aux

P A N T A L L A M E M O R I A

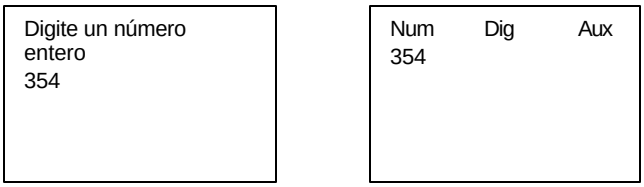


Inicio

Escriba "Digite un número entero "
Lea Num

Se solicita un dato entero, se lee y se almacena en la variable *Num*. Para efectos de nuestra prueba de escritorio vamos a asumir que el número digitado es el 354.

P A N T A L L A M E M O R I A



A continuación se pregunta si el contenido de la variable *Num* es negativo entonces se multiplica por –1 para facilitar los cálculos. Como en este caso el contenido de la variable *Num* es positivo entonces la decisión es Falsa por lo tanto nos saltamos la instrucción a ejecutar en caso de que la decisión fuera Verdadera. Continuando con el ciclo Mientras que está a continuación

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

El ciclo comienza estableciendo una condición para entrar en él. Mientras el contenido de la variable *Num* sea diferente de cero. Como en este caso el valor de la variable *Num* es 354 entonces la condición es Verdadera por lo tanto entramos al ciclo planteado

Mientras Num < > 0

Lo primero que vamos a realizar es la operación

*Dig = Num – Num / 10 * 10*

Según la cual se obtiene el último dígito del contenido de la variable *Num*. De esta forma como *Num* vale 354 entonces

*Dig = Num – Num / 10 * 10*

*Dig = 354 – 354 / 10 * 10*
*Dig = 354 – 35 * 10*
Dig = 354 – 350
Dig = 4

Y efectivamente obtenemos el último dígito del contenido de la variable *Num*. A continuación, siguiendo con nuestro algoritmo generaremos un ciclo en donde la variable *Aux* va a tomar valores desde 1 hasta el valor actual de *Dig* (o sea 4) incrementándola de 1 en 1, ciclo dentro del cual se irá escribiendo progresivamente el valor almacenado en la variable *Aux*

```
Para Aux = 1 hasta Dig (Paso 1)
        Escriba Aux
Fin_Para
```

De esta forma cuando *Aux* tenga el valor de 1 entonces se escribirá

P A N T A L L A

```
Digite un número
entero
354
1
```

M E M O R I A

Num	Dig	Aux
354	4	1

Cuando *Aux* tenga el valor de 2 entonces se escribirá

P A N T A L L A

```
Digite un número
entero
354
1 2
```

M E M O R I A

Num	Dig	Aux
354	4	~~1~~
		2

Cuando *Aux* tenga el valor de 3 entonces se escribirá

P A N T A L L A

```
Digite un número
entero
354
1 2 3
```

M E M O R I A

Num	Dig	Aux
354	4	~~1~~
		~~2~~
		3

Cuando *Aux* tenga el valor de 4 habrá llegado al tope planteado pues el ciclo inicialmente decía que *Aux* tomaría valores entre 1 y el valor actual de *Dig* que es 4 luego esta sería su última iteración por lo cual escribirá

P A N T A L L A

```
Digite un número
entero
354
1 2 3 4
```

M E M O R I A

Num	Dig	Aux
354	4	~~1~~
		~~2~~
		~~3~~
		4

Como ya se finalizó el ciclo *Para* planteado entonces continuamos con la instrucción que se encuentra después del *Fin_Para*

Num = Num / 10

Sabiendo que el contenido de la variable *Num* es 354 el resultado de esta expresión sería

Num = Num / 10
Num = 354 / 10
Num = 35

P A N T A L L A

```
Digite un número
entero
354
1 2 3 4
```

M E M O R I A

Num	Dig	Aux
~~354~~	4	~~1~~
35		~~2~~
		~~3~~
		4

Instrucción con la cual termina el ciclo pues encontramos a continuación el *Fin_Mientras* correspondiente.

Fin_Mientras

De tal manera que volvemos a la condición del ciclo *Mientras* para evaluarla y saber si se continúa ejecutando el cuerpo del ciclo ó se finaliza definitivamente la ejecución del mismo. Al volver al ciclo que decía

Mientras Num < > 0

Como el contenido actual de la variable *Num* es 35 entonces la condición es Verdadera por lo tanto volvemos a ejecutar el cuerpo del ciclo. Por lo tanto ejecutamos la orden

*Dig = Num – Num / 10 * 10*

Cuyo desarrollo sería

*Dig = Num – Num / 10 * 10*
*Dig = 35 – 35 / 10 * 10*
*Dig = 35 – 3 * 10*
Dig = 35 – 30
Dig = 5

Dejando almacenado en la variable *Dig* el valor 5

P A N T A L L A

```
Digite un número
entero
354
1 2 3 4
```

M E M O R I A

Num	Dig	Aux
~~354~~	~~4~~	~~1~~
35	5	~~2~~
		~~3~~
		4

A continuación vamos a generar un ciclo, utilizando la variable *Aux* como índice, para que tome valores entre 1 y el valor actual de *Dig* (que es 5) con incrementos de 1. De esta manera cuando *Aux* valga 1 entonces en pantalla se escribirá su contenido

P A N T A L L A

```
Digite un número
entero
354
1 2 3 4
1
```

M E M O R I A

Num	Dig	Aux
~~354~~	~~4~~	1
35	5	

Cuando *Aux* valga 2 entonces en pantalla se escribirá

P A N T A L L A

```
Digite un número
entero
354
1 2 3 4
1 2
```

M E M O R I A

Num	Dig	Aux
~~354~~	~~4~~	~~1~~
35	5	2

Cuando *Aux* valga 3 entonces en pantalla se escribirá

P A N T A L L A

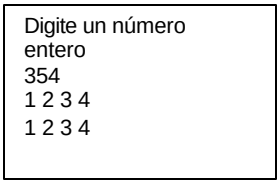
```
Digite un número
entero
354
1 2 3 4
1 2 3
```

M E M O R I A

Num	Dig	Aux
~~354~~	~~4~~	~~1~~
35	5	~~2~~
		3

Cuando *Aux* valga 4 entonces en pantalla se escribirá

P A N T A L L A

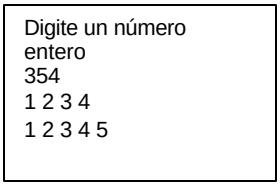


M E M O R I A

Num	Dig	Aux
~~354~~	~~4~~	~~1~~
35	5	~~2~~
		~~3~~
		4

Cuando *Aux* valga 5 entonces habrá llegado a su tope pues los valores tomados por la variable *Aux* llegarían hasta el valor actual de la variable *Dig*. De manera que esta sería su última iteración y en pantalla se escribirá

P A N T A L L A



M E M O R I A

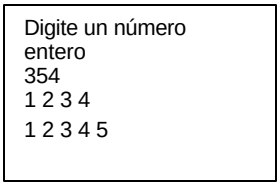
Num	Dig	Aux
~~354~~	~~4~~	~~1~~
35	5	~~2~~
		~~3~~
		~~4~~
		5

Al haber terminado el ciclo *Para* continuamos de nuevo con la instrucción que se encuentra después de su Fin_Para respectivo que es una asignación cuyo desarrollo sería

Num = Num / 10
Num = 35 / 10
Num = 3

Quedando en memoria almacenado en la variable *Num* el valor 3

P A N T A L L A



M E M O R I A

Num	Dig	Aux
~~354~~	~~4~~	~~1~~
~~35~~	5	~~2~~
3		~~3~~
		~~4~~
		5

Con lo cual, y dado que encontramos el fin del ciclo *Mientras*, debemos volver a éste para evaluar si su condición sigue siendo Verdadera

Mientras Num < > 0

Como en este momento el contenido de la variable *Num* es Verdadero dado que *Num* almacena un número 3 entonces volvemos a ejecutar una vez mas el cuerpo del ciclo que inicia con una instrucción de asignación cuyo desarrollo es

*Dig = Num – Num / 10 * 10*
*Dig = 3 – 3 / 10 * 10*
*Dig = 3 – 0 * 10*

Dig = 3 – 0
Dig = 3

Quedando almacenado en la variable *Dig* (en memoria obviamente) el valor 3

P A N T A L L A

Digite un número
entero
354
1 2 3 4
1 2 3 4 5

M E M O R I A

Num	Dig	Aux
~~354~~	~~4~~	
~~35~~	~~5~~	
3	3	

A continuación generamos, utilizando un ciclo *Para*, un secuencia de enteros comprendidos entre 1 y el valor actual de *Dig* (que es 3) con incrementos de 1 escribiendo cada contenido en la pantalla. Así cuando *Aux* valga 1 entonces se escribirá en pantalla

```
Para Aux = 1 hasta Dig (Paso 1)
        Escriba Aux
Fin_Para
```

P A N T A L L A

Digite un número
entero
354
1 2 3 4
1 2 3 4 5
1

M E M O R I A

Num	Dig	Aux
~~354~~	~~4~~	1
~~35~~	~~5~~	
3	3	

Cuando *Aux* valga 2 entonces se escribirá en pantalla

P A N T A L L A

Digite un número
entero
354
1 2 3 4
1 2 3 4 5
1 2

M E M O R I A

Num	Dig	Aux
~~354~~	~~4~~	~~1~~
~~35~~	~~5~~	2
3	3	

Cuando *Aux* valga 3 entonces se escribirá en pantalla el valor 3 y habrá finalizado el ciclo pues éste debería ir desde 1 hasta el valor actual de *Dig* que es 3.

P A N T A L L A

Digite un número
entero
354
1 2 3 4
1 2 3 4 5
1 2 3

M E M O R I A

Num	Dig	Aux
~~354~~	~~4~~	~~1~~
~~35~~	~~5~~	~~2~~
3	3	3

Continuamos con la siguiente orden del algoritmo y es una asignación cuyo desarrollo sería el siguiente

Num = Num / 10
Num = 3 / 10
Num = 0

Con lo cual volvemos a la condición inicial del ciclo a evaluar si sigue siendo Verdadera o no. La condición inicial del ciclo dice

Mientras Num < > 0

Como el valor actual de la variable *Num* es 0 entonces la condición es Falsa por lo cual pasamos a ejecutar la orden que sigue a continuación del *Fin_Mientras* y que corresponde al fin de todo el algoritmo.

Fin

Por último solo nos queda verificar lo que quedó en pantalla

P A N T A L L A

```
Digite un número
entero
354
1 2 3 4
1 2 3 4 5
1 2 3
```

Y podemos ver que en pantalla han quedado todos los enteros comprendidos entre 1 y cada uno de los dígitos de un número leído que era el objetivo inicial con lo cual podemos decir que nuestro algoritmo está bien.

Ejercicios

Algunas posibles soluciones a estos enunciados las puede encontrar en el libro *Algoritmos* del mismo autor.

1. Leer un número entero y mostrar todos los enteros comprendidos entre 1 y el número leído.

2. Leer un número entero y mostrar todos los pares comprendidos entre 1 y el número leído.

3. Leer un número entero y mostrar todos los divisores exactos del número comprendidos entre 1 y el número leído.

4. Leer dos números y mostrar todos los enteros comprendidos entre ellos.

5. Leer dos números y mostrar todos los números terminados en 4 comprendidos entre ellos.

6. Leer un número entero de tres dígitos y mostrar todos los enteros comprendidos entre 1 y cada uno de los dígitos.

7. Mostrar en pantalla todos los enteros comprendidos entre 1 y 100.

8. Mostrar en pantalla todos los pares comprendidos entre 20 y 200.

9. Mostrar en pantalla todos los números terminados en 6 comprendidos entre 25 y 205.

10. Leer un número entero y determinar a cuánto es igual la suma de todos los enteros comprendidos entre 1 y el número leído.

11. Leer un número entero de dos dígitos y mostrar en pantalla todos los enteros comprendidos entre un dígito y otro.

12. Leer un número entero de 3 dígitos y determinar si tiene el dígito 1.

13. Leer un entero y mostrar todos los múltiplos de 5 comprendidos entre 1 y el número leído.

14. Mostrar en pantalla los primeros 20 múltiplos de 3.

15. Escribir en pantalla el resultado de sumar los primeros 20 múltiplos de 3.

16. Mostrar en pantalla el promedio entero de los *n* primeros múltiplos de 3 para un número *n* leído.

17. Promediar los *x* primeros múltiplos de 2 y determinar si ese promedio es mayor que los *y* primeros múltiplos de 5 para valores de *x* y *y* leídos.

18. Leer dos números entero y mostrar todos los múltiplos de 5 comprendidos entre el menor y el mayor.

19. Leer un número entero y determinar si es primo.

20. Leer un número entero y determinar cuántos dígitos tiene.

21. Leer un número entero y determinar a cuánto es igual al suma de sus dígitos.

22. Leer un número entero y determinar cuántas veces tiene el dígito 1.

23. Leer un número entero y determinar si la suma de sus dígitos es también un número primo.

24. Leer un número entero y determinar a cuánto es igual al suma de sus dígitos pares.

25. Leer un número entero y determinar a cuánto es igual el promedio entero de sus dígitos.

26. Leer un número entero y determinar cuál es el mayor de sus dígitos.

27. Leer 2 números enteros y determinar cuál de los dos tiene mayor cantidad de dígitos.

28. Leer 2 números enteros y determinar cual de los dos tiene mayor cantidad de dígitos primos.

29. Leer un número entero y determinar a cuánto es igual el primero de sus dígitos.

30. Leer un número entero y mostrar todos sus componentes numéricos o sea aquellos para quienes el sea un múltiplo.

31. Leer números hasta que digiten 0 y determinar a cuánto es igual el promedio de los números terminados en 5.

32. Leer números hasta que digiten 0 y determinar a cuanto es igual el promedio entero de los número primos leídos.

33. Si 32768 es el tope superior para los números entero cortos, determinar cuál es el número primo mas cercano por debajo de él.

34. Generar los números del 1 al 10 utilizando un ciclo que vaya de 10 a 1.

35. Leer dos números enteros y determinar a cuánto es igual el producto mutuo del primer dígito de cada uno.

36. Mostrar en pantalla la tabla de multiplicar del número 5.

37. Generar todas las tablas de multiplicar del 1 al 10.

38. Leer un número entero y mostrar en pantalla su tabla de multiplicar.

39. Se define la serie de Fibonacci como la serie que comienza con los dígitos 1 y 0 y va sumando progresivamente los dos últimos elementos de la serie, así:

0 1 1 2 3 5 8 13 21 34.....

Utilizando el concepto de ciclo generar la serie de Fibonacci hasta llegar o sobrepasas el número 10000.

40. Leer un número de dos dígitos y determinar si pertenece a la serie de Fibonacci.

41. Determinar a cuánto es igual la suma de los elementos de la serie de Fibonacci entre 0 y 100.

42. Determinar a cuánto es igual el promedio entero de los elementos de la serie de Fibonacci entre 0 y 1000.

43. Determinar cuántos elementos de la serie de Fibonacci se encuentran entre 1000 y 2000.

44. Leer un número y calcularle su factorial.

45. Leer un número y calcularle el factorial a todos los enteros comprendidos entre 1 y el número leído.

46. Leer un número entero y calcular el promedio entero de los factoriales de los enteros comprendidos entre 1 y el número leído.

47. Leer un número entero y calcular a cuánto es igual la sumatoria de todos los factoriales de los números comprendidos entre 1 y el número leído.

48. Utilizando ciclos anidados generar las siguientes parejas de enteros

0 1
1 1
2 2
3 2
4 3
5 3
6 4

```
7       4
8       5
9       5
```

49. Utilizando ciclos anidados generar las siguientes ternas de números

```
1       1       1
2       1       2
3       1       3
4       2       1
5       2       2
6       2       3
7       3       1
8       3       2
9       3       3
```

50. Utilizando ciclos anidados generar las siguientes parejas de números

```
0       1
1       1
2       1
3       1
4       2
5       2
6       2
7       2
```