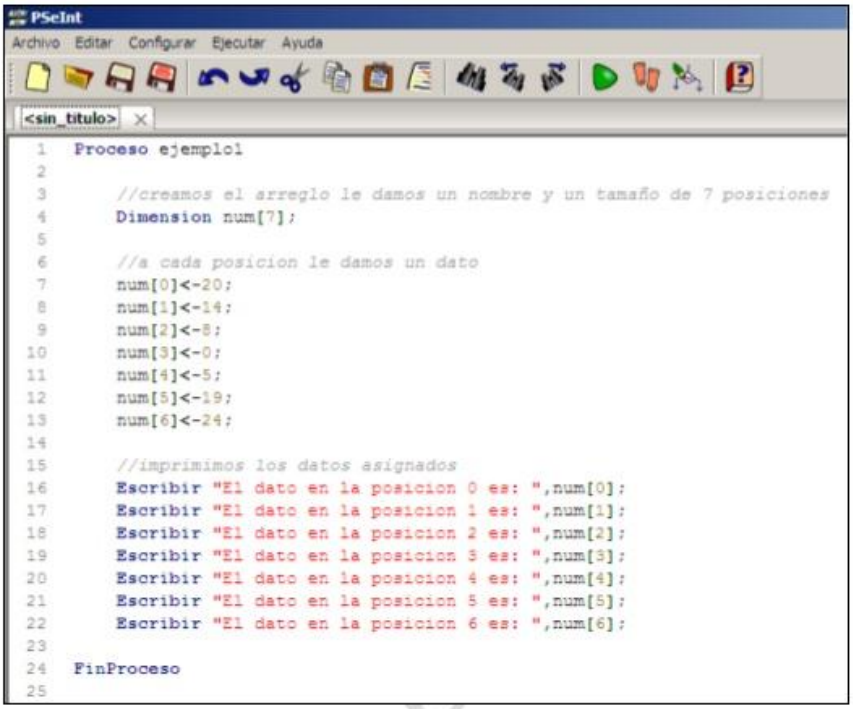
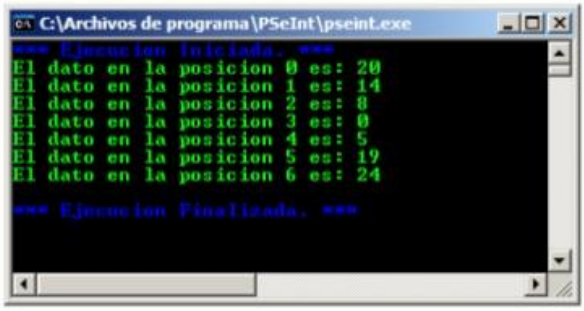
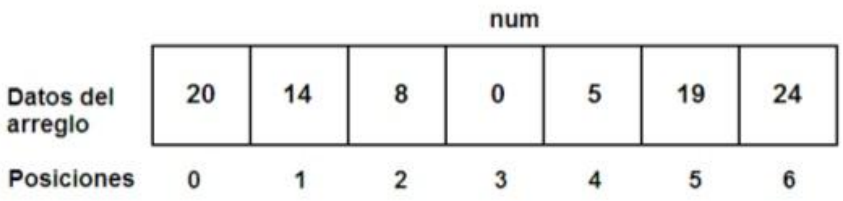
1. Crear un arreglo llamado NUM que almacene los siguientes datos: 20, 14, 8, 0, 5, 19 y 24



Salida:



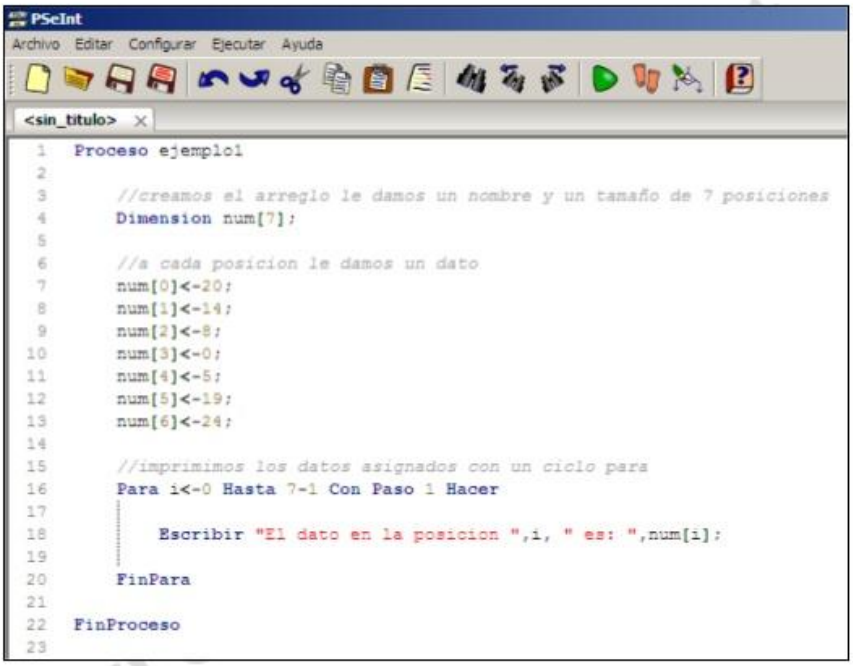
Representación grafica del anterior ejemplo:



Al utilizar arreglos en base cero los elementos validos van de **0** a **n-1**, donde n es el tamaño del arreglo. En el ejemplo 1 las posiciones del arreglo **num** entonces van desde 0 a 7-1, es decir de 0 a 6.

Los ciclos, también conocidos como *bucles* o *estructuras de control repetitivas*, juegan un papel muy importante en los arreglos. En el anterior ejemplo, imprimimos los datos a través de siete mensajes, una tarea que lleva cierto tiempo y más cuando la cantidad de datos son demasiados, por eso para facilitar el proceso, utilizamos un ciclo **Para** y así mostrar todos los datos con un sólo mensaje.

Ejemplo 2:



El ciclo **Para** nos ahorra la tarea de escribir los siete mensajes que muestran los siete datos pedidos inicialmente.

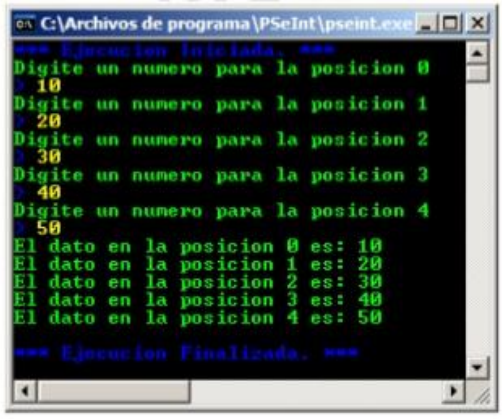
Podemos ver que la salida es la misma:

Pero no solo podemos imprimir los datos del arreglo con un ciclo, también podemos llenar con datos los arreglos con el ciclo **Para**.

Ejemplo 3:

Crear un arreglo de 5 posiciones y llénelo con los números que el usuario desee.

Salida:

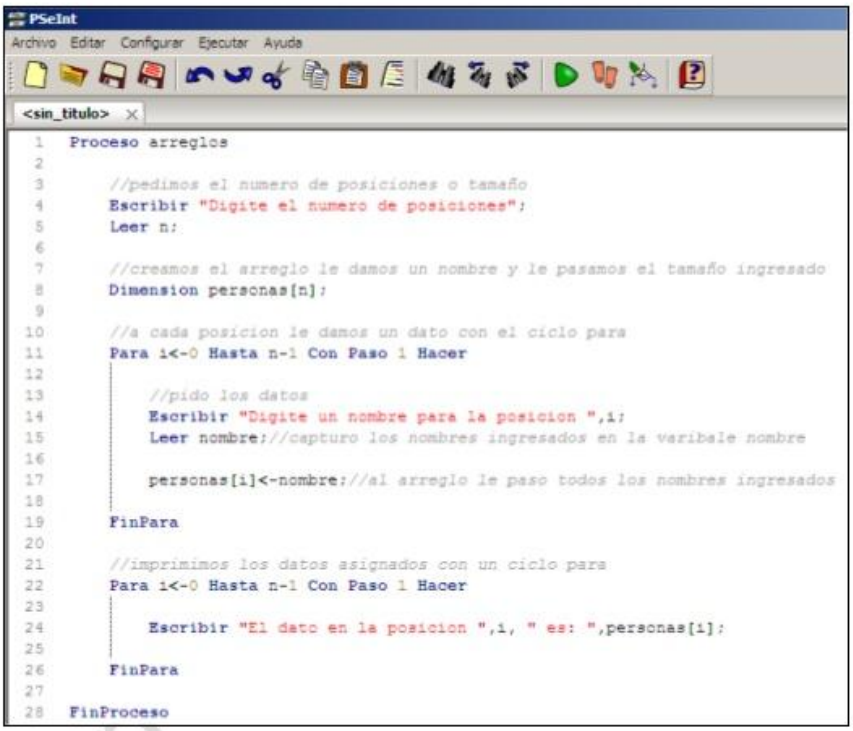


Como se puede apreciar en la salida, los números ingresados por el usuario son: **10**, **20**, **30**, **40**, **50**.

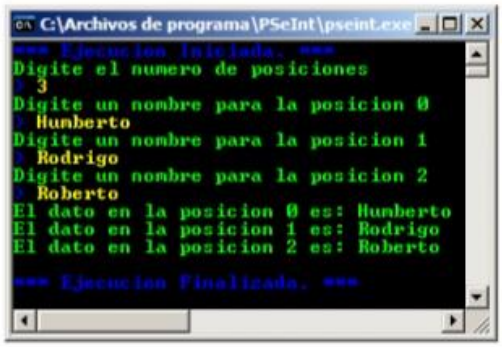
Hemos visto arreglos con datos numéricos, pero también se le pueden llenar con datos de tipo **cadenas de texto**.

Ejemplo 4:

Crear un arreglo de n posiciones y llenarlo con nombres de personas.



Salida:

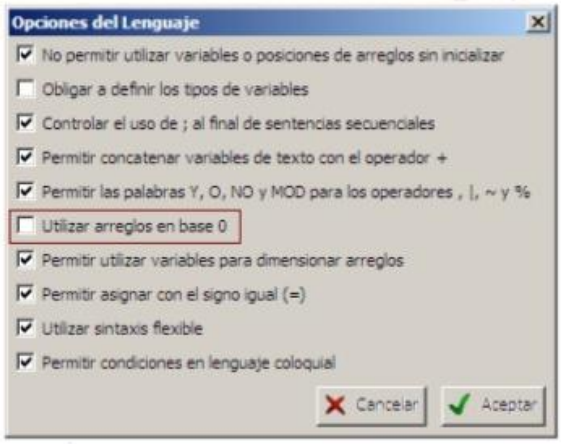


En este ejemplo el usuario eligió 3 posiciones, llenando el arreglo con los siguientes nombres: **Humberto**, **Rodrigo** y **Roberto**.

Arreglos en base 1

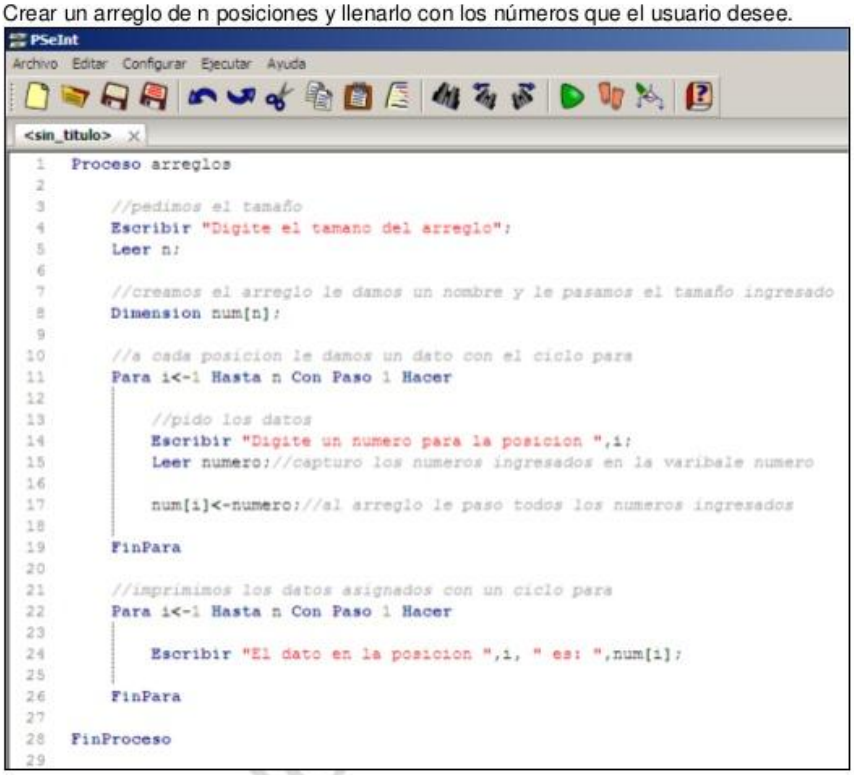
Comienzan desde **1** hasta **n**, donde n es el tamaño del arreglo.

Para programar sus algoritmos en base 1 recuerde tener desmarcada la casilla:



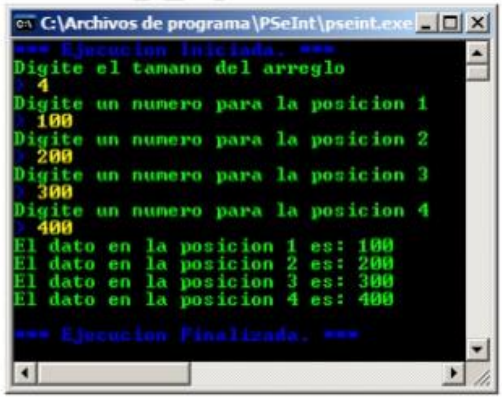
Ejemplo 5:

Crear un arreglo de n posiciones y llenarlo con los números que el usuario desee.



En el ciclo **Para** la variable ya no comienza con cero sino con **uno** y va hasta **n**.

Salida:



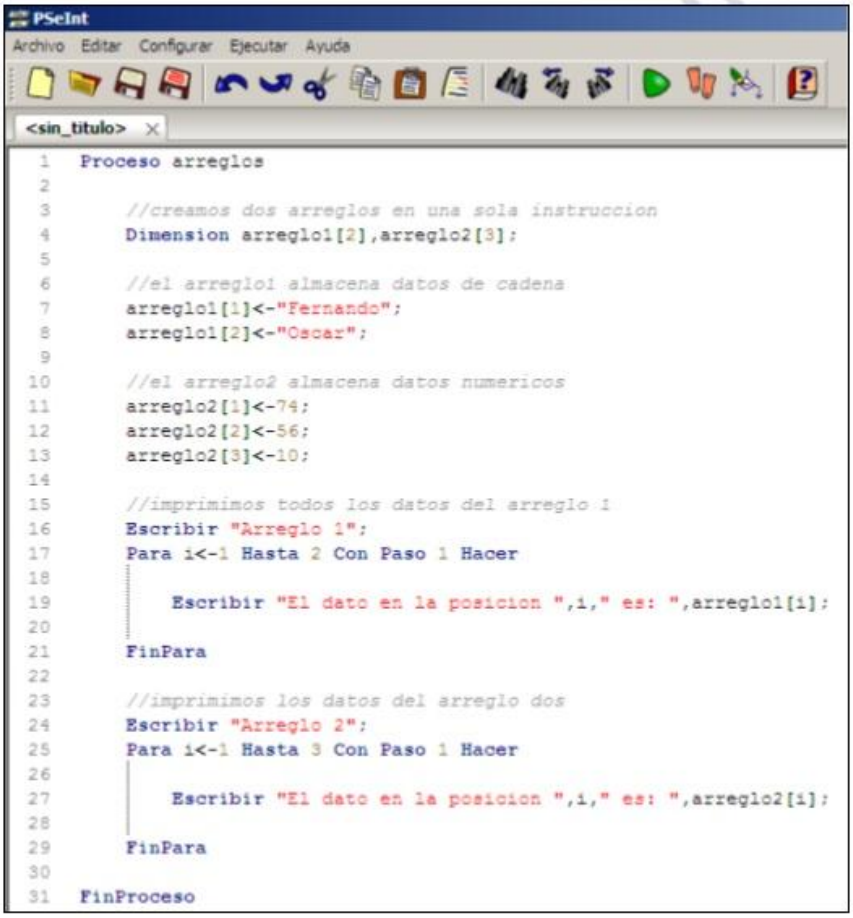
Consideraciones:

*No importa con que base trabajemos en los arreglos: uno o cero, siempre obtendremos los mismos resultados de forma eficiente, pero recomiendo al estudiante trabajar en PSeInt con arreglos en base cero ya que a la hora de aprender un lenguaje de programación como por ejemplo **Java**, los arreglos siempre van a comenzar desde cero.*

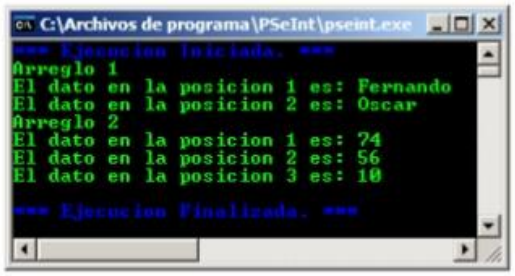
También podemos definir uno o más arreglos en una misma instrucción separándolos con una coma.

Ejemplo 6:

Crear dos arreglos uno que almacene **2** nombres y otro que almacene **3** números.



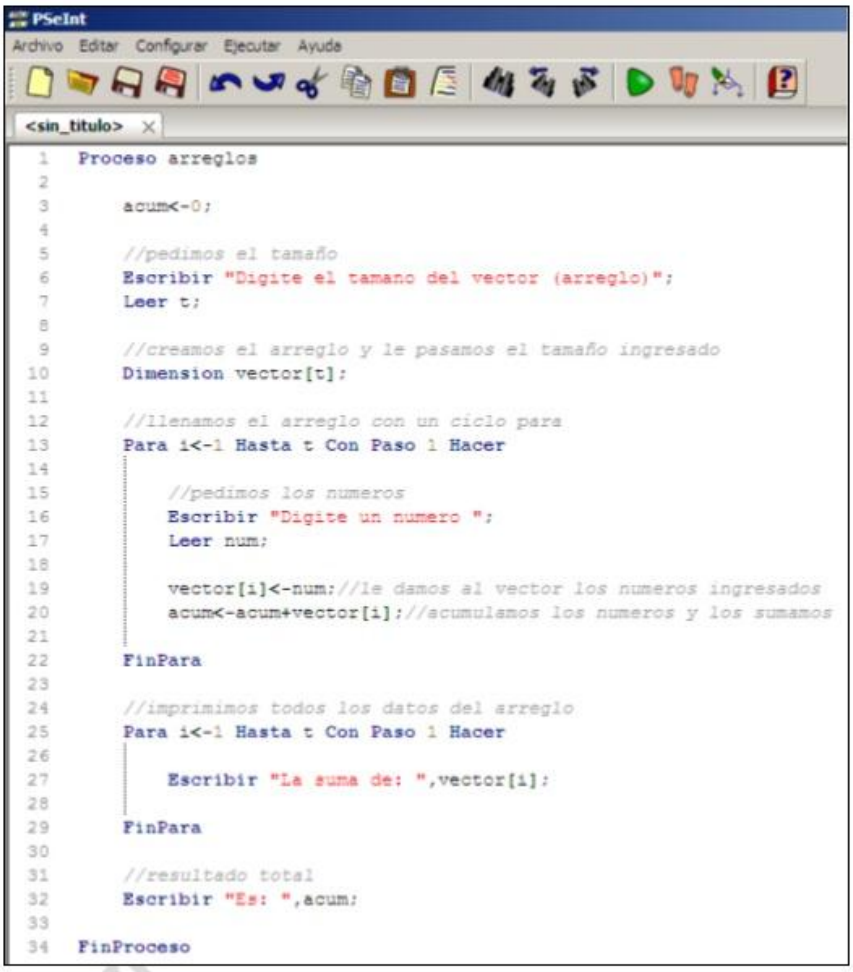
Salida:



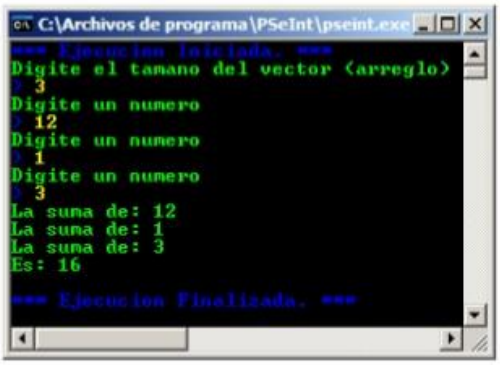
EJERCICIOS RESUELTOS CON ARREGLOS

1. Sumar todos los elementos de un arreglo de tamaño n.

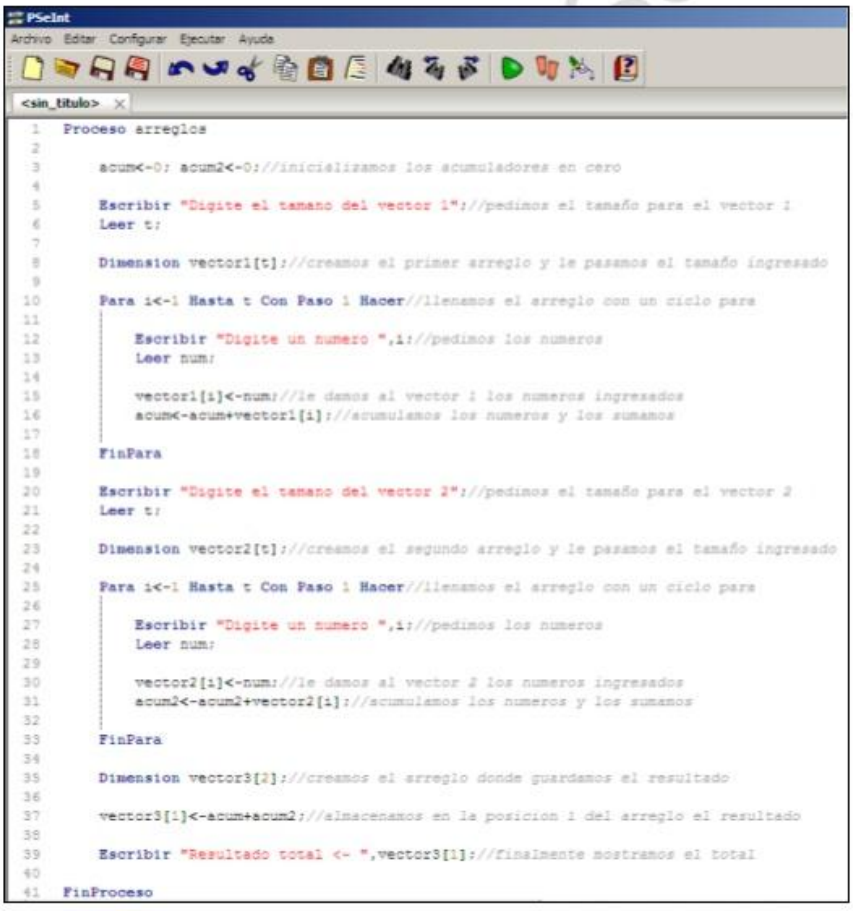
Para sumar los elementos de un vector debemos usar un acumulador inicializado en cero.


PSeInt
Archivo Editar Configurar Ejecutar Ayuda
<sin_titulo>
1 Proceso arreglos
2
3 acum<-0;
4
5 //pedimos el tamaño
6 Escribir "Digite el tamano del vector (arreglo)";
7 Leer t;
8
9 //creamos el arreglo y le pasamos el tamaño ingresado
10 Dimension vector[t];
11
12 //llenamos el arreglo con un ciclo para
13 Para i<-1 Hasta t Con Paso 1 Hacer
14
15 //pedimos los numeros
16 Escribir "Digite un numero ";
17 Leer num;
18
19 vector[i]<-num;//le damos al vector los numeros ingresados
20 acum<-acum+vector[i];//acumulamos los numeros y los sumamos
21
22 FinPara
23
24 //imprimimos todos los datos del arreglo
25 Para i<-1 Hasta t Con Paso 1 Hacer
26
27 Escribir "La suma de: ",vector[i];
28
29 FinPara
30
31 //resultado total
32 Escribir "Es: ",acum;
33
34 FinProceso

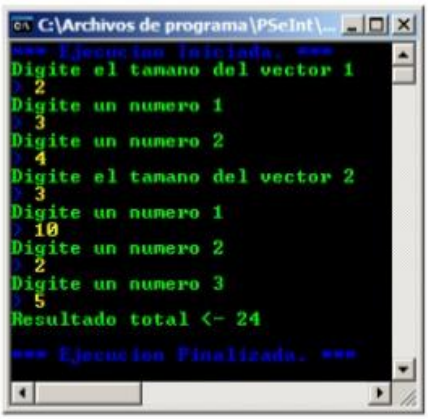
Salida:



2. Sumar los elementos de dos vectores y guardar el resultado en otro vector.

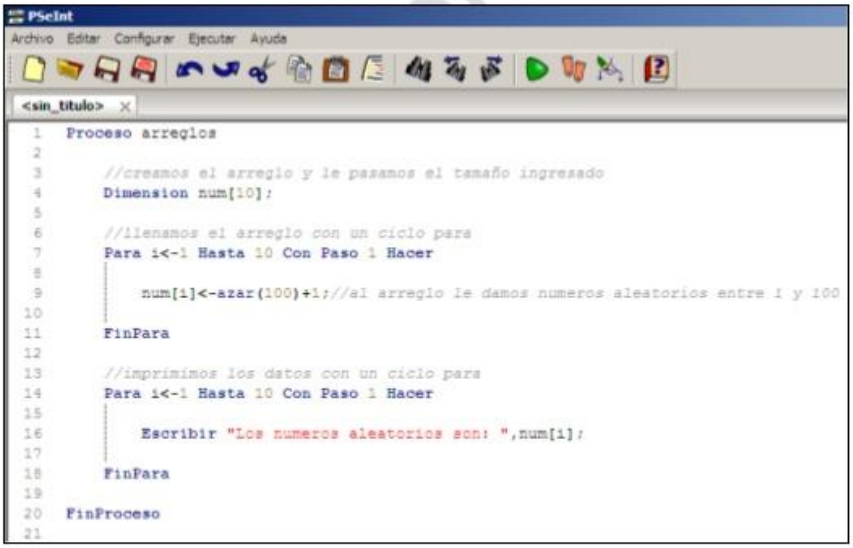


Salida:



3. Llenar un vector de 10 posiciones con números aleatorios entre 1 y 100.

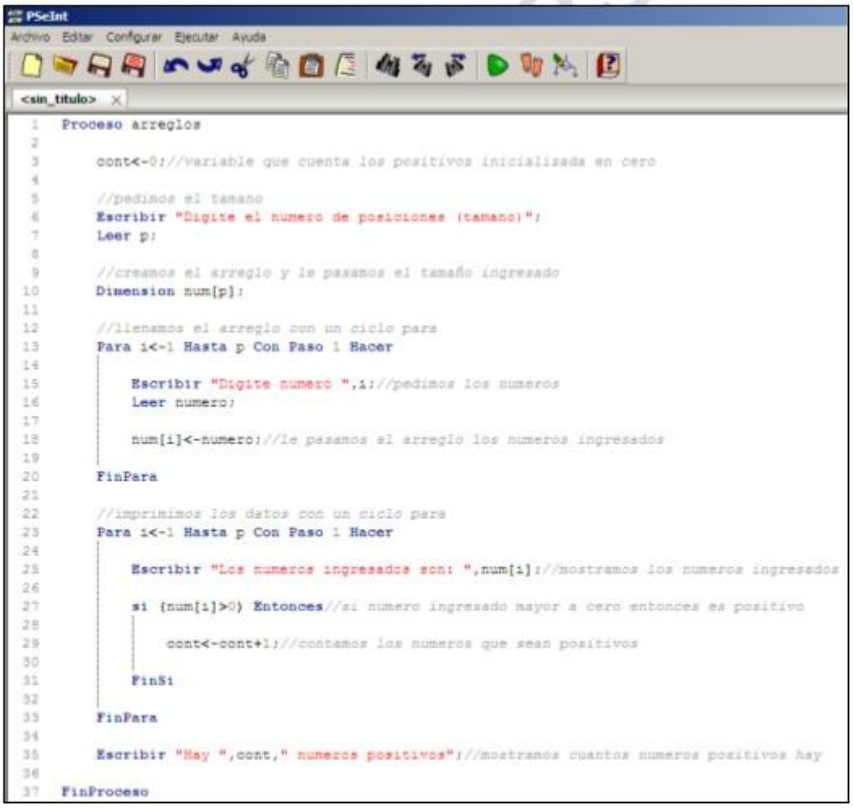
Para los números aleatorios PSeInt utiliza la función **Azar**, ésta escoge un entero aleatorio entre 0 y x-1.



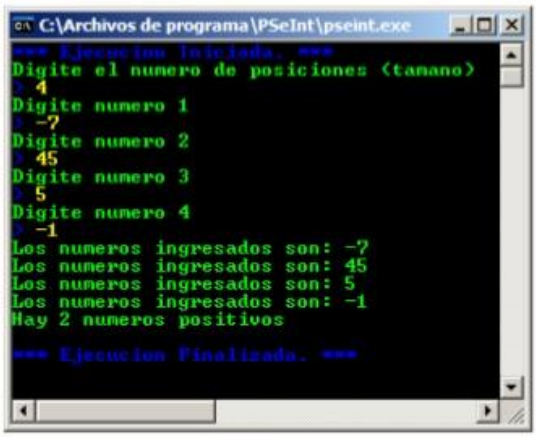
Salida:

De esta manera cada vez que ejecutemos el algoritmo arrojará distintos números.

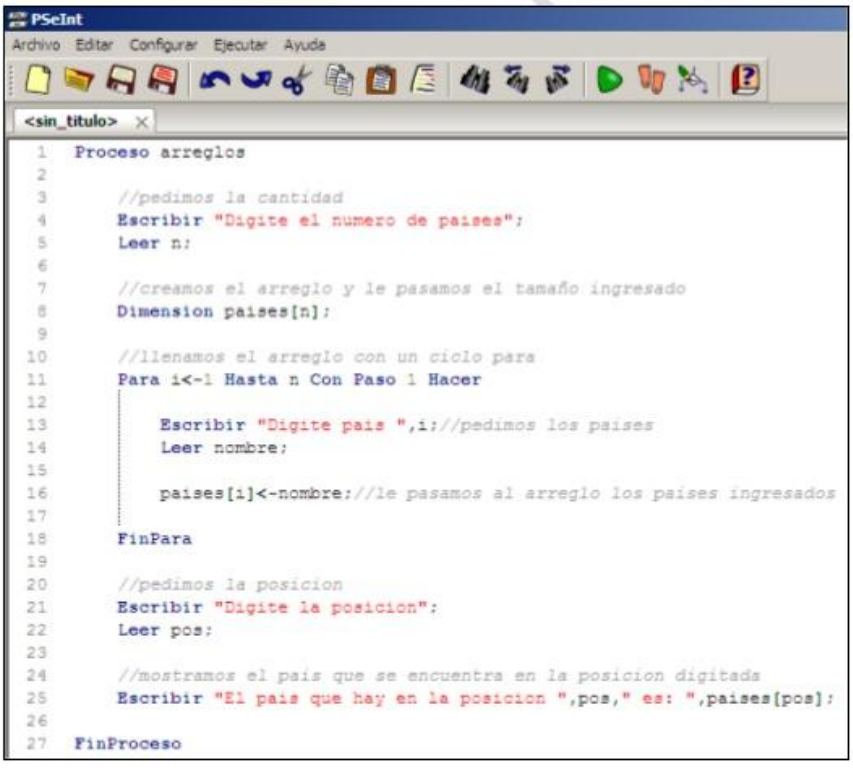
4. Llenar un vector con números enteros (números positivos ó negativos). Mostrar la cantidad de números positivos que hay en dicho arreglo.



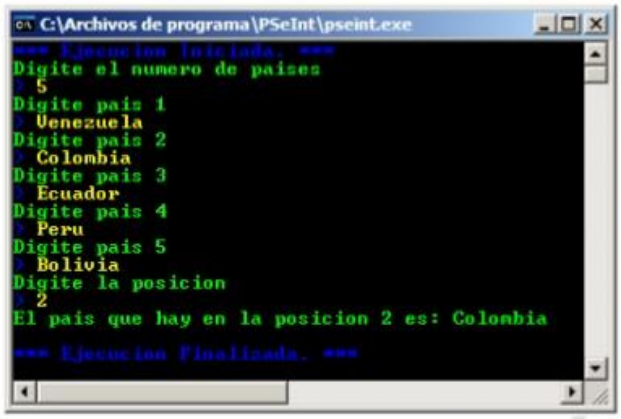
Salida:



5. Almacene en un arreglo de n posiciones nombres de países. Implementar una opción que al digitar una posición muestre el dato que contiene.



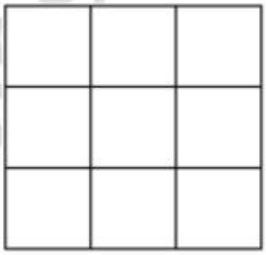
Salida:



ARREGLOS BIDIMENSIONALES (MATRICES)

Hasta ahora hemos trabajado con arreglos de una sola dimensión, es decir con un sólo índice, el índice es el número que encerramos dentro de los corchetes (el tamaño del vector).

Un arreglo bidimensional, también conocido como matriz, es parecido a una tabla ya que se compone de n filas y n columnas. Por ejemplo tenemos la siguiente tabla:



Vemos que está compuesta por tres filas y tres columnas. De esta misma forma podemos representar gráficamente a una matriz, como veremos más adelante.

Para crear una matriz en PSeInt se utiliza la palabra clave **Dimension**, seguido del nombre que la identifica y el número de filas y columnas.

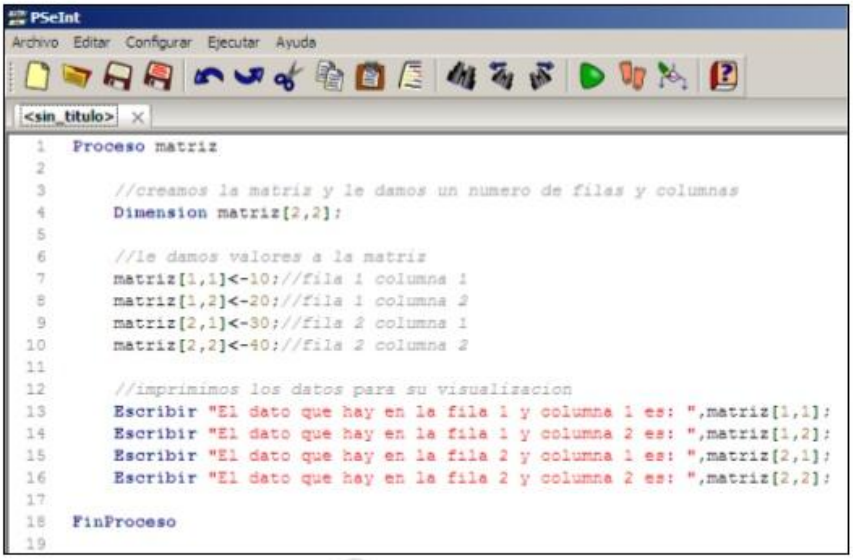
Sintaxis:

Dimension identificador [filas,columnas];

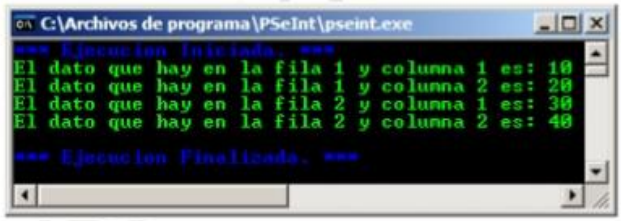
Para comprender mejor el concepto de matrices se realizaran algunos ejemplos y ejercicios.

Ejemplo 1:

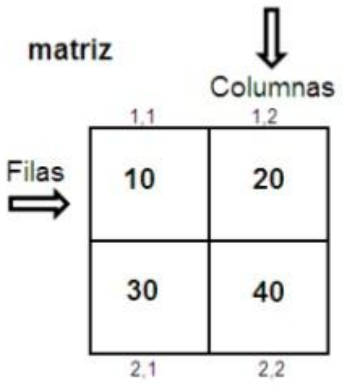
Crear una matriz **2x2** que almacene los siguientes valores: **10**, **20**, **30**, **40**.



Salida:



Representación grafica del anterior ejemplo:



Las filas son **horizontales** y las columnas **verticales**.

En la fila 1 columna 1 el dato es: **10**

En la fila 1 columna 2 el dato es: **20**

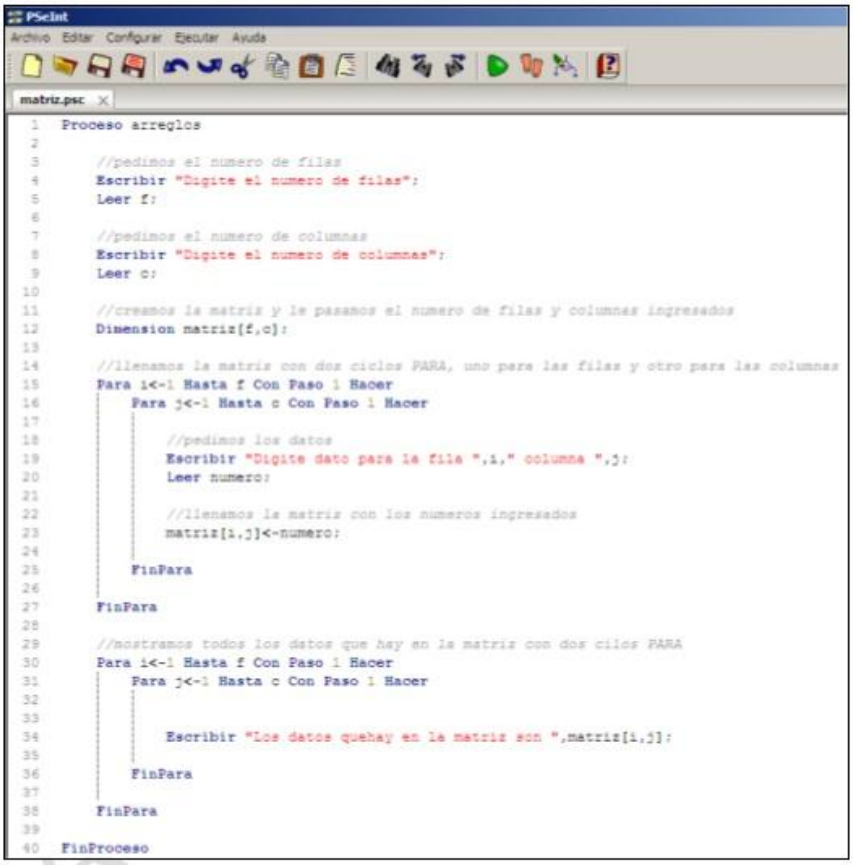
En la fila 2 columna 1 el dato es: **30**

En la fila 2 columna 2 el dato es: **40**

Así como en los arreglos unidimensionales llenábamos el vector con un ciclo **Para**, en las matrices también lo podemos hacer, sólo que ya no se utilizará un ciclo sino **dos**, uno para las filas y otro para las columnas. También los datos se muestran con dos ciclos.

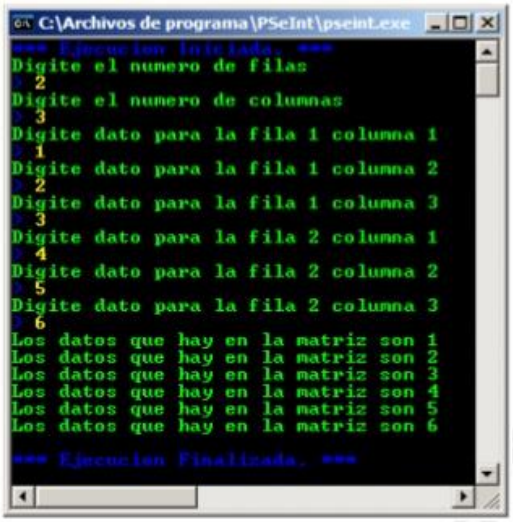
Ejemplo 2:

Crear una matriz de n filas y n columnas. Llenar la matriz con los números que el usuario desee.



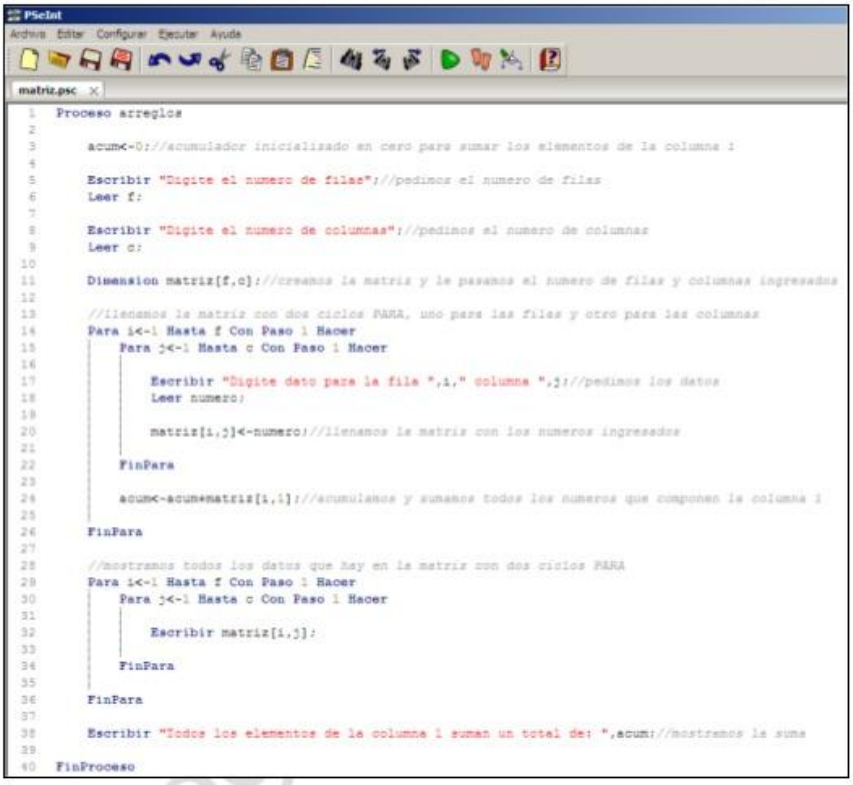

```
1  Proceso arreglos
2
3      //pedimos el numero de filas
4      Escribir "Digite el numero de filas";
5      Leer f;
6
7      //pedimos el numero de columnas
8      Escribir "Digite el numero de columnas";
9      Leer c;
10
11     //creamos la matriz y le pasamos el numero de filas y columnas ingresados
12     Dimension matriz[f,c];
13
14     //llenamos la matriz con dos ciclos PARA, uno para las filas y otro para las columnas
15     Para i<-1 Hasta f Con Paso 1 Hacer
16         Para j<-1 Hasta c Con Paso 1 Hacer
17
18             //pedimos los datos
19             Escribir "Digite dato para la fila ",i," columna ",j;
20             Leer numero;
21
22             //llenamos la matriz con los numeros ingresados
23             matriz[i,j]<-numero;
24
25         FinPara
26
27     FinPara
28
29     //mostramos todos los datos que hay en la matriz con dos ciclos PARA
30     Para i<-1 Hasta f Con Paso 1 Hacer
31         Para j<-1 Hasta c Con Paso 1 Hacer
32
33
34             Escribir "Los datos quehay en la matriz son ",matriz[i,j];
35
36         FinPara
37
38     FinPara
39
40 FinProceso
```

Salida:



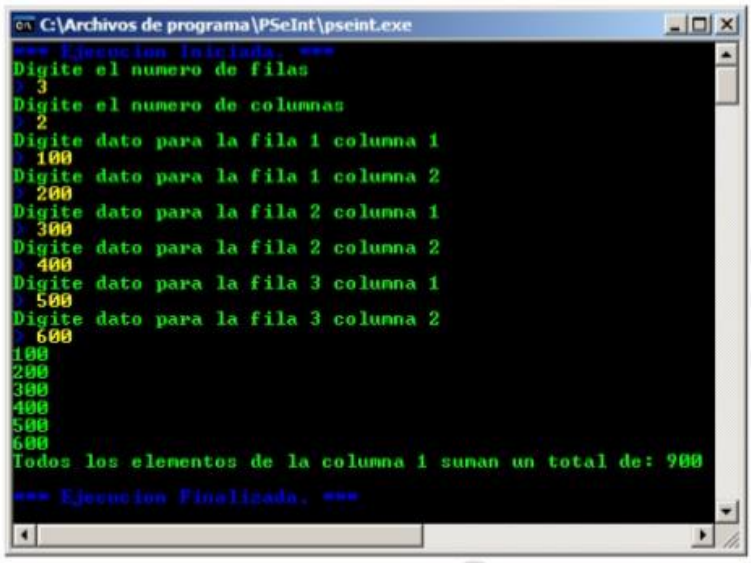
EJERCICIO CON MATRIZ

1. Crear una matriz n x n y llenarla con los números que el usuario desee. Sume todos los números que componga la columna 1.

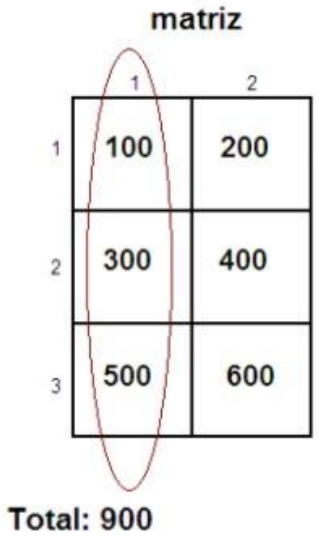



```
Proceso arreglos

    acum<-0;//acumulador inicializado en cero para sumar los elementos de la columna 1

    Escribir "Digite el numero de filas";//pedimos el numero de filas
    Leer f;

    Escribir "Digite el numero de columnas";//pedimos el numero de columnas
    Leer c;

    Dimension matriz[f,c];//creamos la matriz y le pasamos el numero de filas y columnas ingresados

    //llenamos la matriz con dos ciclos PARA, uno para las filas y otro para las columnas
    Para i<-1 Hasta f Con Paso 1 Hacer
        Para j<-1 Hasta c Con Paso 1 Hacer

            Escribir "Digite dato para la fila ",i," columna ",j;//pedimos los datos
            Leer numero;

            matriz[i,j]<-numero;//llenamos la matriz con los numeros ingresados

        FinPara

        acum<-acum+matriz[i,1];//acumulamos y sumamos todos los numeros que componen la columna 1

    FinPara

    //mostramos todos los datos que hay en la matriz con dos ciclos PARA
    Para i<-1 Hasta f Con Paso 1 Hacer
        Para j<-1 Hasta c Con Paso 1 Hacer

            Escribir matriz[i,j];

        FinPara

    FinPara

    Escribir "Todos los elementos de la columna 1 suman un total de: ",acum;//mostramos la suma

FinProceso
```

Salida:



Representación grafica del anterior ejercicio:



Total: 900

2. Llenar una matriz de 3 x 3 completamente de números aleatorios entre 0 y 9.

PSeInt

Archivo Editar Configurar Ejecutar Ayuda

matriz.psc

```
1   Proceso arreglos
2
3       //creamos la matriz y le pasamos 3 filas y 3 columnas
4       Dimension matriz[3,3];
5
6       //llenamos la matriz con dos ciclos PARA, uno para las filas y otro para las columnas
7       Para i<-1 Hasta 3 Con Paso 1 Hacer
8           Para j<-1 Hasta 3 Con Paso 1 Hacer
9
10              //a la matriz le damos numeros aleatorios entre 0 y 9
11              matriz[i,j]<-azar(10);
12
13          FinPara
14
15      FinPara
16
17      //mostramos todos los datos que hay en la matriz
18      Para i<-1 Hasta 3 Con Paso 1 Hacer
19
20          Escribir matriz[i,1]," ",matriz[i,2]," ",matriz[i,3];
21
22      FinPara
23
24  FinProceso
```

Salida: