Capítulo 9

Arreglos

Concepto General

Un arreglo es un conjunto de variables en donde cada una de ellas puede ser referenciada utilizando su posición relativa es decir su ubicación en relación con el primer elemento de dicho conjunto. Estoy seguro que mas de una vez usted habrá notado que el cajero de un Banco para referirse a una de las personas que hacen cola para hacer una consignación puede decir "Venga la quinta persona" y habrá notado como, una persona que no estaba de primera, pasa a ser atendida por él. Varios razonamientos se involucran inconscientemente en esta situación:

1. En primera instancia sabemos que la "quinta persona" se ubica contando a partir de la primera. Esto es algo muy obvio pero para el tema que nos ocupa en el momento es importantísimo.

2. Normalmente todas las personas, sin equivocación, voltean a mirar a una sola persona que es, con toda seguridad, la que ocupa la quinta posición en la fila.

3. El "quinto puesto" en la cola tiene que estar ocupado por alguien. No puede estar vacío pues esa tuvo que haber sido una de las razones para que el cajero se refiriera a dicha persona con tanta seguridad.

4. La cola de personas no es infinita. Tiene una cantidad determinada de clientes.

5. Delante de la Quinta persona deben haber cuatro personas y después de ella pueden haber muchas o ninguna persona.

6. Si existen varias colas solo atenderá el llamado del cajero la “quinta persona” de la cola a la que él se refiera.

7. Si en algún momento otro cajero dijera “Vea, todas las personas de esa cola pasen pásense a esta caja”. Entonces todo el conjunto de personas atendería el llamado de ese otro cajero lo cual significaría que en cualquier momento ese conjunto de personas puede se manejado como una sola unidad.

Estos razonamientos, vuelvo a repetir, son muy obvios. En nuestro caso nos van a permitir caracterizar algunos elementos técnicos en referencia a los arreglos partiendo de unos conceptos muy simplificados y “domésticos”.

Cuando yo estaba en la escuela desatendía la costumbre de llevar todos los días los zapatos embolados y por lo tanto casi todas las mañanas escuchaba cómo, desde la dirección y mientras todos estábamos perfectamente formados, la directora decía “Allá, el tercer niño de la quinta fila acérquese a la Oficina de la Dirección”. Siempre en ese momento todos comenzaban a contar las filas y luego cuando habían ubicado la “quinta fila” comenzaban a buscar cual era el “tercer niño”. Obviamente era yo. Si la Directora solo hubiera dicho alguna vez “Venga el tercer niño para acá” se habría generado una confusión que me habría permitido pasar inadvertido pues como había varias filas entonces muchos “terceros niños” se hubieran sentido aludidos. Sin embargo en esta anécdota, que cuento con algo de vergüenza, también se involucran varios razonamientos:

1. Es evidente que todos los niños estaban formados en varias filas o, para mejor decirlo, existían varias filas en donde cada fila estaba formada por varios niños.

2. Cualquier niño era fácilmente ubicable diciendo sencillamente en qué fila y en qué posición dentro de esa fila se encontraba. Y dicha ubicación no daba espacio a la duda ya que cada uno tenía una única posición dentro de la formación.

3. Las filas se contaban partiendo de la primera en adelante y la posición de los niños dentro de cada fila se contaba partiendo del primer niño en adelante.

4. La escuela tenía una cantidad finita de niños que formaban todas las mañanas o sea que tanto las cantidad de filas como la cantidad de niños en cada fila era finita.

5. Antes de la “quinta fila” existían cuatro filas mas y después de la “quinta fila” podían existir mas filas o ninguna fila. Antes del tercer niño existían dos niños mas y después del tercer niño (en este caso) existían muchos niños mas aunque pudiera no haber existido ninguno.

6. Si en algún momento la Directora dijera “Todos los niños fórmense en el patio de allá” entonces podría manejar todo el grupo de niños como una sola unidad.

7. Cuando se referían a una fila determinada, ésta debía existir (con toda seguridad). Cuando se referían a un niño determinado en una fila determinada, y valga mucho esa redundancia, éste también debería existir.

8. En algún momento la Directora de la escuela pudo haber dicho “La tercera fila fórmese en ese lado de allá” y solo la tercera fila se habría movido quedando el resto de la formación en su estado original.

Qué pasa si frente a un colegio cuyo edificio tiene varios pisos y cada piso tiene varios salones una persona se para y dice: “Necesito al quinto estudiante “. Seguramente todos en el colegio se van a preguntar a partir de donde comienzan a contar para saber quién es el quinto estudiante y lo más posible es que nadie va a salir. Pero si esa persona se para frente al colegio y dice “ Necesito al quinto estudiante que está sentado en la tercera fila del cuarto salón del sexto piso de este colegio “ entonces casi con seguridad alguien va a salir porque fue ubicado apropiadamente.

Hemos tenido tres ejemplos: una cola en un banco, una formación en filas de los niños de una escuela y la ubicación de un estudiante en un colegio. En cada caso podemos ver cómo para ubicar efectivamente a una persona necesitamos utilizar referencias diferentes. Esto quiere decir

a. Para ubicar a una persona dentro de una cola todo lo que tenemos que hacer es utilizar UNA y solo UNA referencia. Es claro pues referirnos a la TERCERA persona ó a la QUINTA persona dentro de una cola pues no habría duda en su ubicación.

b. Para ubicar a un niño dentro de la población estudiantil de una escuela, cuando ésta está formada en el patio, necesitamos utilizar DOS y solo DOS referencias. De manera que al referirnos al TERCER niño de la QUINTA fila estamos determinando exactamente la posición incuestionable de un determinado niño.

c. Para ubicar a un estudiante en un colegio cuyo edificio tiene varios pisos y cada piso tiene varios salones necesitamos entonces utilizar varias referencias. Cuando se dice “Necesito al QUINTO estudiante que está sentado en la TERCERA fila del CUARTO salón del SEXTO piso” nos estamos refiriendo exactamente a una persona de manera incuestionable.

Es curioso pensar que ésta persona pudo haber sido la misma solo que para referirnos a ella tuvimos que ubicarla exactamente de manera diferente en cada una de las situaciones. En los tres casos las características son similares aún a pesar de que no es igual la distribución de los elementos (o personas para este caso).

Indices

Definición

Se conocen como índices todas aquellas variables que nos sirven para ubicar perfectamente un elemento dentro de un arreglo. De esta forma en el ejemplo de la cola de personas en vez de decir

"la quinta persona" podríamos haber dicho la "persona No. 5" o, para decirlo de una manera mas técnica, la persona sub 5. En el ejemplo del "tercer niño de la quinta fila" podríamos haber dicho también el niño No. 3 de la fila No. 5, igualmente con el ánimo de decirlo de una manera mas técnica, el "niño sub 3 de la fila sub 2". Cada dato numérico utilizado para ubicar estos elementos dentro de un arreglo se conocen como Indices.

Cuando el índice es reemplazado por una variable cuyo contenido es el dato entero que necesitamos entonces ésta variable se conoce como ***Subíndice***. De esta forma si, en el ejemplo de la cola tenemos una variable que se llama *Num* y *Num* contiene el número 5 entonces podríamos habernos referido a la "quinta persona" como la "Persona sub *Num*" (eso sí teniendo la seguridad de que *Num* contiene solo el número 5). Luego de esta manera si la cola tiene 10 personas entonces un algoritmo de atención para las 10 personas sería

```
Para Num = 1 hasta 10
        Atienda a Persona sub Num
```

O escrito de una forma más resumida, y de paso mas técnica, podríamos decir que

```
Para Num = 1 hasta 10
        Atienda a Persona ( Num )
```

Y de esta manera sabiendo que la variable *Num* puede tomar valores desde 1 hasta 10, pero que cada vez que tome un valor atiende a una determinada persona, se podrá concluir que

Cuando *Num* valga *1* atenderá a la *Persona (1)* que corresponde a la *primera persona*
Cuando *Num* valga *2* atenderá a la *Persona (2)* que corresponde a la *segunda persona*
Cuando *Num* valga *3* atenderá a la *Persona (3)* que corresponde a la *tercera persona*
Cuando *Num* valga *4* atenderá a la *Persona (4)* que corresponde a la *cuarta persona*
Cuando *Num* valga *5* atenderá a la *Persona (5)* que corresponde a la *quinta persona*
Cuando *Num* valga *6* atenderá a la *Persona (6)* que corresponde a la *sexta persona*
Cuando *Num* valga *7* atenderá a la *Persona (7)* que corresponde a la *séptima persona*
Cuando *Num* valga *8* atenderá a la *Persona (8)* que corresponde a la *octava persona*
Cuando *Num* valga *9* atenderá a la *Persona (9)* que corresponde a la *novena persona*
Cuando *Num* valga *10* atenderá a la *Persona (* 10) que corresponde a la *décima persona*

Como puede ver el manejo de la cola, sin importar cuántas personas tenga, se va a reducir al manejo de un ciclo con una variable importantísima que actuará como subíndice.

Igualmente supongamos que en la escuela existían solo 5 cursos y que cada curso tenía 10 estudiantes. Para referenciar a cada fila vamos a utilizar la variable *Fila* y para referenciar a cada niño dentro de la fila vamos a utilizar la variable *Pos* como para hacer referencia a la posición en la cual se encuentre. Igualmente vamos a asumir que Niño(*Fila*)(*Pos*) representará al niño que se encuentre en la fila *Fila* y dentro de ella en la posición *Pos*.

De manera que si queremos revisar la presentación de cada niño podríamos utilizar el siguiente fragmento de algoritmo

```
Para Fila = 1 hasta 5
        Para Pos =  1 hasta 10
                Revise al Niño (Fila) (Pos)
```

Con lo cual si le hacemos una pequeña "prueba de escritorio" a este algoritmo obtendremos que inicialmente *Fila* tendrá el valor de 1. Mientras mantiene este valor *Pos* va a tomar valores entre 1 y 10. De esta forma

Cuando Pos valga 1 se revisará al Niño (1) (1) o sea el primer niño de la primera fila
Cuando Pos valga 2 se revisará al Niño (1) (2) o sea el segundo niño de la primera fila
Cuando Pos valga 3 se revisará al Niño (1) (3) o sea el tercer niño de la primera fila
Cuando Pos valga 4 se revisará al Niño (1) (4) o sea el cuarto niño de la primera fila
Cuando Pos valga 5 se revisará al Niño (1) (5) o sea el quinto niño de la primera fila
Cuando Pos valga 6 se revisará al Niño (1) (6) o sea el sexto niño de la primera fila
Cuando Pos valga 7 se revisará al Niño (1) (7) o sea el séptimo niño de la primera fila
Cuando Pos valga 8 se revisará al Niño (1) (8) o sea el octavo niño de la primera fila
Cuando Pos valga 9 se revisará al Niño (1) (9) o sea el noveno niño de la primera fila
Cuando Pos valga 10 se revisará al Niño (1) (10) o sea el décimo niño de la primera fila

Al llegar a esta punto, la variable Fila se incrementará en 1 y entonces mientras esta variable tiene el valor de 2, la variable Pos tendrá valores desde 1 hasta 10 luego

Cuando Pos valga 1 se revisará al Niño (2) (1) o sea el primer niño de la segunda fila
Cuando Pos valga 2 se revisará al Niño (2) (2) o sea el segundo niño de la segunda fila
Cuando Pos valga 3 se revisará al Niño (2) (3) o sea el tercer niño de la segunda fila
. . .
. . .
. . .
Cuando Pos valga 9 se revisará al Niño (2) (9) o sea el noveno niño de la segunda fila
Cuando Pos valga 10 se revisará al Niño (2) (10) o sea el décimo niño de la segunda fila

Cuando se llegue a este punto entonces la variable Fila se incrementará en 1 y entonces mientras esta variable tiene el valor de 3, la variable Pos tendrá valores desde 1 hasta 10, luego

Cuando Pos valga 1 se revisará al Niño (3) (1) o sea el primer niño de la tercera fila
Cuando Pos valga 2 se revisará al Niño (3) (2) o sea el segundo niño de la tercera fila
Cuando Pos valga 3 se revisará al Niño (3) (3) o sea el tercer niño de la tercera fila
. . .
. . .
. . .
Cuando Pos valga 9 se revisará al Niño (3) (9) o sea el noveno niño de la tercera fila
Cuando Pos valga 10 se revisará al Niño (3) (10) o sea el décimo niño de la tercera fila

De nuevo en este momento se incrementaría el contenido de la variable Fila en 1 y por lo tanto almacenaría el valor 4. De manera que mientras esta variable sea igual a 4 la variable Pos tomará valores desde 1 hasta 10 por lo tanto

Cuando Pos valga 1 se revisará al Niño (4) (1) o sea el primer niño de la cuarta fila
Cuando Pos valga 2 se revisará al Niño (4) (2) o sea el segundo niño de la cuarta fila
Cuando Pos valga 3 se revisará al Niño (4) (3) o sea el tercer niño de la cuarta fila
. . .
. . .
. . .
Cuando Pos valga 9 se revisará al Niño (4) (9) o sea el noveno niño de la cuarta fila
Cuando Pos valga 10 se revisará al Niño (4) (10) o sea el décimo niño de la cuarta fila

Finalmente mientras la variable Fila vale 5, la variable Pos tomará valores desde 1 hasta 10 y por lo tanto

Cuando Pos valga 1 se revisará al Niño (5) (1) o sea el primer niño de la quinta fila
Cuando Pos valga 2 se revisará al Niño (5) (2) o sea el segundo niño de la quinta fila
Cuando Pos valga 3 se revisará al Niño (5) (3) o sea el tercer niño de la quinta fila
. . .
. . .
. . .
Cuando Pos valga 9 se revisará al Niño (5) (9) o sea el noveno niño de la quinta fila
Cuando Pos valga 10 se revisará al Niño (5) (10) o sea el décimo niño de la quinta fila que correspondería dentro del ejemplo al último niño de la escuela.

Características

Fundamentalmente los índices se conciben como números estrictamente enteros debido a que referenciar una posición dentro de un arreglo siempre se dará en términos enteros. Cuando usted hace una cola en un banco usted puede quedar de primero, de segundo, de tercero, etc. o lo que se lo mismo puede quedar de 1, de 2, de 3, etc. Lo que no se puede concebir es que usted quede de 1.4 ó de 4.6 dentro de la cola. Por esta razón es que se han conceptualizado los índices como datos estrictamente enteros.

Asimismo debido a que normalmente los datos de un arreglo se manejan como un conjunto se acostumbra, por facilidad y flexibilidad de los algoritmos, trabajar los índices a través de variables de tipo entero. Facilitan las expresiones en donde se involucren los elementos de los arreglos y flexibilizan los algoritmos debido a que cambiar el tope final de un ciclo, cuyos valores sirven para que se generen las posiciones correspondientes de un arreglo, es suficiente para que el mismo algoritmo sirva para un arreglo con otras dimensiones.

Tal como se ha visto es muy normal, también por conveniencia técnica, que se utilicen activamente ciclos para facilitar el manejo de esas variables que actuarán como índices.

Vectores

Características

Un vector es un arreglo en donde la ubicación *exacta* de cada uno de sus elementos necesita solamente la utilización de un subíndice. Tal es el ejemplo de una cola de personas en donde cada una de ellas se puede ubicar *exactamente* con un solo numerito: "Venga la quinta persona", "Acérquese la tercera persona", etc. He resaltado en letra cursiva la palabra *exacta* y *exactamente* porque ellas representan la diferencia con los otros tipos de arreglos.

Un vector siempre tendrá:

1. ***Tipo .-*** Por lo que se ha dicho en un vector los datos que se han de almacenar siempre serán del mismo tipo por lo cual es supremamente importante especificar de qué tipo van a ser los datos almacenados en él. No se olvide que los tipos de datos standard son Entero, Real y Carácter, cada uno con características propias que fueron explicadas en los primeros capítulos. En esta parte es importante anotar que el conjunto de datos almacenado en un vector siempre será *homogéneo* o sea que todos los datos son del mismo tipo.

2. ***Nombre.-*** Sabiendo que todos los datos almacenados en un vector van a pertenecer a un mismo arreglo, entonces dicho arreglo deberá tener un nombre ajustado a las mismas reglas con que se le colocan los nombres a las variables convencionales. No ha de olvidarse que un arreglo en el fondo es una variable dividida en varios pedacitos donde cada pedacito puede almacenar un dato diferente en su contenido mas no es su tipo.

3. ***Dimensión.-*** Se refiere a la cantidad de elementos que van a ser utilizados en el vector. Dimensionar un vector significa definir cuántos elementos se van a utilizar. En este aspecto debemos conocer lo mejor posible el objetivo porque cuando el algoritmo se haya convertido en programa y se encuentre en ejecución no podremos cambiar la cantidad de elementos de que consta dicho vector. Esa tal vez será una de las desventajas de la utilización de los arreglos en general y es que su dimensionamiento nos va a arriesgar a subdimensionar o a sobredimensionar, es decir, a definir mas campos de memoria o menos campos de memoria de los que necesitemos.

4. ***Tamaño.-*** El tamaño es la cantidad total de campos de memoria que van a ser utilizados. En el caso de los vectores no existe ninguna diferencia entre la dimensión y el tamaño, diferencia que se hará muy clara cuando expliquemos las matrices.

5. ***Destinación.-*** Es muy importante, tal como lo hicimos con el objetivo, que sepamos cuál va a ser la utilización del vector que vayamos a usar. Así como para nosotros en algún algoritmo era claro que la variable *Cont_Par* era la que iba a contener la cantidad de números pares que se generaran en algún proceso, así también va a ser muy importante que sepa con claridad cuál va a ser el uso de un determinado vector. De esta manera siempre que hagamos referencia al vector sabremos a qué conjunto de datos nos estamos refiriendo.

6. ***Indice.-*** Siempre que vamos a usar un vector es natural pensar que necesitaremos una variable de tipo entero que será utilizada a manera de subíndice o sea que es la que nos va a almacenar cada una de las posiciones posibles que tenga el vector. Dicha variable solo tiene que cumplir con el requisito de ser una variable entera. Es importante que recuerde que por ser el subíndice una variable, ésta, dentro del programa podrá tener todos los usos que le queramos dar dentro del contexto del mismo algoritmo además de ser utilizada como subíndice.

Justificación

Se justifica la utilización de vectores cuando se dan algunas de las siguientes razones:

1. En algún momento se necesita manejar una cantidad de datos como todo un conjunto.
2. Se necesitan almacenar datos que posteriormente se van a volver a utilizar.
3. Se necesitan realizar cálculos de manera que los resultados progresivos se vayan a necesitar mas adelante.
4. Se necesita realizar un determinado proceso con un conjunto de datos, “al tiempo”.
5. Se necesita realizar cálculos tan complejos que resulte mas óptimo almacenar los resultados provisionales que volverlos a calcular.
6. En general cada que necesitemos hacer operaciones con conjuntos de datos.
7. Siempre que se necesite desarrollar algoritmos con conjuntos de datos cuya cantidad pueda en algún momento, no dentro de una misma ejecución, cambiar.

Vamos a desarrollar un algoritmo ineficiente, sin tener en cuenta el concepto de vectores, para que posteriormente usted note la diferencia y entienda la gran utilidad que tienen estos dentro de las aplicaciones de programación.

Ejemplo Ineficiente sin Vectores No.1

Leer 10 números enteros y determinar cuál es el promedio entero de dichos números.

Clarificación del Objetivo

Como vamos a desarrollar una solución sin tener en cuenta el concepto de vectores entonces lo primero que vemos fácilmente es que necesitamos por lo menos 10 variables que nos permitan almacenar cada uno de los números leídos. Igualmente necesitaremos otra variable que nos permita almacenar el promedio entero de dichos números. En el contexto del algoritmo en sí lo que vamos a hacer es que vamos a leer 10 números enteros, cada uno se almacenará en una variable diferente, y cuando se hayan leído todos los números, los sumaremos y dividiremos entre 10 dicha suma, almacenando el resultado final en la variable destinada para tal fin y escribiendo su contenido en pantalla.

Algoritmo

```
Programa Ejem_Inef_sin_Vect_1
Var
        Entero :a,          // Almacenará el primer número leído
                b,          // Almacenará el segundo número leído
                c,          // Almacenará el tercer número leído
                d,          // Almacenará el cuarto número leído
                e,          // Almacenará el quinto número leído
                f,          // Almacenará el sexto número leído
                g,          // Almacenará el séptimo número leído
                h,          // Almacenará el octavo número leído
                i,          // Almacenará el noveno número leído
                j,          // Almacenará el décimo número leído
                Prom        // Almacenará el promedio entero de todos los
                            // números leídos

Inicio
        Escriba "Digite 10 números enteros"   // Avise que va a leer 10 números enteros
        Lea a                                 // y léalos
        Lea b
        Lea c
        Lea d
        Lea e
        Lea f
        Lea g
        Lea h
        Lea i
        Lea j

        Prom = 0                              // Inicialice la variable Prom en Ceros

        Prom = Prom + a                       // Sume progresivamente cada uno de los
                                              //valores leídos
        Prom = Prom + b
        Prom = Prom + c
        Prom = Prom + d
        Prom = Prom + e
        Prom = Prom + f
        Prom = Prom + g
        Prom = Prom + h
        Prom = Prom + i
        Prom = Prom + j

        Prom = Prom / 10                      // Calcule el promedio de los números
                                              // leídos
        Escriba Prom                          // Escriba dicho promedio en pantalla
Fin
```

También hubiéramos podido escribir este algoritmo de la siguiente forma

```
Programa Ejem_Inef_sin_Vect_1
Var
        Entero :a,              // Almacenará el primer número leído
                  b,            // Almacenará el segundo número leído
                  c,            // Almacenará el tercer número leído
                  d,            // Almacenará el cuarto número leído
                  e,            // Almacenará el quinto número leído
                  f,            // Almacenará el sexto número leído
                  g,            // Almacenará el séptimo número leído
                  h,            // Almacenará el octavo número leído
                  i,            // Almacenará el noveno número leído
                  j,            // Almacenará el décimo número leído
                  Prom          // Almacenará el promedio entero de todos los
                                // números leídos

Inicio
        Escriba "Digite 10 números enteros"     // Avise que va a leer 10
                                                // números enteros y léalos
        Lea a, b, c, d, e, f, g, h, i, j

        Prom = ( a + b + c + d + e + f + g + i + j + k) / 10// Calcule el promedio

        Escriba Prom                            // Escríbalo en pantalla

Fin
```

Resulta ser tan simplificada la prueba de escritorio de este algoritmo que a simple vista podemos ver que está bien, es decir, vemos que cumple plenamente con el objetivo planteado. Sin embargo, a pesar de que eso es verdad, este algoritmo tiene como desventaja principal el hecho de que solamente sirve para calcular el promedio de 10 números enteros digitados por el usuario. Usted tal vez dirá que ese precisamente era el objetivo a cumplir, pero sabiendo la gran utilidad que tiene el cálculo de un promedio en cualquier momento sería muy útil tener un algoritmo mas flexible o sea que permitiera calcular el promedio de 10 datos o de 25 datos o de cualquier cantidad de datos.

Si se quisiera ajustar este algoritmo para que permita leer 15 números enteros y calcular su promedio entonces tendríamos que aumentar otras cinco variables y ajustar los cálculos correspondientes en lo que se refiere al promedio. En ese caso el cambio puede no ser mucho pero que tal que se quiera lograr el mismo objetivo pero con 1000 datos enteros. Entonces allí sí tendríamos un verdadero problema porque a nuestra solución inicial tendríamos que adicionarle 990 variables mas y hacer los ajustes correspondientes para que en los cálculos el algoritmo tuviera en cuenta el resto de variables. Como puede ver este algoritmo a pesar de que cumple con el objetivo es demasiado rígido y por lo tanto es muy poco flexible lo cual lo hacer ser un algoritmo altamente ineficiente.

Entonces como haríamos para que además de cumplirse el objetivo se logre obtener un algoritmo eficiente...?

```
Programa Ejem_Efic_con_Vect_1
Variables
        Entero :Vector ( 10 ),          // Almacenará los 10 datos enteros que
                                        // se lean
                Indice,                 // Servirá como variable subíndice
                Promedio                // Almacenará el promedio entero de
                                        // los números leídos
Inicio
        Escriba "Digite 10 números enteros"     // Avisa que va a leer 10 enteros

        Para Indice = 1 hasta 10                // y los lee
                Lea Vector ( Indice )
        Fin_Para

        Promedio = 0                            // Inicializa el promedio en cero

        Para Indice = 1 hasta 10                // Acumula provisionalmente  todos los
                                                // valores en la variable Promedio
                Prom = Prom + Vector ( Indice )
        Fin_Para

        Prom = Prom / 10                        // Calcula el promedio como tal

        Escriba "El promedio entero es ", Prom  // Muestra el promedio en pantalla
Fin
```

Usted podrá notar que la estructura de este algoritmo es la misma pero es verdaderamente mas eficiente que la versión anterior debido a que si queremos que este algoritmo sirva para calcular el promedio entero de 1000 números todo lo que tenemos que hacer es cambiar el número 10 por el número 1000 tantas veces como aparezca. Sin embargo podemos hacerlo todavía mas eficiente de la siguiente forma:

```
Programa Ejem_Efic_con_Vect_1
Variables
        Entero :Vector ( 10 ),          // Almacenará los 10 datos enteros que
                                        // se lean
                Indice,                 // Servirá como variable subíndice
                Promedio,               // Almacenará el promedio entero de
                                        // los números leídos
                Tope                    // Almacenará la cantidad de números
                                        // a leer
Inicio
        Tope = 10
        Escriba "Digite ", Tope, " números enteros"     // Avisa que va a leer 10 enteros

        Para Indice = 1 hasta Tope              // y los lee
                Lea Vector ( Indice )
        Fin_Para

        Promedio = 0                            // Inicializa el promedio en cero

        Para Indice = 1 hasta Tope              // Acumula provisionalmente  todos los
                                                // valores en la variable Promedio
                Prom = Prom + Vector ( Indice )
```

```
        Fin_Para

        Prom = Prom / Tope                  // Calcula el promedio como tal

        Escriba "El promedio entero es ", Prom      // Muestra el promedio en pantalla
Fin
```

Ahora note usted que para hacer que esta nueva versión del mismo algoritmo nos permita leer 1000 números enteros y calcularles su promedio entonces solo tendremos que cambiar la instrucción

```
Tope = 10
```

Por

```
Tope = 1000
```

Y será suficiente para que nuestro algoritmo haya quedado en condiciones de cumplir el mismo objetivo pero con una cantidad diferente de datos. Esa es la esencia de la utilización de los vectores (y en general de los arreglos) que nos permiten desarrollar unos algoritmos altamente flexibles.

En este último algoritmo sé que usted puede pensar que siendo así sería mucho mas fácil que el mismo usuario digitara la cantidad de datos a leer. Pero tenga en cuenta que como la declaración de variables es lo primero que se hace en el algoritmo entonces de todas maneras tendríamos que dimensionar un vector de una cantidad grande de posiciones enteras (por ejemplo un vector de 50000 posiciones) con lo cual es posible que estemos dentro de las necesidades del usuario. Sin embargo dimensionar un vector tan grande nos obliga a pensar que si el usuario solo va a necesitar 10 posiciones entonces de alguna manera se van a desperdiciar las 49990 demás posiciones y si el usuario, por casualidad, necesita mas de 50000 datos entonces este vector no nos va a servir.

Esta es la desventaja que en renglones anteriores mencionábamos acerca de los arreglos en general y es que, como hay que dimensionarlos al momento de su declaración, corremos el riesgo de separar mucha mas memoria de la que necesitamos o mucha menos. Ahora sí con esta conceptualización veamos un ejemplo eficiente de utilización de vectores.

Ejemplo Con Vectores No.1

Desarrollar un programa que permita leer 10 números enteros y determinar en qué posición entró el número mayor.

Clarificación del Objetivo

Fundamentalmente el objetivo de este algoritmo radica en tener 10 números almacenados en memoria y determinar, asumiendo el orden de lectura, en qué posición está el mayor de los números leídos. Debe tenerse en cuenta que para saber cuál es el dato mayor de un conjunto de datos debemos tener dicho conjunto completo. Por ejemplo, para que usted sepa cuál es el mas alto de su familia entonces tendrá que haber conocido a toda su familia...cierto? Por esa misma razón, en este caso, "conocer" significa almacenar los datos por lo cual para poder determinar cuál es el mayor primero los debemos almacenar y tenerlos de manera que puedan ser manejables. Precisamente esto es lo que justifica la utilización de un vector en este ejercicio.

Algoritmo

Nunca olvide que los algoritmos que se presentan en este libro no son mas que una versión solución planteada por el autor. Recuerde que si usted desarrolla un algoritmo que cumpla el mismo objetivo de alguno de los algoritmos aquí presentados y nota que es diferente al que aparezca en este libro eso no quiere decir que su algoritmo esté mal o que el algoritmo de este libro esté mal. Sencillamente que son dos algoritmos diferentes que cumplen el mismo objetivo y que eso es completamente normal.

```
Programa Posic_Mayor
Variables
        Entero :Vector ( 10 ),          // Almacenará los 10 números enteros
                                        // que se van a leer
                Indice,                 // Servirá como variable subíndice
                Pos_May                 // Almacenará la posición del número
                                        // mayor que se vaya encontrando
                                        // provisionalmente y del número mayor
                                        // absoluto cuando se haya finalizado el
                                        // ciclo de búsqueda del mayor
Inicio
        Escriba "Digite 10 números enteros"     // Avisa que va a leer 10 números
                                                // enteros

        Para Indice = 1 hasta 10                // Genera un ciclo apoyado en la
                                                // variable índice que tomará valores
                                                // desde 1 hasta 10
                Lea Vector ( Indice )           // Lea un entero y guárdelo en el vector
                                                // en la posición que sea igual al valor
                                                // almacenado en la variable Indice
        Fin_Para                                // Fin del ciclo

        Pos_May = 1                             // Se inicializa la variable Pos_May con
                                                // el valor 1 para asumir,
                                                // provisionalmente, que el mayor se
                                                // encuentra en la primera posición. La
                                                // intención es ir comparando
                                                // progresivamente con este dato cada
```

```
                                        // uno de los restantes datos que se
                                        // encuentran almacenados en el vector
    Para Indice = 2 hasta 10            // Genera un ciclo desde 2 hasta 10
                                        // para comparar contra el primer dato
                                        // el resto de datos
        Si Vector (Indice) > Vector (Pos_May)  // Si el dato almacenado en el vector en
                                        // la posición donde vaya el índice en el
                                        // momento es mayor que el dato que
                                        // provisionalmente es el mayor
            Pos_May = Indice            // eso quiere decir que el mayor que
                                        // estaba en la posición que decía la
                                        // variable Pos_May ya no es el mayor
                                        // y que el nuevo mayor está en la
                                        // posición en donde está el índice por
                                        // lo cual la variable Pos_May debe ser
                                        // igual al contenido de la variable
                                        // Indice
        Fin_Si                          // Fin de la decisión
    Fin_Para                            // Fin del ciclo

                                        // Al final escribirá el valor solicitado y
                                        // es la posición en la cual está el
                                        // número mayor de entre los números
                                        // leídos. Esa posición está almacenada
                                        // en la variable Pos_May
    Escriba "El número mayor está en la posición ", Pos_May
Fin                                     // Fin del algoritmo
```

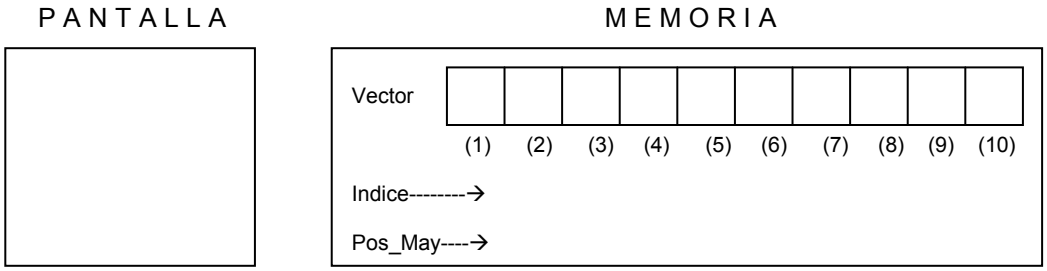
Prueba de Escritorio

Vamos a desarrollar una prueba paso a paso tal como la haría el computador internamente.

Como es natural pensar, lo primero que se hace en este algoritmo es declarar las variables o sea separar el espacio de memoria que se va a utilizar. Por tal motivo separamos en memoria espacio para 10 datos enteros que serán manejados a través del nombre de un vector y sus respectivas posiciones, espacio para una variables que se llamará *Indice* y otra que se llamará *Pos_may* y que almacenará la posición en la que se encuentre el número mayor.

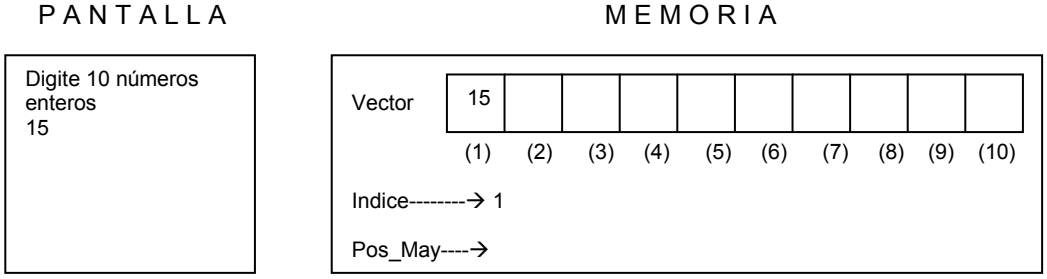
Para facilitar la distribución estética de las variables vamos a desarrollar la prueba de escritorio colocando los diferentes valores de cada variable horizontalmente.

```
Programa Posic_Mayor
Variables
    Entero :Vector ( 10 ),
                Indice,
                Pos_May
```

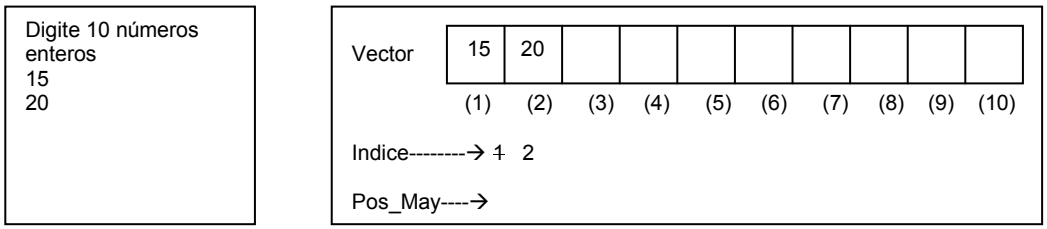


Nuestro algoritmo comienza anunciando en pantalla lo que se dispone a leer y, por supuesto, leyéndolo.

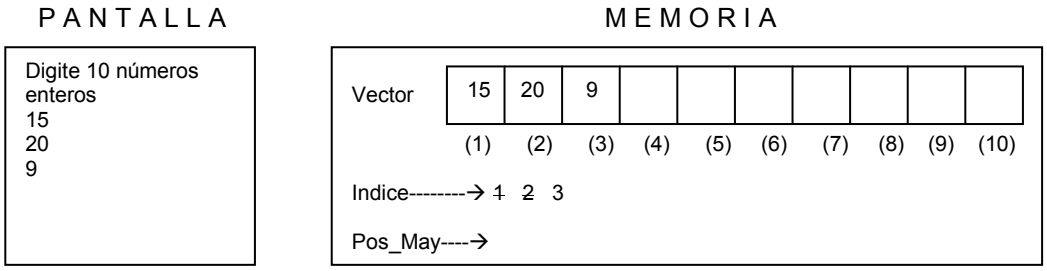
```
Inicio
        Escriba "Digite 10 números enteros"

        Para Indice = 1 hasta 10
                Lea Vector ( Indice )
        Fin_Para
```



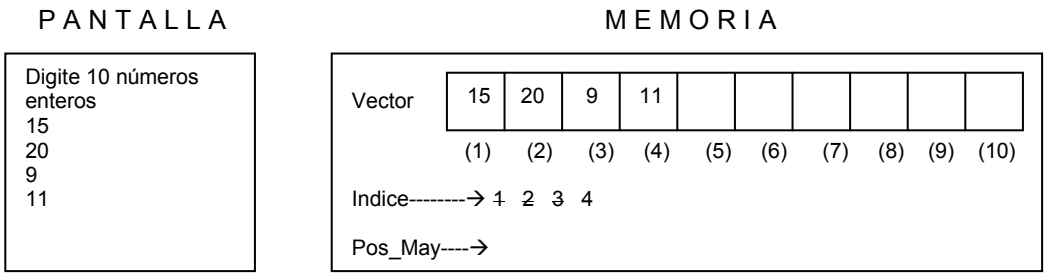
Se colocó en pantalla el título correspondiente, se inició el ciclo comenzando con la variable Indice en 1 (hemos de llevar hasta 10 el contenido de esta variable) y se recibió el primer número (supongamos que es 15). Como el contenido de la variable *Indice* es igual a 1 y la orden es *Lea Vector (Indice)* o sea *Lea un dato entero y guárdelo en Vector en la posición Indice* entonces el dato recibido se almacena en *Vector (1)*. Como se encuentra el fin del ciclo (*Fin_Para*) entonces se regresa a incrementar en 1 el valor de la variable *Indice* o sea que su contenido es igual a 2. Razón por la cual al leer el siguiente número éste quedará almacenado en el vector *Vector* en la posición 2. No se olvide que para efectos de la prueba de escritorio vamos a asumir algunos valores. Supongamos pues que el siguiente valor leído es igual a 20.



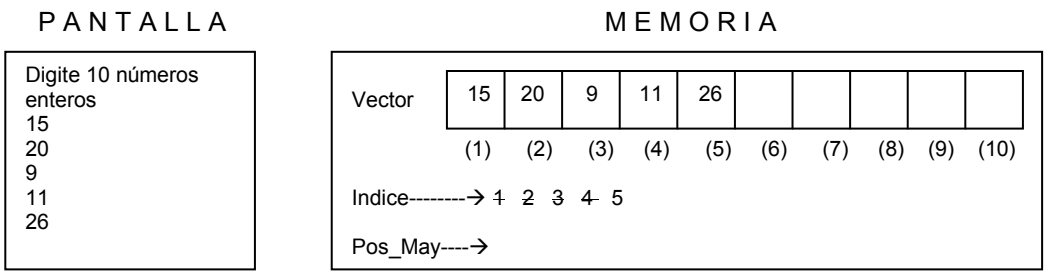
Cuando *Indice* valga 3 y asumiendo que el valor leído sea 9 entonces



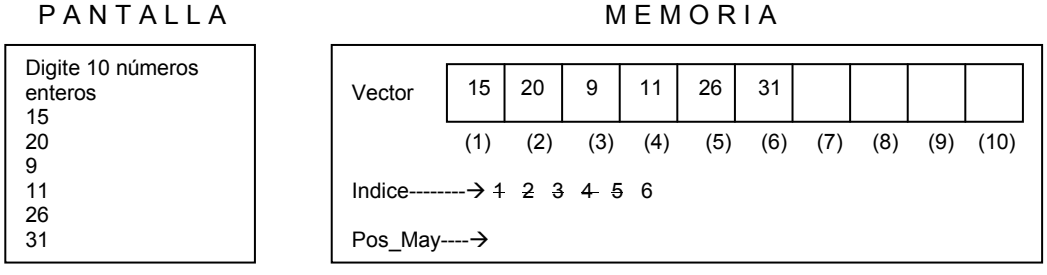
Cuando *Indice* valga 4 y asumiendo que el valor leído sea 11 entonces



Cuando *Indice* valga 5 y asumiendo que el valor leído sea 26 entonces



Cuando *Indice* valga 6 y asumiendo que el valor leído sea 31 entonces este valor se almacenará en el vector en la posición 6



Cuando *Indice* valga 7 y asumiendo que el valor leído sea 19 entonces este valor se almacenará en el vector en la posición 7

P A N T A L L A

.
.
.
26
31
19

M E M O R I A

Vector	15	20	9	11	26	31	19			
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Indice--------→ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ 7

Pos_May----→

Cuando *Indice* valga 8 y asumiendo que el valor leído sea 20 entonces este valor se almacenará en el vector en la posición 8

P A N T A L L A

.
.
.
26
31
19
20

M E M O R I A

Vector	15	20	9	11	26	31	19	20		
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Indice--------→ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ 8

Pos_May----→

Cuando *Indice* valga 9 y asumiendo que el valor leído sea 6 entonces este valor se almacenará en el vector en la posición 9

P A N T A L L A

.
.
.
19
20
6

M E M O R I A

Vector	15	20	9	11	26	31	19	20	6	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Indice--------→ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ 9

Pos_May----→

Y cuando la variable *Indice* valga 10, asumiendo que el valor leído sea 8 entonces este valor se almacenará en el vector en la posición 10. Además como la variable *Indice* llega al tope planteado inicialmente entonces la ejecución continúa con el resto de instrucciones, obviamente saliéndose del ciclo.

P A N T A L L A

.
.
.
19
20
6
8

M E M O R I A

Vector	15	20	9	11	26	31	19	20	6	8
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

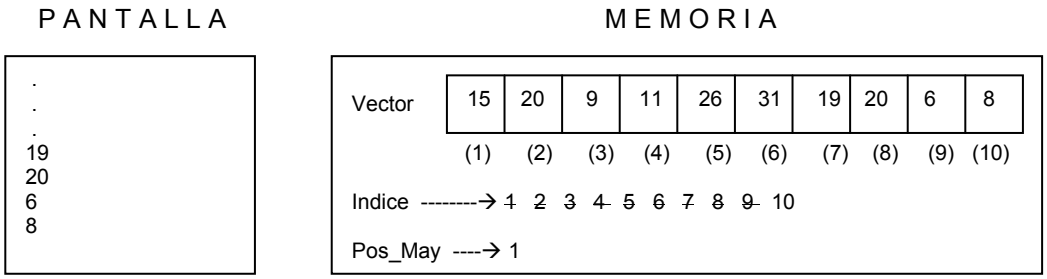
Indice--------→ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ 10

Pos_May----→

Como puede ver de esta manera queda “cargado” con datos el vector y todo lo que tuvimos que hacer fue utilizar apropiadamente un ciclo para que a través de una variable se controlaran las posiciones en donde progresivamente se iban a ir almacenando cada uno de los valores digitados.

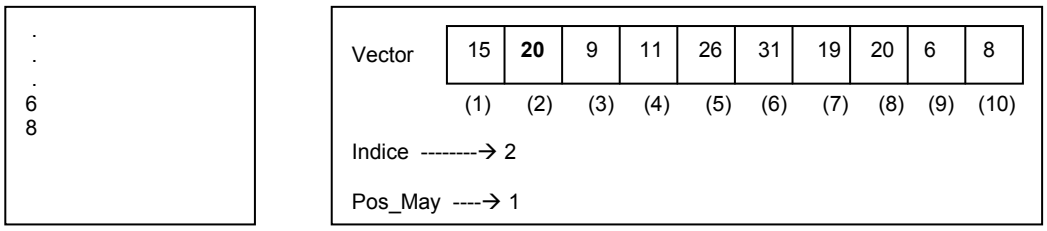
El algoritmo continúa inicializando la variable *Pos_May* con el valor de 1.

```
Pos_May = 1
```



Luego continúa realizando un ciclo *Para* utilizando la variable *Indice* que va a tomar valores desde 2 hasta 10 y que va a servir como referencia para compara desde el elemento *Sub 2* hasta el elemento *Sub 10* del vector con el primer elemento.

```
Para Indice = 2 hasta 10
        Si Vector (Indice) > Vector (Pos_May)
                Pos_May = Indice
        Fin_Si
Fin_Para
```



Cuando la variable *Indice* valga 2 entonces la pregunta

```
Si Vector  ( Indice ) > Vector ( Pos_May )
```

se traducirá en

```
Si Vector ( 2 ) > Vector ( 1 )
```

Dado que la variable *Pos_May* vale 1. Como Vector (2) es igual a 20 y Vector (1) es igual a 15 entonces internamente la pregunta se convierte en

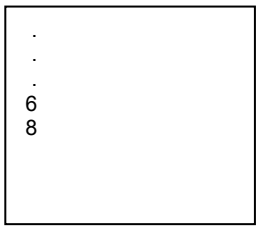
```
Si 20 > 15
```

Lo cual es Verdadero por lo tanto se ejecuta la orden

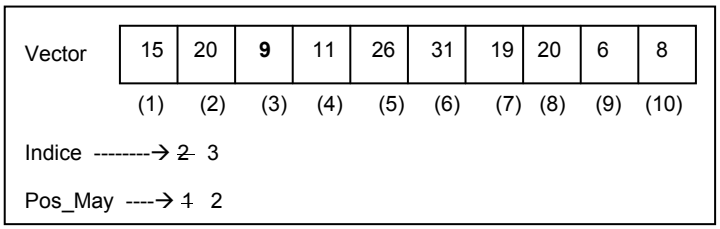
Pos_May = Indice

Con lo cual la variable Pos_May queda con el valor 2. Como a continuación encontramos tanto el fin de la decisión como el fin del ciclo entonces esto nos indica que debemos incrementar el contenido de la variable Indice en 1 con lo cual queda en dicha variable el valor 3.

P A N T A L L A



M E M O R I A



Vuelve a hacerse la pregunta

Si Vector (Indice) > Vector (Pos_May)

Como la variable Indice es igual a 3 y la variable Pos_May es igual a 2 entonces esta pregunta se convierte en

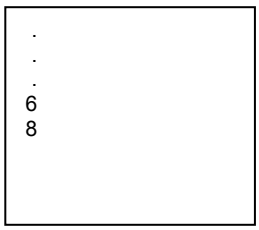
Si Vector (3) > Vector (2)

Por los valores almacenados en el vector vemos que la pregunta se convierte en

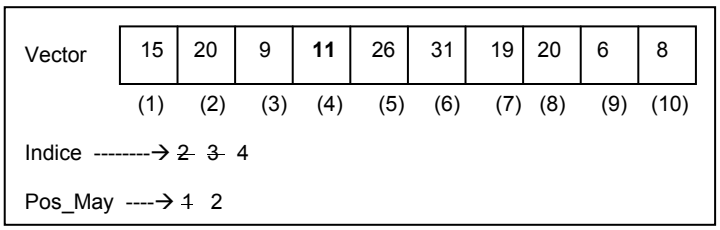
Si 9 > 20

Como es falso, y dado que a continuación de esta decisión sigue el fin de ella y el fin del ciclo, entonces volvemos a incrementar el valor de la variable Indice en 1.

P A N T A L L A



M E M O R I A



De la misma manera volvemos a desarrollar la decisión

Si Vector (Indice) > Vector (Pos_May)

Como la variable *Indice* es igual a 4 y la variable *Pos_May* es igual a 2 entonces esta pregunta se convierte en

Si Vector (4) > Vector (2)

Por los valores almacenados en el vector vemos que la pregunta se convierte en

Si 11 > 20

Note usted, a esta altura de la prueba de escritorio, que este algoritmo va comparando cada número con el último que haya encontrado como mayor. Como esta última decisión también es Falsa, y luego de haber encontrado el fin de la decisión y el fin el ciclo, entonces volvemos a incrementar en 1 el valor almacenado en la variable *Indice*.

PANTALLA

```
.
.
.
6
8
```

MEMORIA

Vector	15	20	9	11	**26**	31	19	20	6	8
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Indice --------→ ~~2~~ ~~3~~ ~~4~~ 5

Pos_May ----→ ~~1~~ 2

Volvemos a hacer la pregunta

Si Vector (Indice) > Vector (Pos_May)

Como la variable *Indice* es igual a 5 y la variable *Pos_May* es igual a 2 entonces esta pregunta se convierte en

Si Vector (5) > Vector (2)

Por los valores almacenados en el vector vemos que la pregunta se convierte en

Si 26 > 20

Como es Verdadero entonces se ejecuta la asignación

Pos_May = Indice

Con lo cual en la variable *Pos_May* queda almacenado el valor 5. Como luego de esta asignación encontramos el fin de la decisión y el fin del ciclo entonces incrementamos de nuevo el valor de la variable Indice en 1

PANTALLA

```
.
.
.
6
8
```

MEMORIA

Vector	15	20	9	11	26	**31**	19	20	6	8
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Indice --------→ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

Pos_May ----→ ~~1~~ ~~2~~ 5

Volvemos a ejecutar la decisión

Si Vector (Indice) > Vector (Pos_May)

Como la variable *Indice* es igual a 6 y la variable *Pos_May* es igual a 5 entonces esta pregunta se convierte en

Si Vector (6) > Vector (5)

Por los valores almacenados en el vector vemos que la pregunta se convierte en

Si 31 > 26

Como vemos que es Verdadero entonces almacenamos en la variable *Pos_May* el contenido almacenado en la variable Indice. Volvemos entonces a incrementar el contenido de la variable Indice en 1

P A N T A L L A

```
.
.
.
6
8
```

M E M O R I A

Vector	15	20	9	11	26	31	**19**	20	6	8
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Indice --------→ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ 7

Pos_May ----→ ~~1~~ ~~2~~ ~~5~~ 6

Ejecutamos la decisión

Si Vector (Indice) > Vector (Pos_May)

Como la variable *Indice* es igual a 7 y la variable *Pos_May* es igual a 6 entonces esta pregunta se convierte en

Si Vector (7) > Vector (6)

Por los valores almacenados en el vector vemos que la pregunta se convierte en

Si 19 > 31

Como es Falso entonces volvemos a incrementar el valor almacenado en la variable Indice

P A N T A L L A

```
.
.
.
6
8
```

M E M O R I A

Vector	15	20	9	11	26	31	19	**20**	6	8
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Indice --------→ ~~2~~ ~~3~~ 4 ~~5~~ ~~6~~ ~~7~~ 8

Pos_May ----→ ~~1~~ ~~2~~ ~~5~~ 6

Ejecutamos de nuevo la decisión

Si Vector (Indice) > Vector (Pos_May)

Como la variable *Indice* es igual a 8 y la variable *Pos_May* es igual a 6 entonces esta pregunta se convierte en

Si Vector (8) > Vector (6)

Por los valores almacenados en el vector vemos que la pregunta se convierte en

Si 20 > 31

Igualmente al caso anterior, esta decisión es Falsa, razón por la cual, y luego de haber encontrado el fin de la decisión y el fin del ciclo, incrementamos el contenido de la variable Indice

P A N T A L L A

```
.
.
.
6
8
```

M E M O R I A

Vector	15	20	9	11	26	31	19	20	**6**	8
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Indice --------→ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ 9

Pos_May ----→ ~~1~~ ~~2~~ ~~5~~ 6

Se realizar una vez mas la decisión

Si Vector (Indice) > Vector (Pos_May)

Como la variable *Indice* es igual a 9 y la variable *Pos_May* es igual a 6 entonces esta pregunta se convierte en

Si Vector (9) > Vector (6)

Por los valores almacenados en el vector vemos que la pregunta se convierte en

Si 2 > 31

Como es Falsa la respuesta, entonces incrementamos de nuevo el contenido de la variable Indice.

P A N T A L L A

```
.
.
.
6
8
```

M E M O R I A

Vector	15	20	9	11	26	31	19	20	6	**8**
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Indice --------→ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ 10

Pos_May ----→ ~~1~~ ~~2~~ ~~5~~ 6

Se ejecuta de nuevo la decisión

Si Vector (Indice) > Vector (Pos_May)

Como la variable *Indice* es igual a 10 y la variable *Pos_May* es igual a 6 entonces esta pregunta se convierte en

Si Vector (10) > Vector (6)

Por los valores almacenados en el vector vemos que la pregunta se convierte en

Si 8 > 31

Cuya respuesta también es Falsa. Con esto hemos llegado al final del ciclo pues inicialmente el ciclo planteaba la generación de números entre 2 y el número 10 para la variable Indice. Como esta variable ya vale 10 y se ha ejecutado con este valor el ciclo, entonces continuamos con la instrucción que se encuentra después del *Fin_Para* que representa el fin del ciclo y que es la instrucción que nos muestra el resultado solicitado en pantalla.

Escriba "El número mayor está en la posición ", Pos_May

Con lo cual se escribiría en pantalla lo que está entre comillas dobles seguido por el contenido de la variable Pos_May

P A N T A L L A

```
.
.
.
6
8
El número mayor está
en la posición 6
```

M E M O R I A

Vector	15	20	9	11	26	31	19	20	6	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Indice --------→ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ 10

Pos_May ----→ ~~1~~ ~~2~~ ~~5~~ 6

Por último lo que encontramos es el fin del algoritmo

Fin

Hemos terminado la prueba de escritorio y solo nos queda determinar si el algoritmo si cumplió su objetivo inicial que era leer 10 números enteros y determinar en que posición estaba el mayor de los números leídos. Vemos que efectivamente el número 31 es el mayor de los números que se encuentran almacenados en el vector y que corresponden a los números leídos. Con esto podemos concluir que el algoritmo está correcto y que es solución al objetivo planteado inicialmente.

Y qué tal que se quisiera decir, además de la posición del mayor, cuál es el número mayor...? Entonces el algoritmo que originalmente es así

```
Programa Posic_Mayor
Variables
        Entero :Vector ( 10 ),
                        Indice,
                        Pos_May
Inicio
        Escriba "Digite 10 números enteros"
```

```
        Para Indice = 1 hasta 10
                Lea Vector ( Indice )
        Fin_Para

        Pos_May = 1

        Para Indice = 2 hasta 10
                Si Vector (Indice) > Vector (Pos_May)
                        Pos_May = Indice
                Fin_Si
        Fin_Para

        Escriba "El número mayor está en la posición ", Pos_May
Fin
```

Deberá quedar así (he resaltado los cambios para que se noten mas)

```
Programa Posic_Mayor
Variables
        Entero :Vector ( 10 ),
                        Indice,
                        Pos_May,
                        Auxiliar                    //Variable que almacenará el número
                                                    // que se vaya encontrando como
                                                    // mayor
Inicio
        Escriba "Digite 10 números enteros"

        Para Indice = 1 hasta 10
                Lea Vector ( Indice )
        Fin_Para

        Pos_May = 1
        Auxiliar = Vector ( 1 )                     // Almacena el contenido que hay en la
                                                    // primera posición del vector

        Para Indice = 2 hasta 10
                Si Vector (Indice) > Vector (Pos_May)
                        Pos_May = Indice
                        Auxiliar = Vector ( Indice )    // Guarda provisionalmente en la
                                                        // variable Auxiliar el valor almacenado
                                                        // en el Vector en la posición Indice
                Fin_Si
        Fin_Para

        Escriba "El número mayor es ", Auxiliar, " y está en la posición ", Pos_May
                                                    // Muestra el número mayor y la
                                                    // posición en la cual se encuentra
Fin
```

Y si el objetivo hubiera sido determinar si el número mayor de 10 números leídos es par...? Entonces los cambios también serían muy sencillos. La última línea de escritura en donde está

```
Escriba "El número mayor es ", Auxiliar, " y está en la posición ", Pos_May
                                                    // Muestra el número mayor y la
                                                    // posición en la cual se encuentra
```

se reemplazaría por

```
Si Auxiliar / 2 * 2 = Auxiliar
    Escriba "El número mayor es ", Auxiliar, " y es par "
                                                    // Muestra el número mayor y la
                                                    // posición en la cual se encuentra
Sino
    Escriba "El número mayor es igual a ", Auxiliar, " y  no es par"
```

Como puede ver realizando algunos leves cambios en un algoritmo se pueden obtener otros y por lo tanto lograr otros objetivos.

Ejemplo Con Vectores No. 2

Leer 10 números enteros. Luego leer 1 número entero y determinar si este último número está entre los 10 primeros números leídos.

Clarificación del objetivo

De acuerdo al enunciado primero vamos a leer 10 datos enteros y los vamos a almacenar en un vector (obviamente de 10 posiciones). Luego de tener almacenados los 10 enteros vamos a leer otro número que será almacenado en otra variable. Se tratará entonces de preguntar si el contenido de la última variable es igual al contenido de alguna de las posiciones del vector. Si la respuesta a dicha pregunta es Verdadera entonces deberemos avisar por pantalla y asimismo en caso de que sea Falsa.

Algoritmo

No olvide que cada algoritmo presentado aquí es apenas una de las posibles soluciones que pueda tenerse para alcanzar el objetivo.

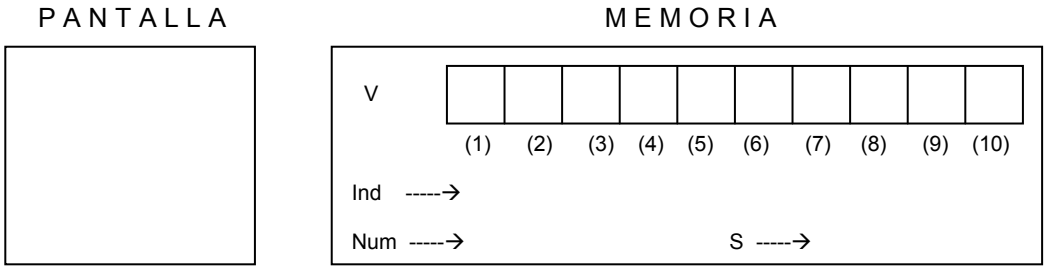
```
Programa Búsqueda
Variables
        Entero :V ( 10 ),                   // Almacenará los 10 números que el
                                                    // usuario va a digitar
                        Ind,                        // Esta es la variable que nos va a
                                                    // servir como subíndice del vector
                        Num,                        // Almacenará el número que se va a
                                                    // buscar
                        S                           // El contenido de esta variable nos va
                                                    // a indicar si el número se encontró o
                                                    // no.
Inicio
        Escriba "Digite 10 números enteros"         // Solicita los 10 números enteros

        Para Ind = 1 hasta 10 Paso 1                // Y los lee almacenando cada uno en
                                                    // una posición diferente del vector
                Lea V ( Ind )
        Fin_Para

        Escriba "Ahora digite un número"            // Solicita el número a buscar
        Lea Num                                     // y lo lee almacenándolo en la variable
                                                    // Num

        S = 0                                       // Inicializa la variable "misteriosa" en 0

        Para Ind = 1 hasta 10 Paso 1                // Recorre el vector desde la primera
                                                    // posición hasta la última preguntando
                                                    // si el contenido de cada posición es
                                                    // igual al contenido de la variable Num
                Si V ( Ind ) = Num
                        Escriba " El número ", Num, " si está en los 10 números digitados"
                        S = 1
                Fin_Si                              // Si dicha respuesta es Verdadera
                                                    // entonces lo muestra en pantalla y
                                                    // almacena en la variable "misteriosa" // el
                                                    valor 1
        Fin_Para

                                                    // Cuando haya terminado el ciclo
                                                    // pregunta por el valor almacenado en
                                                    // la variable S. Si este valor aún es
                                                    // cero quiere decir que no encontró el
                                                    // número buscado y por lo tanto lo
                                                    // avisa en pantalla
        Si S = 0
                Escriba "El número ", Num, " no está en los 10 números digitados"
        Fin_Si

Fin
```

Puede usted notar el uso de la variable S. Esta variable a lo largo de todo el algoritmo toma solo dos valores 1 ó 0. Esto representa que la variable es utilizada a manera de interruptor (o también llamada switche). Este forma de manejo de variables nos permite conocer respuestas que de otra forma sería muy difícil saberlas a nivel algorítmico.

Prueba de Escritorio

Vamos a desarrollar paso a paso esta prueba no sin antes recomendarle que observe, en el desarrollo de la misma, el recorrido gráfico. La prueba obviamente inicia con la declaración de variables en memoria

Programa Búsqueda
Variables
Entero :V (10),
Ind,
Num
S

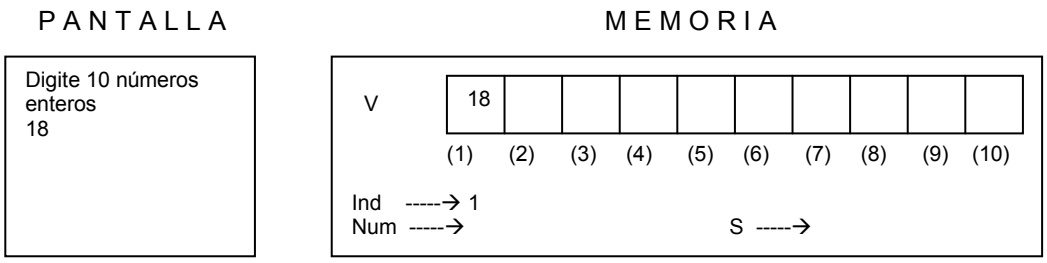


Lo primero que se hará en el algoritmo será "cargar" el vector con datos. Para ello en pantalla saldrá un título, a continuación se leerán los datos y cada dato numérico se almacenará en cada una de las "casillas" del vector.

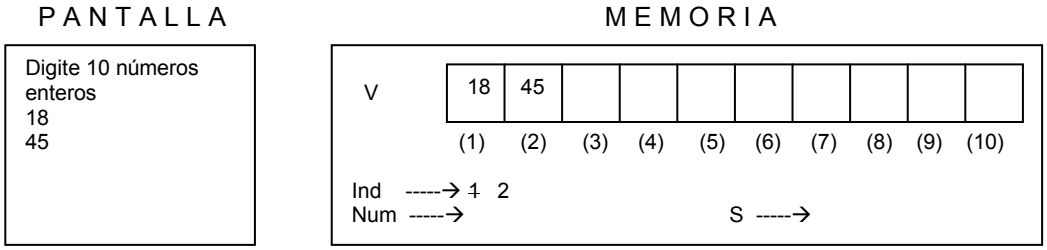
Inicio
Escriba "Digite 10 números enteros"

Para Ind = 1 hasta 10 Paso 1
Lea V (Ind)
Fin_Para

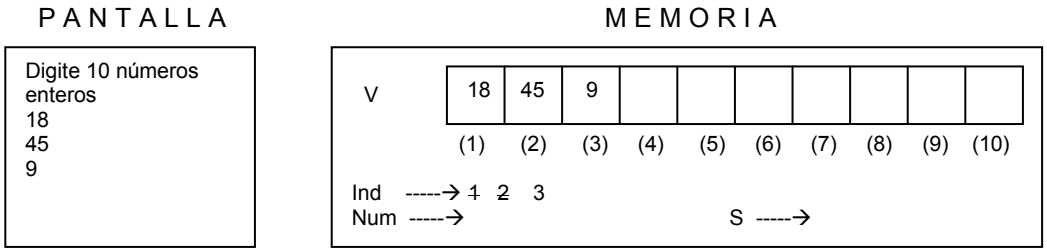
Se coloca el título en pantalla. Se inicia la variable *Ind* en 1 (e iremos incrementándola de 1 en 1 hasta llegar a 10) y a continuación se lee un entero y se almacena en el vector V en la posición 1. Supongamos que el número leído es 18 entonces



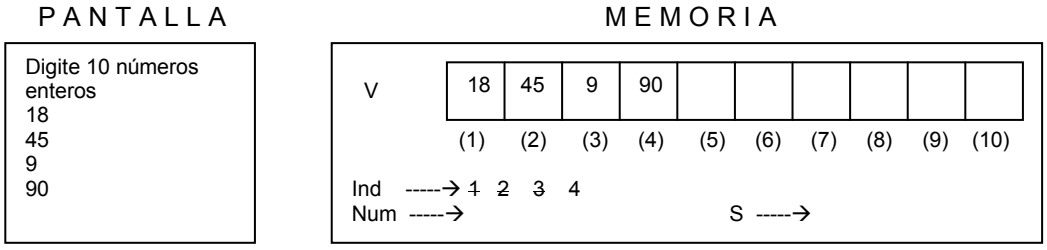
A continuación se incrementa el contenido de la variable *Ind* en 1 y se lee un nuevo dato almacenándose en el vector V en la posición 2. Supongamos que el número leído sea 45 entonces



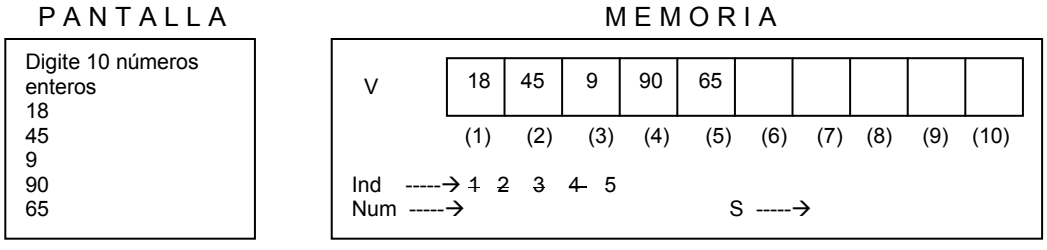
Incrementamos el contenido de la variable *Ind* en 1 y volvemos a leer otro dato almacenándolo en el vector V en la posición 3. Asumamos que el valor leído es 9.



Incrementamos el contenido de la variable *Ind* en 1 y volvemos a leer otro dato almacenándolo en el vector V en la posición 4. Asumamos que el valor leído es 90.



Volvemos a incrementar en 1 el contenido de la variable *Ind* y leemos un dato entero que quedará almacenado en el vector V en la posición 4. Asumamos que el valor leído es 65

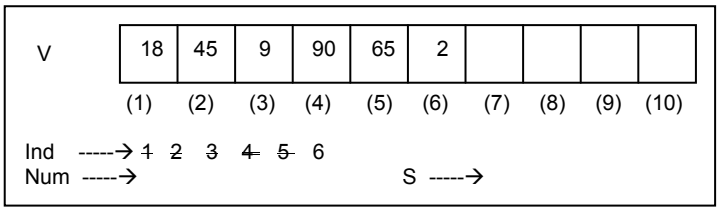


Se incrementa el contenido de la variable *Ind* la cual queda con el valor 6 y se lee un nuevo dato que se almacenará en el vector V en la posición 6. Asumamos que el valor leído es 2

P A N T A L L A

```
.
.
65
2
```

M E M O R I A

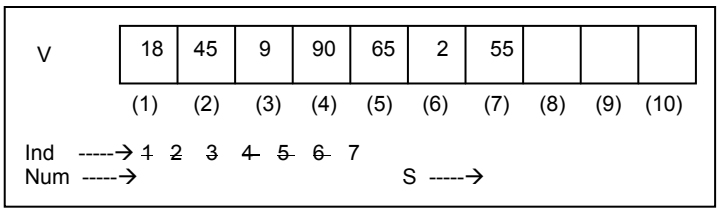


Incrementamos de nuevo el contenido de *Ind* en 1 y leemos un nuevo dato

P A N T A L L A

```
.
.
65
2
55
```

M E M O R I A

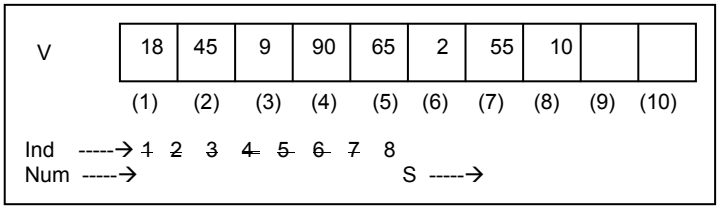


Volvemos a incrementar el contenido de la variable *Ind* y volvemos a leer un nuevo dato que quedará almacenado en el vector V en la posición 8.

P A N T A L L A

```
.
.
65
2
55
10
```

M E M O R I A

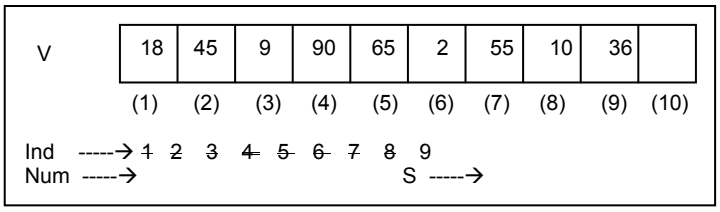


Incrementamos de nuevo el contenido de la variable *Ind* en 1 y leemos un dato entero que quedará almacenado en el vector V en la posición 9. Asumamos que el valor leído es 36

P A N T A L L A

```
.
.
65
2
55
10
36
```

M E M O R I A

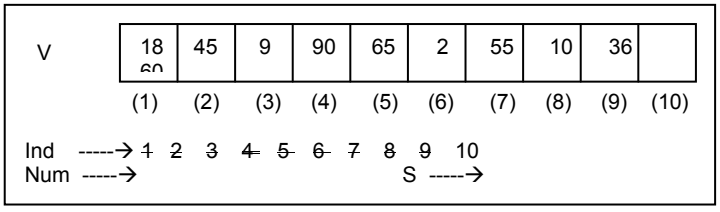


Por último incrementamos el contenido de la variable *Ind* y llegamos al tope (10). Leemos el último dato que va a ser almacenado en el vector V en la posición 10.

P A N T A L L A

```
.
.
.
10
36
60
```

M E M O R I A



Como hemos llegado al tope del ciclo entonces continuamos con las instrucciones que se encuentran después del *Fin_Para* correspondiente al ciclo que acabamos de terminar. Estas instrucciones son una orden de escritura para solicitar un número y la lectura de un número entero que se ha de almacenar en la variable *Num*. Supongamos que ese número leído es el número 10

Escriba "Ahora digite un número"
Lea Num

P A N T A L L A

.
.
.
60
Ahora digite un
número
10

M E M O R I A

V	18	45	9	90	65	2	55	10	36	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Ind -----→ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ 10
Num -----→ 10 S -----→

Inicializamos la variable S en 0 y generamos un ciclo que va desde 1 hasta 10 sobre la variable *Ind*, ciclo que nos servirá para comparar el contenido de la variable *Num* (que es el número que queremos buscar) con cada uno de los números almacenados en el vector V.

```
S = 0

Para Ind = 1 hasta 10 Paso 1
        Si V ( Ind ) = Num
                Escriba " El número ", Num, " si está en los 10 números digitados"
                S = 1
        Fin_Si
Fin_Para
```

P A N T A L L A

.
.
.
60
Ahora digite un
número
10

M E M O R I A

V	**18**	45	9	90	65	2	55	10	36	60
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Ind -----→ 1
Num -----→ 10 S -----→ 0

De esta forma cuando el contenido de la variable *Ind* sea 1 la decisión

Si V (Ind) = Num

se convertirá en

Si V (1) = Num

y como el vector V en la posición 1 almacena el número 18 y el contenido de la variable *Num* es 10 entonces la decisión internamente, en últimas, se convierte en

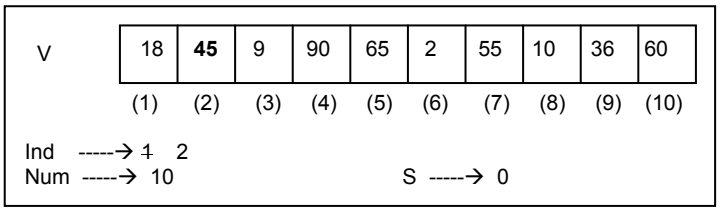
Si 18 = 10

Como la respuesta es Falso entonces, luego de encontrar el *Fin_Si* y el *Fin_Para*, volvemos a incrementar el contenido de la variable Ind en 1 y volvemos a hacer la pregunta correspondiente.

P A N T A L L A

```
.
.
Ahora digite un
número
10
```

M E M O R I A



De esta forma cuando el contenido de la variable *Ind* sea 2 la decisión

Si V (Ind) = Num

se convertirá en

Si V (2) = Num

y como el vector V en la posición 2 almacena el número 45 y el contenido de la variable *Num* es 10 entonces la decisión internamente, en últimas, se convierte en

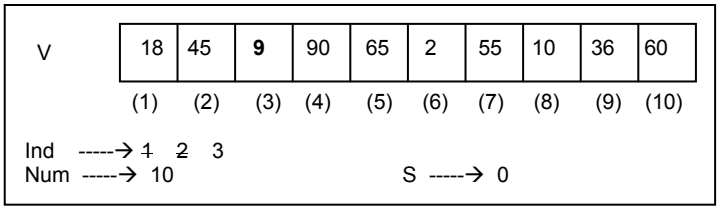
Si 45 = 10

Como la respuesta es Falso entonces, luego de encontrar el Fin_Si y el Fin_Para, volvemos a incrementar el contenido de la variable *Ind* en 1 y volvemos a hacer la pregunta correspondiente.

P A N T A L L A

```
.
.
Ahora digite un
número
10
```

M E M O R I A



De esta forma cuando el contenido de la variable *Ind* sea 3 la decisión

Si V (Ind) = Num

se convertirá en

Si V (3) = Num

y como el vector V en la posición 3 almacena el número 9 y el contenido de la variable *Num* es 10 entonces la decisión internamente, en últimas, se convierte en

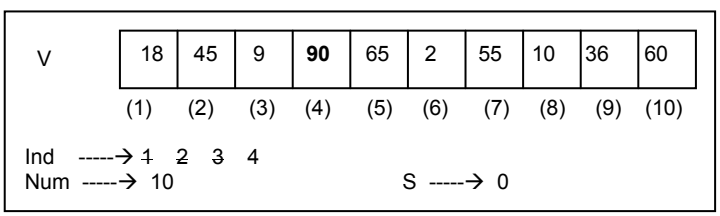
Si 9 = 10

Como la respuesta es Falso entonces, luego de encontrar el *Fin_Si* y el *Fin_Para*, volvemos a incrementar el contenido de la variable *Ind* en 1 y volvemos a hacer la pregunta correspondiente.

P A N T A L L A

```
.
.
Ahora digite un
número
10
```

M E M O R I A



Cuando el contenido de la variable *Ind* sea 4 la decisión *Si V (Ind) = Num*

se convertirá en *Si V (4) = Num*

y como el vector V en la posición 4 almacena el número 90 y el contenido de la variable *Num* es 10 entonces la decisión internamente, en últimas, se convierte en

Si 90 = 10

Como la respuesta es Falso entonces, luego de encontrar el *Fin_Si* y el *Fin_Para*, volvemos a incrementar el contenido de la variable *Ind* en 1 y volvemos a hacer la pregunta correspondiente.

P A N T A L L A

```
.
.
Ahora digite un
número
10
```

M E M O R I A

V	18	45	9	90	**65**	2	55	10	36	60
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Ind -----→ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5
Num -----→ 10 S -----→ 0

Cuando el contenido de la variable *Ind* sea 5 la decisión se convertirá en *Si V (Ind) = Num*
Si V (5) = Num

y como el vector V en la posición 5 almacena el número 65 y el contenido de la variable *Num* es 10 entonces la decisión internamente, en últimas, se convierte en

Si 65 = 10

Como la respuesta es Falso entonces, luego de encontrar el *Fin_Si* y el *Fin_Para*, volvemos a incrementar el contenido de la variable *Ind* en 1 y volvemos a hacer la pregunta correspondiente.

P A N T A L L A

```
.
.
Ahora digite un
número
10
```

M E M O R I A

V	18	45	9	90	65	**2**	55	10	36	60
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Ind -----→ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6
Num -----→ 10 S -----→ 0

Cuando el contenido de la variable *Ind* sea 6 la decisión se convertirá en *Si V (Ind) = Num*
Si V (6) = Num
y como el vector V en la posición 6 almacena el número 2 y el contenido de la variable *Num* es 10 entonces la decisión internamente, en últimas, se convierte en

Si 2 = 10

Como la respuesta es Falso entonces, luego de encontrar el *Fin_Si* y el *Fin_Para*, volvemos a incrementar el contenido de la variable *Ind* en 1 y volvemos a hacer la pregunta correspondiente.

P A N T A L L A

```
.
.
Ahora digite un
número
10
```

M E M O R I A

V	18	45	9	90	65	2	**55**	10	36	60
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Ind -----→ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ 7
Num -----→ 10 S -----→ 0

Cuando el contenido de la variable *Ind* sea 7 la decisión *Si V (Ind) = Num*
se convertirá en *Si V (7) = Num*

y como el vector V en la posición 7 almacena el número 55 y el contenido de la variable *Num* es 10 entonces la decisión internamente, en últimas, se convierte en *Si 55 = 10.* Como la respuesta es Falso entonces, luego de encontrar el *Fin_Si* y el *Fin_Para*, volvemos a incrementar el contenido de la variable *Ind* en 1 y volvemos a hacer la pregunta correspondiente.

P A N T A L L A

.
.
Ahora digite un
número
10

M E M O R I A

V	18	45	9	90	65	2	55	**10**	36	60
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Ind -----→ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ 8
Num -----→ 10 S -----→ 0

Cuando el contenido de la variable *Ind* sea 8 entonces la decisión *Si V (Ind) = Num*
Se convierte en *Si V (8) = Num*
y haciendo uso de sus respectivos contenidos la misma decisión representa

Si 10 = 10

Como la respuesta es Verdadero entonces se ejecutan las órdenes que están dentro de la parte Verdadera de la decisión o sea las siguientes dos instrucciones:

Escriba " El número ", Num, " si está en los 10 números digitados"
S = 1

Por lo tanto se coloca el título en pantalla y se cambia el valor de S a 1

P A N T A L L A

.
Ahora digite un
número
10
El número 10 sí está
en los 10 números
digitados

M E M O R I A

V	18	45	9	90	65	2	55	**10**	36	60
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Ind -----→ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ 8
Num -----→ 10 S -----→ ~~0~~ 1

Encontramos de nuevo el *Fin_Si* y el *Fin_Para* correspondiente con lo cual volvemos de todas formas a incrementar en 1 el contenido de la variable *Ind* y volvemos a hacer la pregunta.

P A N T A L L A

.
Ahora digite un
número
10
El número 10 sí está
en los 10 números
digitados

M E M O R I A

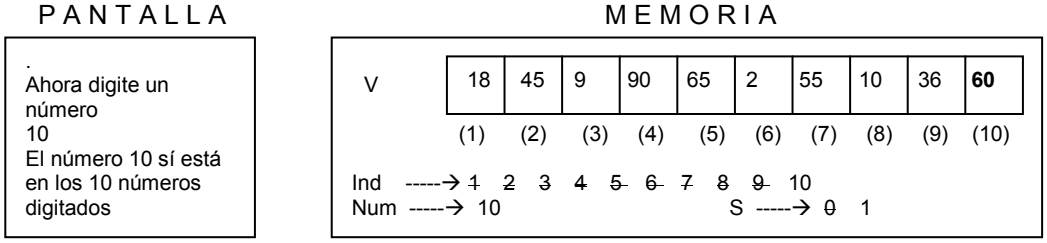
V	18	45	9	90	65	2	55	10	**36**	60
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Ind -----→ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ 9
Num -----→ 10 S -----→ ~~0~~ 1

Cuando el contenido de *Ind* sea igual a 9 entonces la decisión *Si V (Ind) = Num* se convierte en *Si V (9) = Num* y como V (9) es igual a 36 y Num es igual a 10 entonces la decisión finalmente es

Si 36 = 10

Como la respuesta es Falso entonces volvemos a incrementar (por última vez) el contenido de la variable Ind.



Cuando el contenido de *Ind* sea igual a 10 entonces la decisión *Si V (Ind) = Num* se convierte en *Si V (10) = Num* y como V (10) es igual a 36 y Num es igual a 10 entonces la decisión finalmente es

Si 60 = 10

Como la respuesta es Falso y ya llegamos al tope del ciclo entonces seguimos con la instrucción que está después del *Fin_Para* correspondiente

```
Si S = 0
        Escriba "El número ", Num, " no está en los 10 números digitados"
Fin_Si
```

Podemos ver que el contenido actual de la variable S es 1 por lo tanto la pregunta Si S = 0 se Falsa y por lo tanto pasamos a la instrucción que está después del *Fin_Si* y es el *Fin* del algoritmo.

Fin

Puede usted notar que el algoritmo falla en el hecho de que a pesar que se encuentre el valor buscado el sigue buscando sabiendo que el objetivo era sencillamente decir si el datos buscado esta o no. Para ello le sugiero que le haga una prueba de escritorio a la siguiente versión solución de este mismo problema

```
Programa Búsqueda
Variables
        Entero :V ( 10 ),                   // Almacenará los 10 números que el
                                                    // usuario va a digitar
                        Ind,                        // Esta es la variable que nos va a
                                                    // servir como subíndice del vector
                        Num,                        // Almacenará el número que se va a
                                                    // buscar
                        S                           // El contenido de esta variable nos va
                                                    // a indicar si el número se encontró o
                                                    // no.
Inicio
        Escriba "Digite 10 números enteros"         // Solicita los 10 números enteros
```

```
        Para Ind = 1 hasta 10 Paso 1                    // Y los lee almacenando cada uno en
                                                        // una posición diferente del vector
                Lea V ( Ind )
        Fin_Para

        Escriba "Ahora digite un número"                // Solicita el número a buscar
        Lea Num                                         // y lo lee almacenándolo en la variable
                                                        // Num

        S = 0                                           // Inicializa la variable "misteriosa" en 0
        Ind = 1

        Mientras Ind <= 10 Y S = 0
                Si V ( Ind ) = Num
                        Escriba " El número ", Num, " si está en los 10 números leídos "
                        S = 1
                Fin_Si
        Fin_Mientras

        Si S = 0
                Escriba "El número buscado no está "
        Fin_Si
Fin
```

La parte resaltada busca condicionar el ciclo a que la variable Ind no haya llegado hasta 10 y al mismo tiempo a que no se haya encontrado el dato buscado pues esa es la única forma de que S mantenga el valor 0 pues apenas se encuentre el contenido de la variable S cambiará a 1. Usted tal vez se preguntará Y qué pasa si se quisiera implementar el mismo control pero esta vez utilizando el ciclo Para...? Entonces basados en el ciclo Para que era

```
        Para Ind = 1 hasta 10 Paso 1
                Si V ( Ind ) = Num
                        Escriba " El número ", Num, " si está en los 10 números digitados"
                        S = 1
                Fin_Si
        Fin_Para
```

Con el control que se propone quedaría

```
        Para Ind = 1 hasta 10 Paso 1
                Si V ( Ind ) = Num
                        Escriba " El número ", Num, " si está en los 10 números digitados"
                        S = 1
                        Ind = 11
                Fin_Si
        Fin_Para
```

Siendo su único cambio la línea u orden que se encuentra en negrilla. Sugiero que le realice una buena prueba de escritorio para que vea la diferencia en la ejecución de este algoritmo incluyendo el control que se le ha incorporado.

Ejemplo Con Vectores No. 3

Leer 10 números enteros. Luego leer 1 número entero y determinar cuántas veces está entre los 10 primeros, además decir en qué posiciones está.

Clarificación del objetivo

Como usted puede notar el algoritmo anterior se podrá tomar como base para desarrollar otro algoritmo que nos permita lograr este objetivo. En esencia lo que se busca es leer 10 números enteros, que obviamente serán almacenados en un vector, y luego leer otro número entero con el objetivo de buscarlo entre los 10 números leídos y determinar no solo la cantidad de veces que está repetido sino también en qué posiciones se encuentra. Esto quiere decir que si el vector se "cargara" con los siguientes números

Vector V	15	20	16	24	23	16	19	16	25	82
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

y el número leído fuera 16 entonces en pantalla deberá aparecer

```
El número 16 se encuentra en las siguientes posiciones
3
6
8
En total está 3 veces
```

Algoritmo.- Es obvio pensar en la gran similitud que tendrá este algoritmo con el anterior. Solo por ejercicio vamos a desarrollarlo usando solamente ciclos *Mientras*.

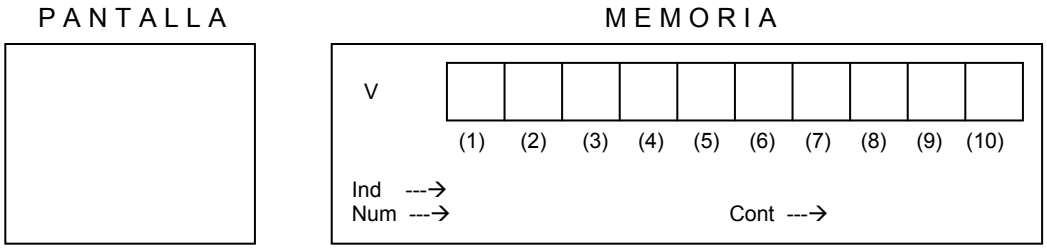
```
Programa Busqueda_Detallada
Variables
    Entero :    V ( 10 ),       // Almacenará los 10 números leídos
                Ind,            // Servirá como Indice de los ciclos para
                                // manejar las diferentes posiciones del vector
```

```
                    Num,                    // Almacenará el número a buscar
                    Cont                    // Almacenará la cantidad de veces que esté el
                                            // número en el vector y servirá para determinar // si
                                            el número estaba o no
Inicio
        Escriba " Digite 10 números enteros "   // Solicita los 10 números

        Ind = 1                             // Inicializa la variable Ind en 1 para hacer
                                            // referencia a la primer posición
        Mientras Ind <= 10                  // Mientras no se haya llegado a la última
                                            // posición del vector
                Lea V ( Ind )               // Lea un dato entero y guárdelo en la posición
                                            // Ind del Vector V
                Ind = Ind + 1               // Pase a la siguiente posición del vector
        Fin_Mientras                        // Fin del Ciclo

        Escriba "Digite un número entero"   // Solicita el número a  buscar
        Lea Num                             // Lo lee y lo almacena en la variable Num

        Cont = 0                            // Inicializa el contador en ceros
        Ind = 1                             // Inicializa la variable indice en 1 para hacer
                                            // referencia a la primera posición

        Mientras Ind <= 10                  // Mientras no se haya llegado a la última
                                            // posición en el vector
                Si Num = V ( Ind )          // Si el número buscado es igual al número
                                            // almacenado en la posición Ind del vector V
                                            // Escriba en donde lo encontró
                        Escriba " El número ", Num, " está en la posición ", Ind
                        Cont = Cont + 1// Incremente el contador pues lo ha
                                            // encontrado una vez ( mas )
                Fin_Si                      // Fin de la decisión

                Ind = Ind + 1               // Pase a la siguiente posición en el vector
        Fin_Mientras                        // Fin del Ciclo

        Si Cont = 0                         // Si el contador es cero o sea si en ningún
                                            // momento lo encontró entonces avise
                Escriba " El número no se encuentra "
        Sino                                // Si el número sí estaba entonces muestre en
                                            // pantalla cuántas veces estaba
                Escriba " En total está ", Cont, " veces "

        Fin_Si

Fin                                         // Fin del algoritmo
```

Prueba de Escritorio

De nuevo vamos a desarrollar una prueba de escritorio paso a paso apoyándonos en el soporte gráfico para conceptualizar mejor el flujo del algoritmo. Primero que nada declaramos en memoria las variables que se van a necesitar

Programa Busqueda_Detallada
Variables
Entero : V (10), Ind, Num, Cont

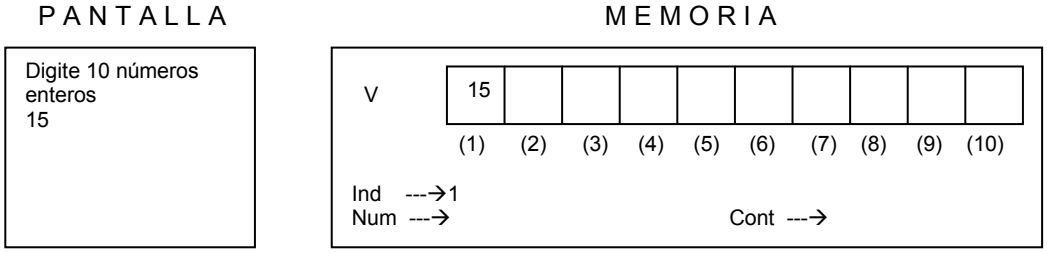


Seguidamente se inicia el algoritmo escribiendo en pantalla un aviso solicitando que se digiten 10 números. Luego se inicia la variable Ind en 1 y mientras esta variable no haya llegado a 10 (o sea mientras o se haya llegado a la última posición del vector) se va a leer un dato y se almacenará en el vector V en la posición Ind y luego se va a incrementar Ind en 1.

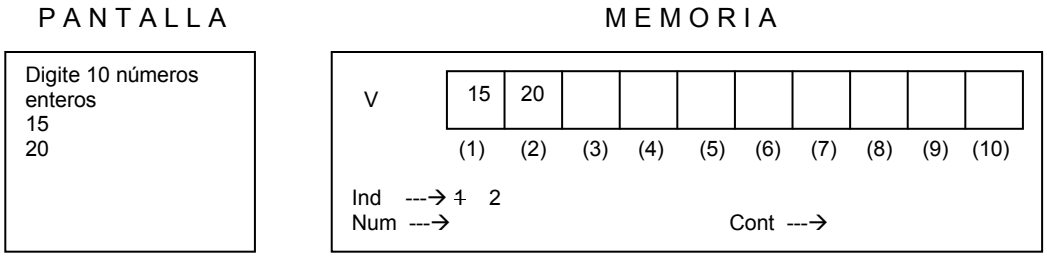
Inicio
Escriba " Digite 10 números enteros "

Ind = 1

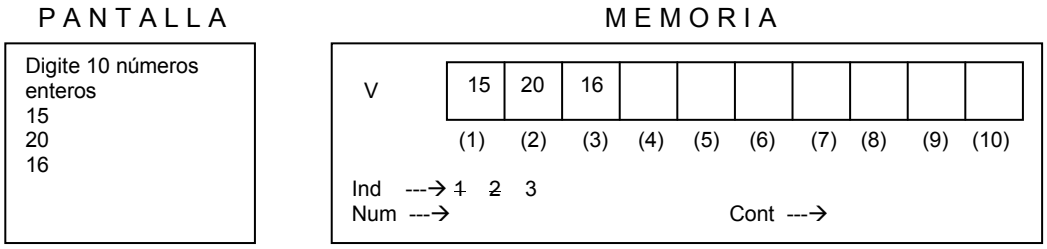
Mientras Ind <= 10
Lea V (Ind)
Ind = Ind + 1
Fin_Mientras



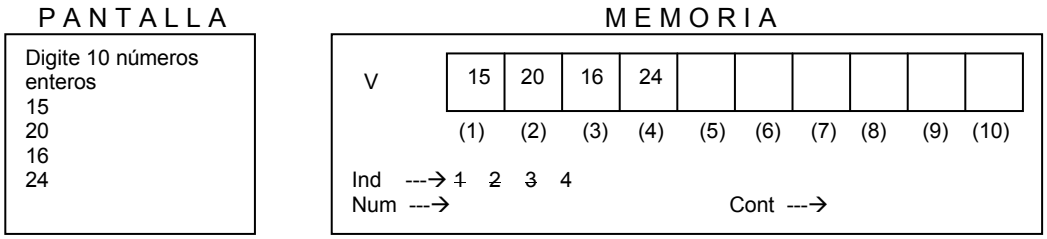
Cuando Ind valga 1 entonces el dato se almacenará en el vector V en la posición 1. Luego de esto se incrementará en 1 el contenido de la variable Ind y se verifica si todavía *Ind es menor o igual que 10*. Como es Verdadero entonces volvemos a leer otro dato.



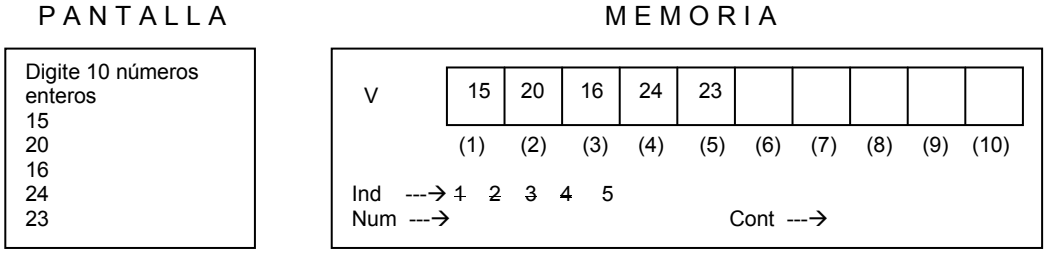
Cuando Ind valga 2 entonces el dato se almacenará en el vector V en la posición 2. Luego de esto se incrementará en 1 el contenido de la variable Ind y se verifica si todavía *Ind es menor o igual que 10*. Como es Verdadero entonces volvemos a leer otro dato.



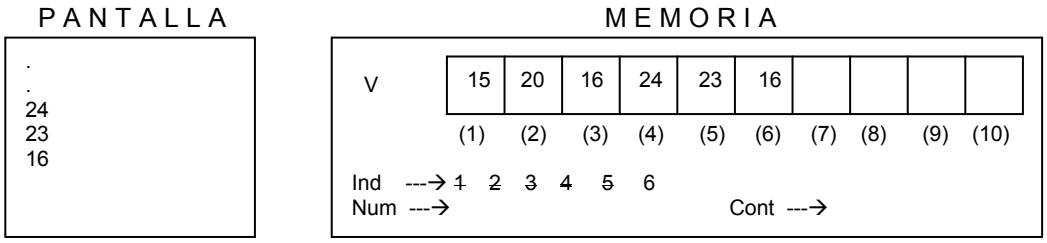
Cuando Ind valga 3 entonces el dato se almacenará en el vector V en la posición 3. Luego de esto se incrementará en 1 el contenido de la variable Ind y se verifica si todavía *Ind es menor o igual que 10*. Como es Verdadero entonces volvemos a leer otro dato.



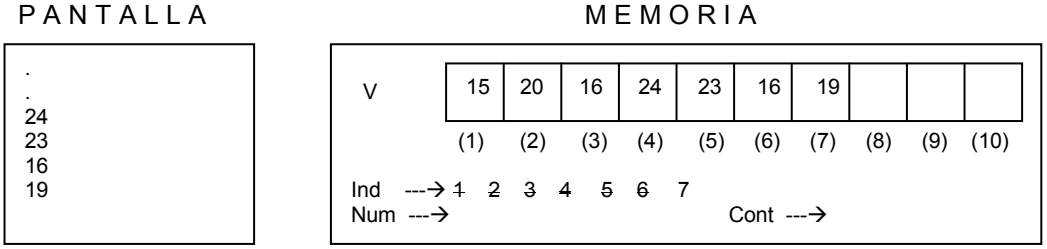
Cuando Ind valga 4 entonces el dato se almacenará en el vector V en la posición 4. Luego de esto se incrementará en 1 el contenido de la variable Ind y se verifica si todavía *Ind es menor o igual que 10*. Como es Verdadero entonces volvemos a leer otro dato.



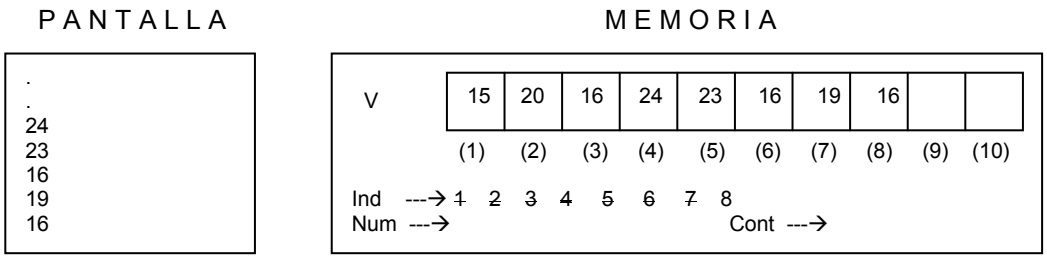
Cuando Ind valga 5 entonces el dato se almacenará en el vector V en la posición 5. Luego de esto se incrementará en 1 el contenido de la variable Ind y se verifica si todavía *Ind es menor o igual que 10*. Como es Verdadero entonces volvemos a leer otro dato.



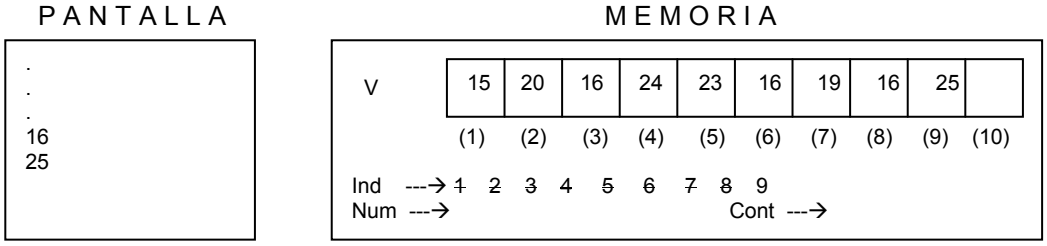
Cuando Ind valga 6 entonces el dato se almacenará en el vector V en la posición 6. Luego de esto se incrementará en 1 el contenido de la variable Ind y se verifica si todavía *Ind es menor o igual que 10*. Como es Verdadero entonces volvemos a leer otro dato.



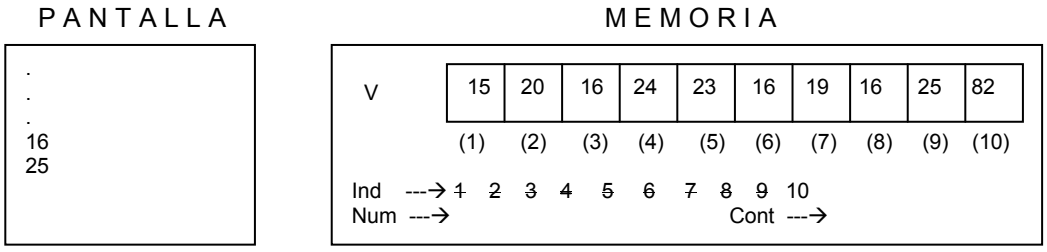
Cuando Ind valga 7 entonces el dato se almacenará en el vector V en la posición 7. Luego de esto se incrementará en 1 el contenido de la variable Ind y se verifica si todavía *Ind es menor o igual que 10*. Como es Verdadero entonces volvemos a leer otro dato.



Cuando Ind valga 8 entonces el dato se almacenará en el vector V en la posición 8. Luego de esto se incrementará en 1 el contenido de la variable Ind y se verifica si todavía *Ind es menor o igual que 10*. Como es Verdadero entonces volvemos a leer otro dato.



Cuando Ind valga 9 entonces el dato se almacenará en el vector V en la posición 9. Luego de esto se incrementará en 1 el contenido de la variable Ind y se verifica si todavía *Ind es menor o igual que 10*. Como es Verdadero entonces volvemos a leer otro dato.



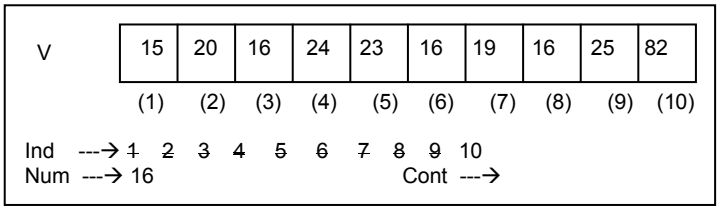
Y cuando Ind valga 10 entonces el dato se almacenará en el vector V en la posición 10. Como este valor era el tope del ciclo entonces éste se finaliza y se continúa con la instrucción que se encuentra después de su respectivo Fin_Para. Continuando con la instrucción correspondiente aparecerá en pantalla un título solicitando un número y seguidamente se deberá leer un número entero que ha de quedar almacenado en la variable Num. Vamos pues a asumir que el número leído para ser buscado es el número 16.

Escriba "Digite un número entero"
Lea Num

P A N T A L L A

```
.
.
25
Digite un número
entero
16
```

M E M O R I A



Luego de leído el número a buscar, iniciamos la variable Cont con el valor 0 dado que esta es la variable que nos va a permitir contar cuántas veces está el número buscado entre los 10 números leídos. Igualmente se inicializa (de nuevo) la variable Ind en 1 para hacer referencia a la primera posición del Vector y con ella misma controlar el recorrido a lo largo de las diferentes posiciones del mismo. Seguidamente se inicia un ciclo *Mientras* en donde su condición establece *Mientras no se haya llegado a la última posición del Vector* (o como quien dice *Mientras Ind <= 0*). En este ciclo se va a evaluar si el contenido del Vector V en la posición Ind es igual al número que estamos buscando. De ser así entonces se escribirá en pantalla que el número que buscamos está en la posición que esté almacenada en la variable Ind y a continuación se incrementará en 1 el contenido de la variable Cont que es la que va a almacenar la cantidad de veces que se encuentre el número en el vector.

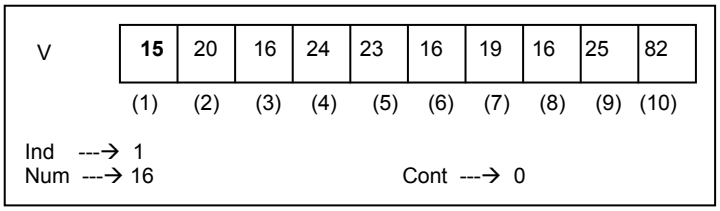
De otra parte sea Verdadera o Falsa la pregunta Si Num = V (Ind) se incrementará de todas maneras el contenido de la variable Ind para pasar a evaluar el valor almacenado en la siguiente "casilla" del vector. Así se seguirá progresivamente hasta llegar a la última posición tal como lo indica la condición del ciclo.

```
Cont = 0
Ind = 1

Mientras Ind <= 10
        Si Num = V ( Ind )
                Escriba " El número ", Num, " está en la posición ", Ind
                Cont = Cont + 1
        Fin_Si
        Ind = Ind + 1
Fin_Mientras
```

P A N T A L L A

```
.
.
25
Digite un número
entero
16
```

M E M O R I A



Cuando Ind valga 1 la decisión *Si Num = V (Ind) se convierte en Si Num = V (1)* y como el contenido del vector V en la posición 1 es 15 entonces la decisión se convierte en *Si 16 = 15* lo cual es Falso. Entonces se incrementa en 1 el contenido de la variable Ind y se vuelve a evaluar la condición del ciclo pues se ha encontrado el correspondiente Fin_Mientras.

P A N T A L L A

```
.
.
25
Digite un número
entero
16
```

M E M O R I A

V	15	**20**	16	24	23	16	19	16	25	82
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Ind ---→ ~~1~~ 2
Num ---→ 16 Cont ---→ 0

Cuando Ind valga 2 la decisión *Si Num = V (Ind) se convierte en Si Num = V (2)* y como el contenido del vector V en la posición 2 es 20 entonces la decisión se convierte en *Si 16 = 20* lo cual es Falso. Entonces se incrementa en 1 el contenido de la variable Ind y se vuelve a evaluar la condición del ciclo pues se ha encontrado el correspondiente Fin_Mientras.

P A N T A L L A

```
.
.
25
Digite un número
entero
16
```

M E M O R I A

V	15	20	**16**	24	23	16	19	16	25	82
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Ind ---→ ~~1~~ ~~2~~ 3
Num ---→ 16 Cont ---→ 0

Cuando Ind valga 3 la decisión *Si Num = V (Ind) se convierte en Si Num = V (3)* y como el contenido del vector V en la posición 3 es 16 entonces la decisión se convierte en *Si 16 = 16* lo cual es Verdadero. Por lo tanto ejecutamos las órdenes

Escriba " El número ", Num, " está en la posición ", Ind
Cont = Cont + 1

O sea que en pantalla aparece el aviso correspondiente diciendo que el número buscado está en la posición que es igual al contenido de la variable Ind (que en este momento vale 3) y se incrementa en 1 el contenido de la variable Cont. Posteriormente se incrementará en 1 el contenido de la misma variable Ind.

P A N T A L L A

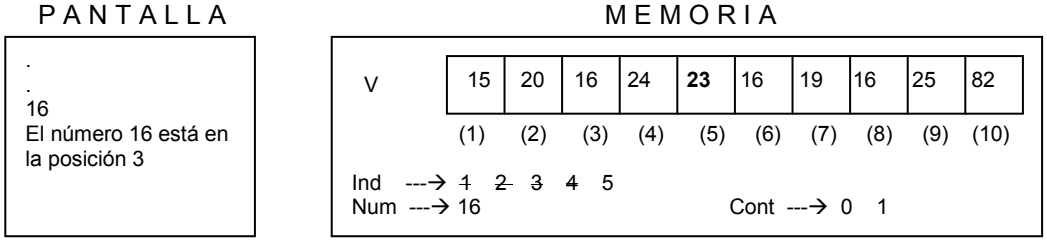
```
.
.
16
El número 16 está en
la posición 3
```

M E M O R I A

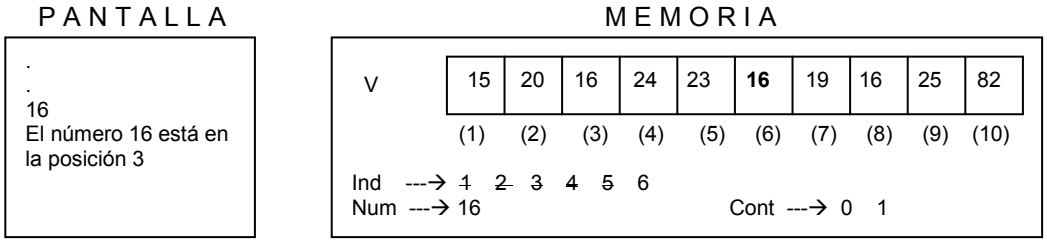
V	15	20	16	**24**	23	16	19	16	25	82
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

Ind ---→ ~~1~~ ~~2~~ ~~3~~ 4
Num ---→ 16 Cont ---→ 0 1

Cuando Ind valga 4 la decisión *Si Num = V (Ind) se convierte en Si Num = V (4)* y como el contenido del vector V en la posición 4 es 24 entonces la decisión se convierte en *Si 16 = 24* lo cual es Falso. Entonces se incrementa en 1 el contenido de la variable Ind y se vuelve a evaluar la condición del ciclo pues se ha encontrado el correspondiente Fin_Mientras.



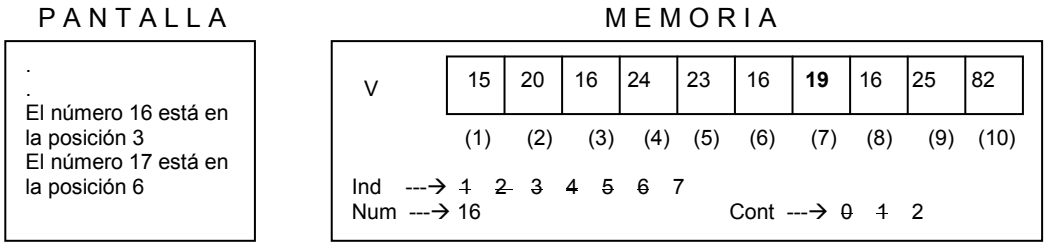
Cuando Ind valga 5 la decisión *Si Num = V (Ind) se convierte en Si Num = V (5)* y como el contenido del vector V en la posición 5 es 23 entonces la decisión se convierte en *Si 16 = 23* lo cual es Falso. Entonces se incrementa en 1 el contenido de la variable Ind y se vuelve a evaluar la condición del ciclo pues se ha encontrado el correspondiente Fin_Mientras.



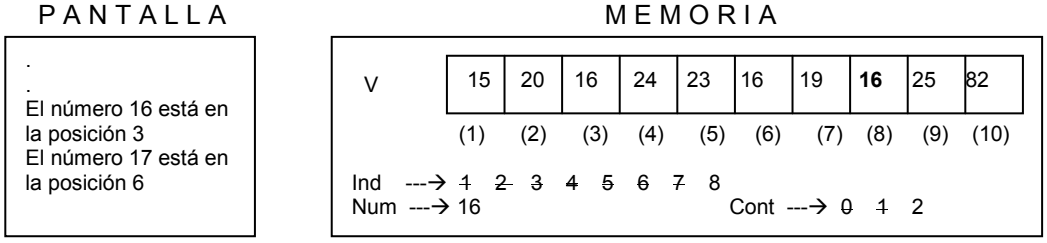
Cuando Ind valga 6 la decisión *Si Num = V (Ind) se convierte en Si Num = V (6)* y como el contenido del vector V en la posición 6 es 16 entonces la decisión se convierte en *Si 16 = 16* lo cual es Verdadero. Por lo tanto se ejecutan las ordenes

Escriba " El número ", Num, " está en la posición ", Ind
Cont = Cont + 1

Es decir se escribe en pantalla que el valor almacenado en la variable *Num* (que es el dato que estamos buscando) está en la posición 6 (que es igual al contenido de la variable *Ind*) y luego se incrementará el contenido de la variable *Cont* (que es la que va a almacenar la cantidad de veces que se encuentra el dato buscado). Después de esto se incrementará en 1 el contenido de la variable *Ind* y se volverá a evaluar la condición del ciclo para repetir el mismo proceso pero con el dato que se encuentra enseguida de la posición actual.



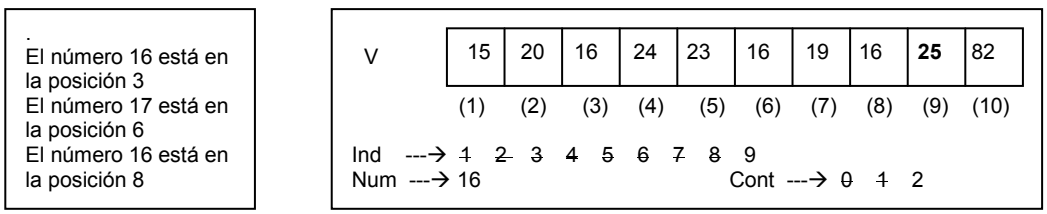
Cuando *Ind* valga 7 la decisión *Si Num = V (Ind) se convierte en Si Num = V (7)* y como el contenido del vector V en la posición 7 es 19 entonces la decisión se convierte en *Si 16 = 19* lo cual es Falso. Entonces se incrementa en 1 el contenido de la variable *Ind* y se vuelve a evaluar la condición del ciclo pues se ha encontrado el correspondiente *Fin_Mientras*.



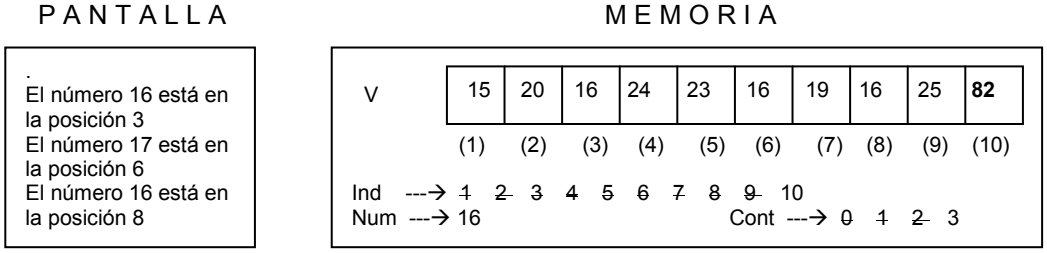
Cuando *Ind* valga 8 la decisión *Si Num = V (Ind) se convierte en Si Num = V (8)* y como el contenido del vector V en la posición 8 es 16 entonces la decisión se convierte en *Si 16 = 16* lo cual es Verdadero. Por lo tanto se ejecutan las ordenes

Escriba " El número ", Num, " está en la posición ", Ind
Cont = Cont + 1

Es decir se escribe en pantalla que el valor almacenado en la variable *Num* (que es el dato que estamos buscando) está en la posición 8 (que es igual al contenido de la variable *Ind*) y luego se incrementará el contenido de la variable *Cont* (que es la que va a almacenar la cantidad de veces que se encuentra el dato buscado). Después de esto se incrementará en 1 el contenido de la variable *Ind* y se volverá a evaluar la condición del ciclo para repetir el mismo proceso pero con el dato que se encuentra enseguida de la posición actual.



Cuando *Ind* valga 9 la decisión *Si Num = V (Ind) se convierte en Si Num = V (9)* y como el contenido del vector V en la posición 9 es 25 entonces la decisión se convierte en *Si 16 = 25* lo cual es Falso. Entonces se incrementa en 1 el contenido de la variable Ind y se vuelve a evaluar la condición del ciclo pues se ha encontrado el correspondiente *Fin_Mientras*.



Cuando *Ind* valga 10 la decisión *Si Num = V (Ind) se convierte en Si Num = V (10)* y como el contenido del vector V en la posición 10 es 82 entonces la decisión se convierte en *Si 16 = 82* lo cual es Falso. En este punto finaliza el ciclo dado que se alcanza el ultimo valor planteado al inicio el mismo.

Luego de que se ha realizado este ciclo se pasa a la decisión

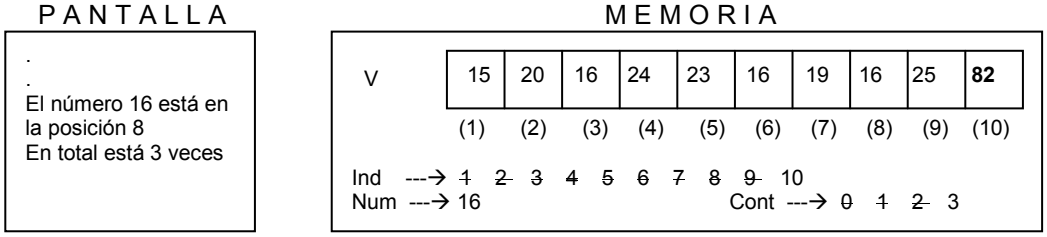
Si Cont = 0
Escriba " El número no se encuentra "
Sino
Escriba " En total está ", Cont, " veces "

Fin_Si

Con la cual podemos saber si el número buscado fue encontrado o no. Este resultado depende del contenido de la variable *Cont* pues si esta variable contiene un cero es porque el número buscado no estuvo presenta ni siquiera una sola vez. Si el contenido de esta variable es diferente de cero entonces quiere decir que el algoritmo encontró el número buscado al menos 1 vez. En este caso como el contenido de la variable Cont es 3 entonces se ejecuta

Sino
Escriba " En total está ", Cont, " veces "

Con lo cual aparecería en pantalla



Lo cual, por las características específicas de la prueba de escritorio, es verdad. Luego de esto encontramos el fin del algoritmo

Fin

Y con ello termina el mismo. Es momento entonces de evaluar los resultados y compararlos con el objetivo propuesto. Vemos que si entre los dígitos

15	20	16	24	23	16	19	16	25	82

Queremos buscar el número 16, encontramos que éste está 3 veces. También vemos que la primera ocurrencia de este número se da en la posición 3, la segunda ocurrencia se da en la posición 6 y la tercera ocurrencia se da en la posición 8. Como esto precisamente coincide con los resultados entregados por el algoritmo entonces podemos garantizar que este algoritmo está bien.

Ejemplo Con Vectores No. 4

"Cargar" dos vectores cada uno con 5 datos enteros y determinar si los datos almacenados en ambos vectores son exactamente los mismos tanto en contenido como en posición.

Clarificación del objetivo

Se trata entonces de comparar dos conjuntos de datos y determinar si ambos conjuntos son exactamente iguales. Para ello lo que vamos a hacer será leer 5 datos enteros y los almacenaremos en un vector, luego leeremos otros 5 datos enteros y los almacenaremos en otro vector. Luego implementaremos un proceso de comparación que nos permite determinar si ambos vectores son exactamente iguales. Tenga en cuenta que la comparación entre vectores no se puede hacer como se hace con las variables. Si usted tiene una variable X con un contenido igual a 8

X = 8

y tiene una variable y con un contenido igual a 8

Y = 8

entonces en cualquier momento usted puede preguntar

Si X = Y

Y el computador entenderá que deberá comparar los contenidos de dichas variables dado que dichos contenidos son únicos. Pero si se tiene el vector V1 con los siguientes contenidos

Vector V1	5	8	9	12	18
	(1)	(2)	(3)	(4)	(5)

Y tenemos el vector V2 con los siguientes contenidos

Vector V2	6	9	11	5	8
	(1)	(2)	(3)	(4)	(5)

No podremos realizar la pregunta

Si V1 = V2

Dado que como cada uno es un vector y no contiene datos únicos entonces realmente no sabría el computador (cuando este algoritmo se convierta en un programa) qué es lo que realmente tiene que comparar.

Precisamente para implementar dicha comparación lo que vamos a hacer será comparar una a una, y análogamente, cada contenido de los vectores. Por lo dicho aquí tendremos que recorrer los vectores, posición a posición, valiéndonos de un ciclo que irá desde la primera hasta la última posición (en este caso de 1 hasta 5). Nuestra pregunta será

Si V1 (Ind) = V2 (Ind)

Siendo *Ind* la variable que vamos a utilizar para referenciar el subíndice de posición de cada uno de los datos. Antes de iniciar con este ciclo de comparación asignaremos a una variable el valor 0 y esta variable actuará como un interruptor, es decir, cambiará su valor solo si se encuentra que, en algún momento, la decisión planteada es Falsa. Al finalizar el ciclo veremos cuál fue el valor de la variable interruptor y con ello podremos determinar si los dos vectores eran iguales o no. Debemos tener en cuenta que el proceso de comparación de dos vectores es diferente al proceso de comparación de dos variables dado que los vectores son conjuntos de variables y las variables solas no.

Algoritmo

No olvide lo que a lo largo de este libro le he recordado, es posible que usted desarrolle un algoritmo que logre este mismo objetivo, si no es igual al que aquí se expone no se preocupe. Todo lo que tiene que hacer es verificar si cumple con el objetivo o no realizándole una buena prueba de escritorio. Si su solución cumple con el objetivo entonces estará bien y si la solución expuesta en este libro cumple con el objetivo entonces también estará bien. Siendo así ambas soluciones serán correctas.

```
Programa Compara_Vectores
Variables
        Entero :        V1 ( 5 ),           // Vector en donde se  almacenará el primer
                                            // conjunto de 5 datos enteros
                        V2 ( 5 ),           // Vector en donde se almacenará el segundo
                                            // conjunto de 5 datos enteros
                        Ind,                // Variable que servirá como subíndice
                        S                   // Variable que permitirá determinar si el
                                            // contenido de los dos vectores era
                                            // exactamente igual o no
Inicio
        Escriba " Digite el primer conjunto de 5 enteros "

        Para Ind = 1 hasta 5                // Se solicitan 5 números enteros y se leen
                Lea V1 ( Ind )              // almacenando cada uno en una posición
        Fin_Para                            // diferente del vector V1
```

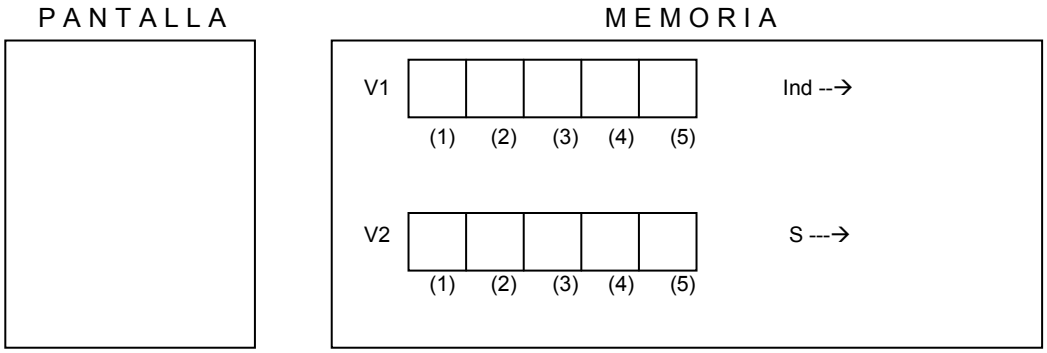
```
        Escriba " Digite el segundo conjunto de 5 enteros "

        Para Ind = 1 hasta 5                    // Se solicitan los otros 5 números enteros y
                Lea V2 ( Ind )                  // se leen almacenando cada uno en una
        Fin_Para                                // diferente del vector V2


        S = 0                                   // Se inicializa la variable S en 0
        Ind = 1                                 // Se inicializa la variable Ind en 1 para hacer
                                                // referencia al primer elemento de los vectores
                                                // e iniciar allí la comparación

        Mientras Ind <= 5 Y S = 0               // Mientras no se haya llegado el final del vector
                                                // (o sea a su última posición) y mientras la
                                                // variable S siga con el valor 0 (o sea mientras
                                                // los vectores sigan iguales en sus contenidos)
                Si V1 ( Ind ) != V2 ( Ind )     // Si el contenido del primer vector en una
                                                // posición determinada es igual al contenido
                                                // del segundo vector en la misma posición
                        S = 1                   // Cambie el contenido de la variable S a 1
                Fin_S                           // Fin de la decisión
                Ind = Ind + 1                   // Pase a la siguiente posición del vector
        Fin_Mientras                            // Fin del Ciclo

        Si S = 0                        // Si esta variable aún permanece con el valor 0
                                        // quiere decir que en ningún momento se le
                                        // asignó el valor 1 o sea que en ningún
                                        // momento el contenido de los vectores fue
                                        // diferente por lo tanto los vectores son
                                        // exactamente iguales
                Escriba " El contenido de los dos vectores es exactamente igual "
        Sino                            // Si la variable S vale 1 entonces quiere decir
                                        // que en algún momento el contenido de los
                                        // vectores fueron diferente por lo tanto no son
                                        // exactamente iguales
                Escriba " El contenido de los dos vectores es diferente "
        Fin_Si                          // Fin de la decisión

Fin                                     // Fin del ciclo
```

Prueba de Escritorio

Realice un seguimiento de esta prueba paso a paso tanto en lo que corresponde al manejo de la pantalla como lo que corresponde al manejo de las variables en la memoria. Lo primero que hacemos es declarar las variables correspondientes en memoria

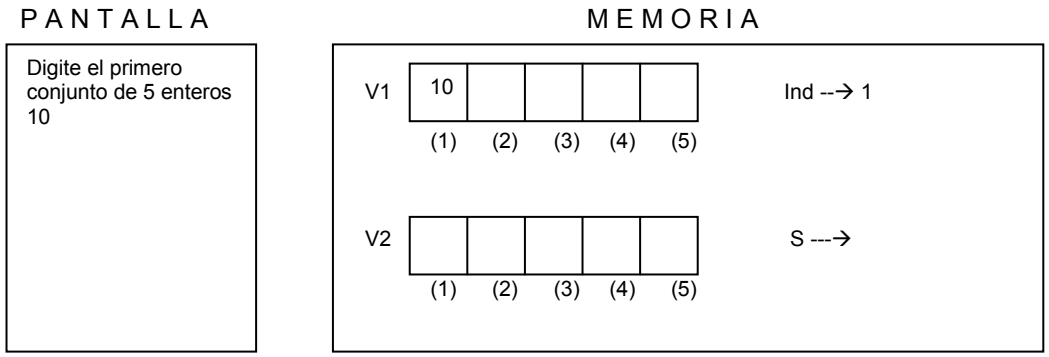
```
Programa Compara_Vectores
Variables
        Entero :        V1 ( 5 ),
```

V2 (5),
Ind,
S

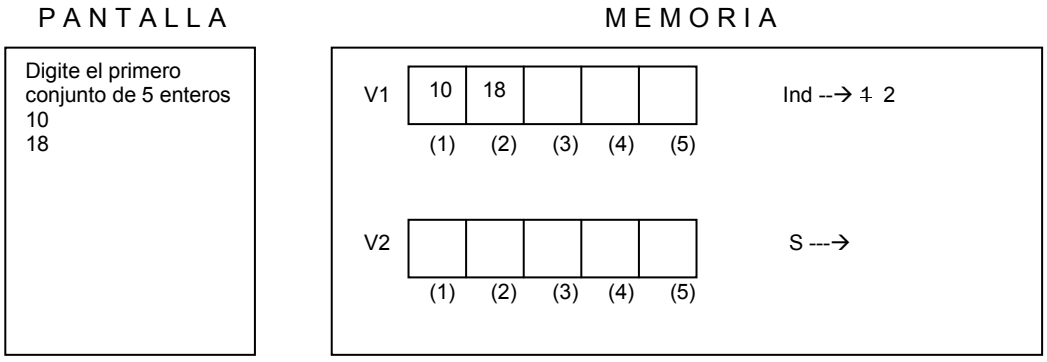


A continuación solicitamos 5 datos enteros y los leemos valiéndonos de un ciclo que, utilizando la variable *Ind*, vaya desde 1 hasta 5 para referenciar cada una de las posiciones dentro del vector. Con este ciclo queda "cargado" el vector *V1*.

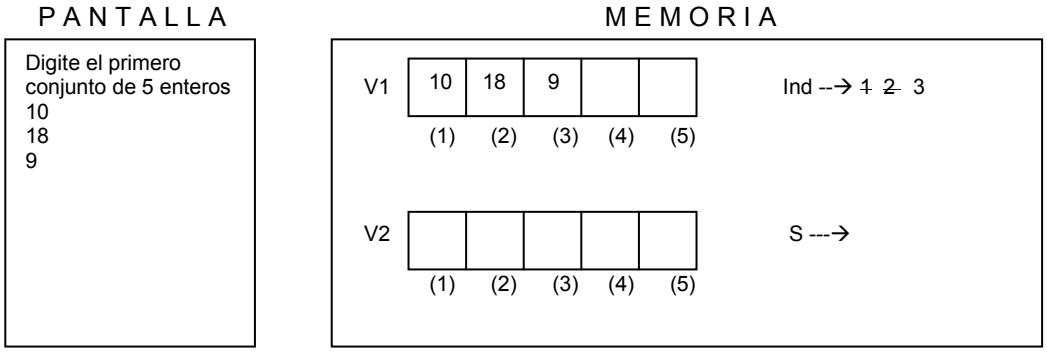
```
Inicio
        Escriba " Digite el primer conjunto de 5 enteros "

        Para Ind = 1 hasta 5
                Lea V1 ( Ind )
        Fin_Para
```



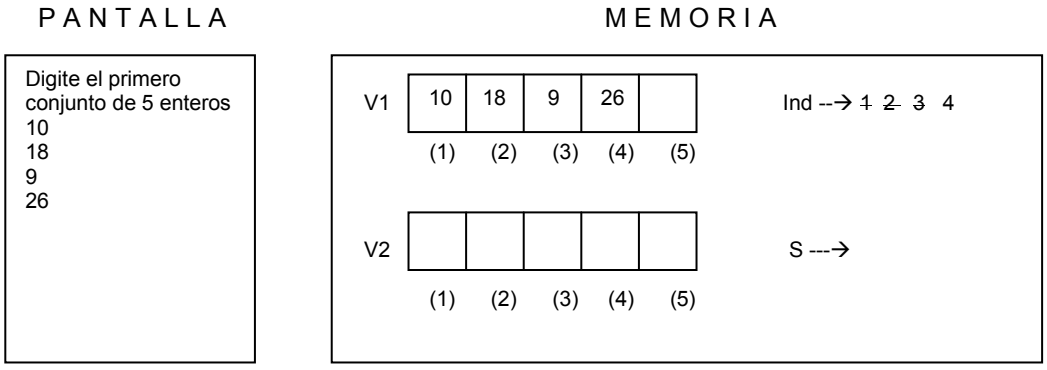
Cuando la variable *Ind* almacene el número 1, y asumiendo que el número leído sea el 10, dicho número quedará almacenado en el vector V1 en la posición 1. Se incrementa entonces el valor de la variable *Ind* en 1.



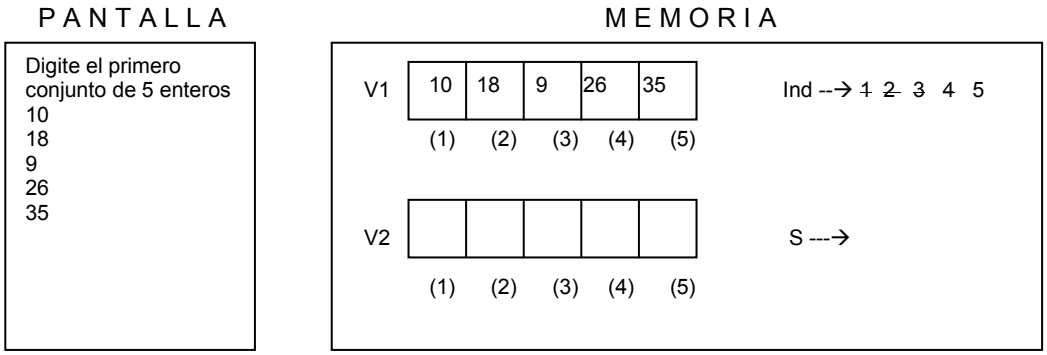
Cuando la variable *Ind* almacene el número 2, y asumiendo que el número leído sea el 18, dicho número quedará almacenado en el vector V1 en la posición 2. Se incrementa entonces el valor de la variable *Ind* en 1.



Cuando la variable *Ind* almacene el número 3, y asumiendo que el número leído sea el 9, dicho número quedará almacenado en el vector V1 en la posición 3. Se incrementa entonces el valor de la variable *Ind* en 1.



Cuando la variable *Ind* almacene el número 4, y asumiendo que el número leído sea el 9, dicho número quedará almacenado en el vector V1 en la posición 4. Se incrementa entonces el valor de la variable *Ind* en 1.

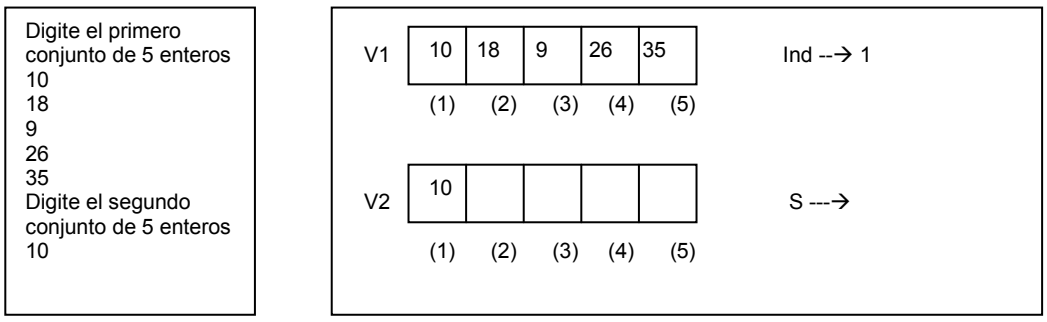


Y cuando la variable *Ind* almacene el número 5, y asumiendo que el número leído sea el 35, dicho número quedará almacenado en el vector V1 en la posición 5. Como este es era el tope hasta donde debía llegar la variable Ind entonces se finaliza este ciclo se continúa con la instrucción que se encuentra después del respectivo Fin_Para.

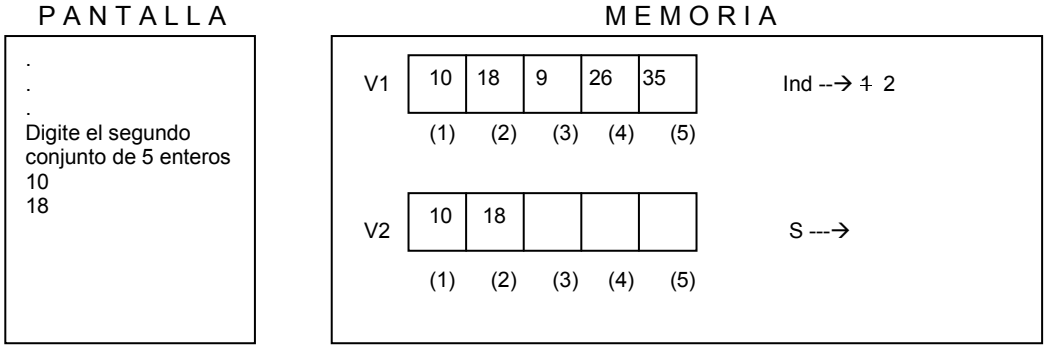
Terminada esta parte en donde se leen 5 datos enteros y se almacenan en el primer vector pasamos a la segunda parte en donde solicitamos otros 5 datos enteros pero esta vez los vamos a almacenar en el segundo vector (que hemos llamado V2). Volvemos a generar los números del 1 al 5 utilizando la variable Ind como índice y valiéndonos de un ciclo.

Escriba " Digite el segundo conjunto de 5 enteros "

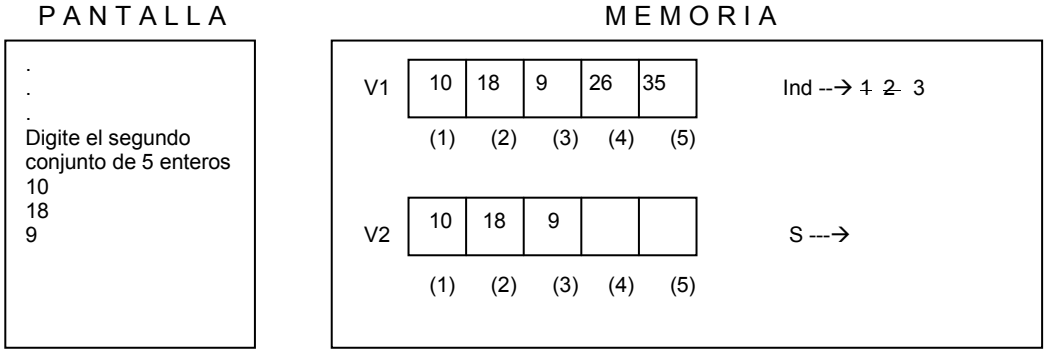
```
Para Ind = 1 hasta 5
        Lea V2 ( Ind )
Fin_Para
```



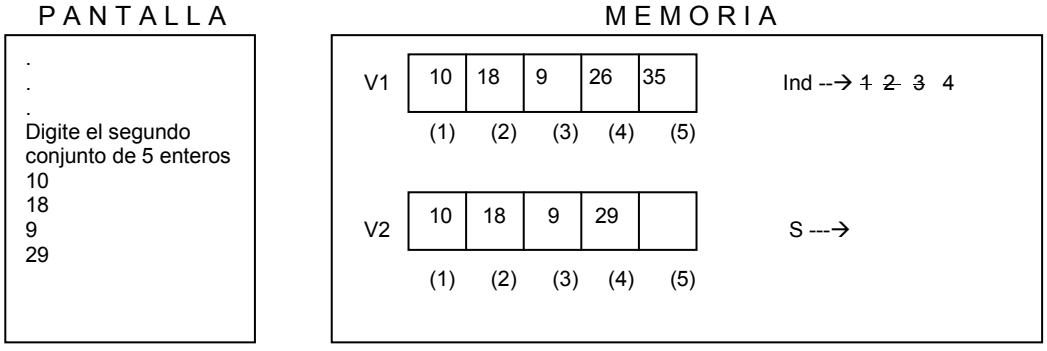
Cuando la variable *Ind* almacene el número 1, y asumiendo que el número leído sea el 10, dicho número quedará almacenado en el vector V2 en la posición 1. Se incrementa entonces el valor de la variable *Ind* en 1.



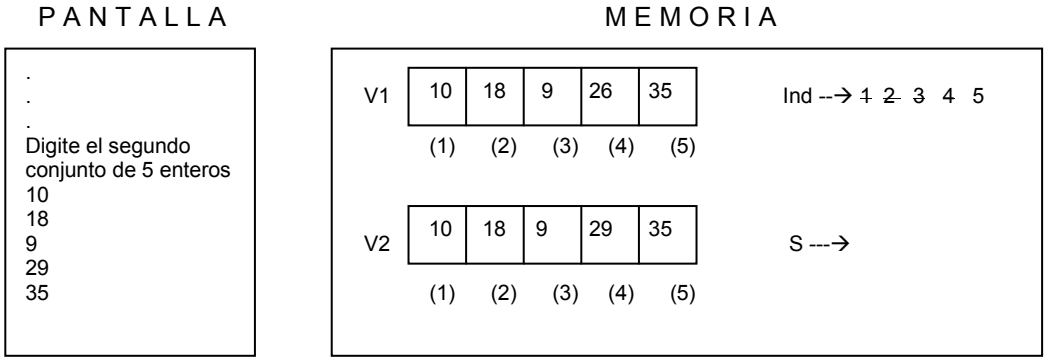
Cuando la variable *Ind* almacene el número 2, y asumiendo que el número leído sea el 18, dicho número quedará almacenado en el vector V2 en la posición 2. Se incrementa entonces el valor de la variable *Ind* en 1.



Cuando la variable *Ind* almacene el número 3 y asumiendo que el número leído sea el 9, dicho número quedará almacenado en el vector V2 en la posición 3. Se incrementa entonces el valor de la variable *Ind* en 1.



Cuando la variable *Ind* almacene el número 4 y asumiendo que el número leído sea el 29, dicho número quedará almacenado en el vector V2 en la posición 4. Se incrementa entonces el valor de la variable *Ind* en 1.



Y cuando la variable *Ind* almacene el número 5 y asumiendo que el número leído sea el 35, dicho número quedará almacenado en el vector V2 en la posición 5. Con esto terminará este segundo ciclo dado que el tope del mismo era 5 y ya se llegó a este valor.

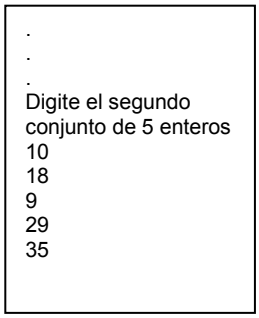
Con esto finalizamos la "carga" de los vectores con lo cual ya podemos entrar a determinar si son exactamente iguales o no. Es natural pensar en este momento que los dos vectores no son iguales y por ello pareciera ser inoficioso este algoritmo pero por ahora nos interesa cumplir con el objetivo, es decir, demostrar que a través de este algoritmo un computador puede determinar si un conjunto de datos (almacenado en un vector) es igual a otro conjunto de datos o no. Por tal motivo nuestro algoritmo continúa almacenando en las variables *S* e *Ind* los valores 0 y 1 respectivamente. Luego se plantea un ciclo que debe permanecer mientras el contenido de la variable Ind sea menor o igual que 5 (o sea mientras no se haya llegado a la posición del último dato en los vectores) y mientras la variable *S* siga almacenando el número 0.

Dentro de este ciclo se preguntará si el contenido del vector V1 en la posición *Ind* es diferente al valor almacenado en el Vector *V2* en la misma posición (para cualquier valor que almacene la variable *Ind*). En el instante en que esto sea Verdadero entonces se cambiará el dato almacenado en la variable *S*, que estaba en 0, por un 1 y con esto se "abortaría" el ciclo que se esta ejecutando dado que la condición dice *Mientras Ind <= 5* ***Y*** *S = 0*. Sea pues Verdadera o Falsa la decisión se le adicionará 1 al valor almacenado en la variable *Ind* para poder hacer referencia a la siguiente posición dentro de los vectores.

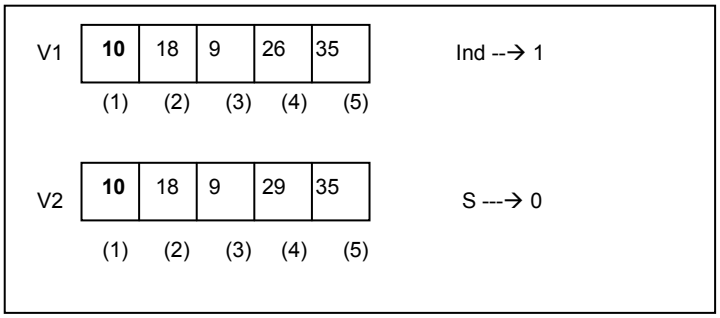
```
S = 0
Ind = 1

Mientras Ind <= 5 Y S = 0

        Si V1 ( Ind ) != V2 ( Ind )
                S = 1
        Fin_Si

        Ind = Ind + 1
```

Fin_Mientras

P A N T A L L A



M E M O R I A



Cuando la variable *Ind* valga 1 la decisión *Si V1 (Ind) != V2 (Ind)* se convierte en

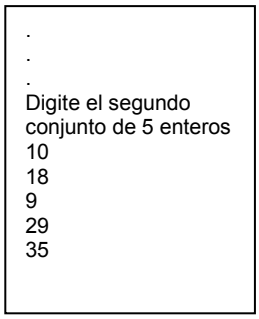
Si V1 (1) != V2 (1)

Es de anotar que el signo **!=** lo estamos utilizando en este libro para representar *Diferente de* tal como se usa en el Lenguaje C. El lector podrá utilizar cualquier otra equivalencia para este operador relacional (tales como < > ó sencillamente *diferente de).* Como el contenido del vector V1 en la posición 1 es 10 y el contenido del vector V2 en la posición 1 es 10 entonces la decisión se transforma, internamente, en

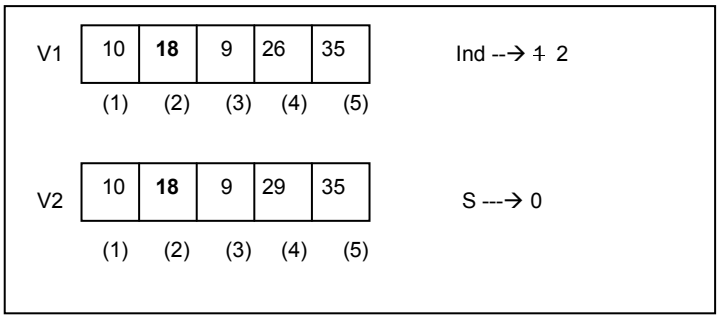
Si 10 != 10

Es decir, si 10 es diferente de 10. Como la respuesta a esta decisión es Falso entonces continuamos con la instrucción que se encuentra después del respectivo *Fin_Si* y que corresponde a incrementar en 1 el valor almacenado en la variable *Ind* quedando con el valor *2*. Como seguidamente encontramos el *Fin* del ciclo entonces volvemos a evaluar la condición de dicho ciclo para entrar de nuevo al cuerpo del mismo debido a que el contenido de la variable *Ind* es menor que 5 y también debido a que el contenido de la variable *S* sigue siendo 0.

P A N T A L L A



M E M O R I A



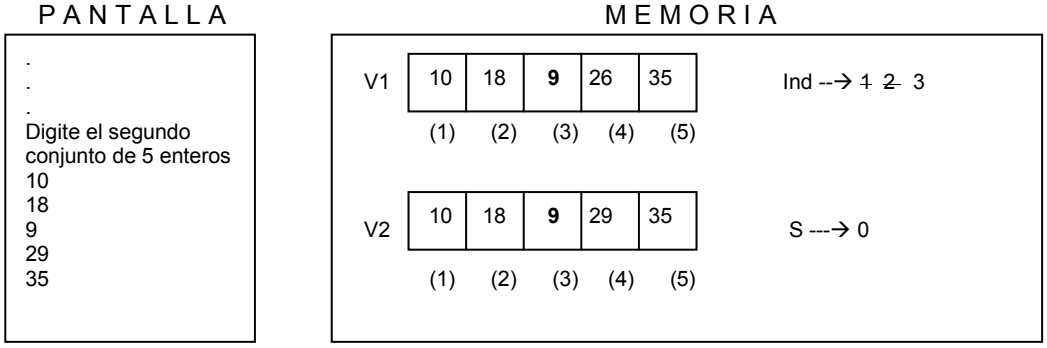
Cuando la variable *Ind* valga 2 la decisión *Si V1 (Ind) != V2 (Ind)* se convierte en

Si V1 (2) ! = V2 (2)

Como el contenido del vector V1 en la posición 2 es 18 y el contenido del vector V2 en la posición 2 es 18, la instrucción se convierte internamente en

Si 18 ! = 18

Como la respuesta a esta decisión es Falso entonces ejecutamos la orden que está después del Fin_Si o sea incrementamos en 1 el contenido de la variable Ind que ahora quedará con el valor 3.



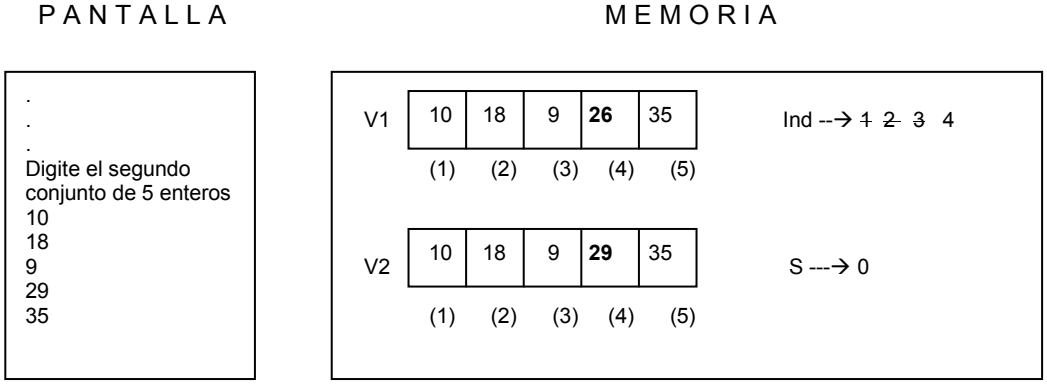
Cuando la variable *Ind* valga 3 la decisión *Si V1 (Ind) != V2 (Ind)* se convierte en

Si V1 (3) ! = V2 (3)

Como el contenido del vector V1 en la posición 3 es 9 y el contenido del vector V2 en la posición 3 es 9, la instrucción se convierte internamente en

Si 9 ! = 9

Como la respuesta a esta decisión es Falso entonces ejecutamos la orden que está después del Fin_Si o sea incrementamos en 1 el contenido de la variable *Ind* que ahora quedará con el valor 4.



Cuando la variable *Ind* valga 4 la decisión *Si V1 (Ind) != V2 (Ind)* se convierte en

Si V1 (4) ! = V2 (4)

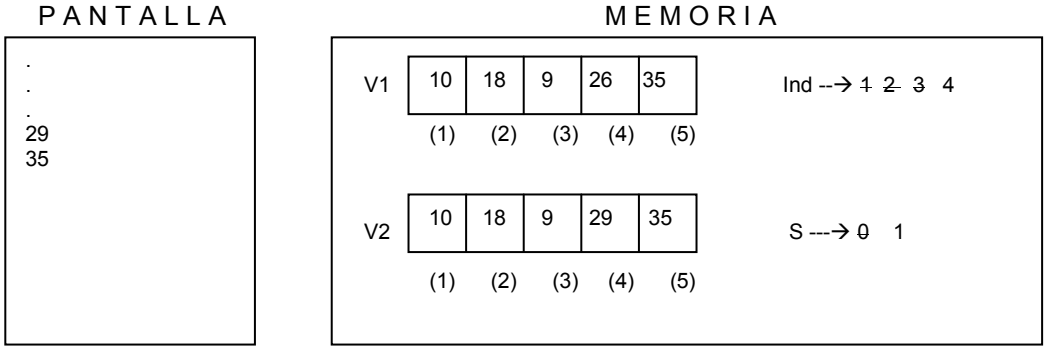
Como el contenido del vector V1 en la posición 4 es 26 y el contenido del vector V2 en la posición 4 es 29, la instrucción se convierte internamente en

Si 26 ! = 29

Como la respuesta a esta decisión es Verdadero entonces se ejecuta la orden

S = 1

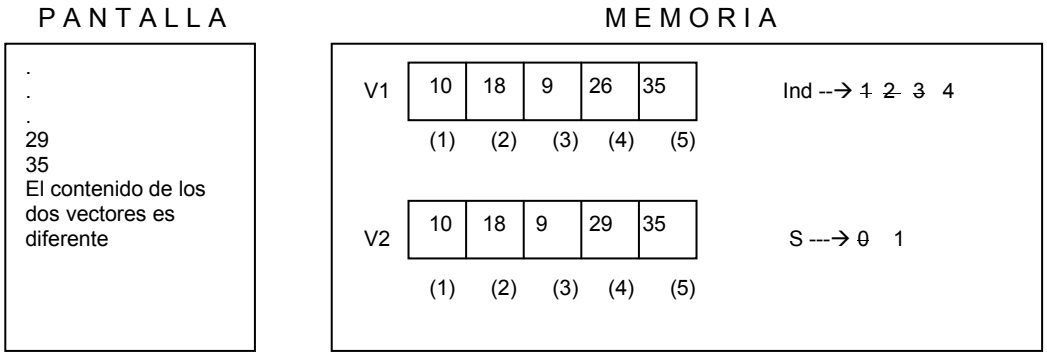
Con lo cual se modifica el contenido de la variable S. Seguidamente ejecutamos la orden que incrementa en 1 el contenido de la variable Ind. Volvemos, entonces, a evaluar la condición del ciclo y vemos que Ind todavía es menor que 5 pero esta vez S ya no es igual a 0 por lo tanto la primera parte de la condición es Verdadera pero la segunda parte de la condición es Falsa, como están unidas por un operador **Y** entonces basados en su tabla de verdad vemos que *Verdadero* ***Y*** *Falso* nos da como resultado total *Falso.* Por lo tanto esto indica que finalizamos la ejecución del cuerpo del ciclo y pasamos a la instrucción que está después del correspondiente Fin_Mientras.



Con las variables en memoria tal como están aquí se realiza la siguiente decisión

```
Si S = 0
        Escriba " El contenido de los dos vectores es exactamente igual "
Sino
        Escriba " El contenido de los dos vectores es diferente "
Fin_Si
```

Se pregunta si el contenido de la variable S es igual a 0. Como vemos dicho contenido es 1 por lo tanto la respuesta a dicha decisión es *Falso*. Se ejecuta la instrucción que está seguidamente al *Sino* de esta decisión. Por lo tanto saldrá en pantalla



Lo cual es Verdadero dado que realmente los dos vectores no son exactamente iguales. Después de esto llegamos al fin del algoritmo.

Fin

Es importante que note usted el papel que desempeña en este algoritmo la variable S que es la que finalmente nos permite determinar si los dos vectores son iguales (en contenido) o no. Como se puede notar el resultado del algoritmo es cierto y por ahora podemos decir que en el caso en el cual los vectores tengan contenidos diferentes este algoritmo lo determinará acertadamente. Ahora realícele una prueba de escritorio a este algoritmo asumiendo que el contenido de los dos vectores es exactamente igual para saber si el algoritmo en ese caso también lo determina. Si es así entonces podremos decir que el algoritmo aquí presentado está bien.

Ejercicios

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

1. Leer 10 enteros, almacenarlos en un vector y determinar en qué posición del vector está el mayor número leído.

2. Leer 10 enteros, almacenarlos en un vector y determinar en qué posición del vector está el mayor número par leído.

3. Leer 10 enteros, almacenarlos en un vector y determinar en qué posición del vector está el mayor número primo leído.

4. Cargar un vector de 10 posiciones con los 10 primeros elementos de la serie de Fibonacci y mostrarlo en pantalla.

5. Almacenar en un vector de 10 posiciones los 10 números primos comprendidos entre 100 y 300. Luego mostrarlos en pantalla.

6. Leer dos números enteros y almacenar en un vector los 10 primeros números primos comprendidos entre el menor y el mayor. Luego mostrarlos en pantalla.

7. Leer 10 números enteros, almacenarlos en un vector y determinar en qué posiciones se encuentra el número mayor.

8. Leer 10 números enteros, almacenarlos en un vector y determinar en qué posiciones se encuentran los números terminados en 4.

9. Leer 10 números enteros, almacenarlos en un vector y determinar cuántas veces está repetido el mayor.

10. Leer 10 números enteros, almacenarlos en un vector y determinar en qué posiciones se encuentran los números con mas de 3 dígitos.

11. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos números tienen, de los almacenados allí, tienen menos de 3 dígitos.

12. Leer 10 números enteros, almacenarlos en un vector y determinar a cuánto es igual el promedio entero de los datos del vector.

13. Leer 10 números enteros, almacenarlos en un vector y determinar si el promedio entero de estos datos está almacenado en el vector.

14. Leer 10 números enteros, almacenarlos en un vector y determinar cuántas veces se repite el promedio entero de los datos dentro del vector.

15. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos datos almacenados son múltiplos de 3.

16. Leer 10 números enteros, almacenarlos en un vector y determinar cuáles son los datos almacenados múltiplos de 3.

17. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos números negativos hay.

18. Leer 10 números enteros, almacenarlos en un vector y determinar en qué posiciones están los números positivos.

19. Leer 10 números enteros, almacenarlos en un vector y determinar cuál es el número menor.

20. Leer 10 números enteros, almacenarlos en un vector y determinar en qué posición está el menor número primo.

21. Leer 10 números enteros, almacenarlos en un vector y determinar en qué posición está el número cuya suma de dígitos sea la mayor.

22. Leer 10 números enteros, almacenarlos en un vector y determinar cuáles son los números múltiplos de 5 y en qué posiciones están.

23. Leer 10 números enteros, almacenarlos en un vector y determinar si existe al menos un número repetido.

24. Leer 10 números enteros, almacenarlos en un vector y determinar en qué posición está el número con mas dígitos.

25. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos de los números leídos son números primos terminados en 3.

26. Leer 10 números enteros, almacenarlos en un vector y calcularle el factorial a cada uno de los números leídos almacenándolos en otro vector.

27. Leer 10 números enteros, almacenarlos en un vector y determinar a cuánto es igual el promedio entero de los factoriales de cada uno de los números leídos.

28. Leer 10 números enteros, almacenarlos en un vector y mostrar en pantalla todos los enteros comprendidos entre 1 y cada uno de los números almacenados en el vector.

29. Leer 10 números enteros, almacenarlos en un vector y mostrar en pantalla todos los enteros comprendidos entre 1 y cada uno de los dígitos de cada uno de los números almacenados en el vector.

30. Leer 10 números enteros, almacenarlos en un vector. Luego leer un entero y determinar si este último entero se encuentra entre los 10 valores almacenados en el vector.

31. Leer 10 números enteros, almacenarlos en un vector. Luego leer un entero y determinar cuantos divisores exactos tiene este último número entre los valores almacenados en el vector.

32. Leer 10 números enteros, almacenarlos en un vector. Luego leer un entero y determinar cuántos números de los almacenados en el vector terminan en el mismo dígito que el último valor leído.

33. Leer 10 números enteros, almacenarlos en un vector y determinar a cuánto es igual la suma de los dígitos pares de cada uno de los números leídos.

34. Leer 10 números enteros, almacenarlos en un vector y determinar cuántas veces en el vector se encuentra el dígito 2. No se olvide que el dígito 2 puede estar varias veces en un mismo número.

35. Leer 10 números enteros, almacenarlos en un vector y determinar si el promedio entero de dichos números es un número primo.

36. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos dígitos primos hay en los números leídos.

37. Leer 10 números enteros, almacenarlos en un vector y determinar a cuántos es igual el cuadrado de cada uno de los números leídos.

38. Leer 10 números enteros, almacenarlos en un vector y determinar si la semisuma entre el valor mayor y el valor menor es un número primo.

39. Leer 10 números enteros, almacenarlos en un vector y determinar si la semisuma entre el valor mayor y el valor menor es un número par.

40. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos números de los almacenados en dicho vector terminan en 15.

41. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos números de los almacenados en dicho vector comienzan con 3.

42. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos números con cantidad par de dígitos pares hay almacenados en dicho vector.

43. Leer 10 números enteros, almacenarlos en un vector y determinar en qué posiciones se encuentra el número con mayor cantidad de dígitos primos.

44. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos de los números almacenados en dicho vector pertenecen a los 100 primeros elementos de la serie de Fibonacci.

45. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos números de los almacenados en dicho vector comienzan por 34.

46. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos números de los almacenados en dicho vector son primos y comienzan por 5.

47. Leer 10 números enteros, almacenarlos en un vector y determinar en qué posiciones se encuentran los números múltiplos de 10. No utilizar el número 10 en ninguna operación.

48. Leer 10 números enteros, almacenarlos en un vector y determinar en qué posición se encuentra el número primo con mayor cantidad de dígitos pares.

49. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos números terminar en dígito primo.

50. Leer 10 números enteros, almacenarlos en un vector y determinar cuántos números de los almacenados en dicho vector comienzan en dígito primo.