

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

```
PSeInt
Archivo Editar Configurar Ejecutar Ayuda
<sin_titulo> x
1 Proceso ejemplo
2
3 //creamos el arreglo le damos un nombre y un tamaño de 7 posiciones
4 Dimension num[7];
5
6 //a cada posicion le damos un dato
7 num[0]<-20;
8 num[1]<-14;
9 num[2]<-8;
10 num[3]<-0;
11 num[4]<-5;
12 num[5]<-19;
13 num[6]<-24;
14
15 //imprimimos los datos asignados
16 Escribir "El dato en la posicion 0 es: ",num[0];
17 Escribir "El dato en la posicion 1 es: ",num[1];
18 Escribir "El dato en la posicion 2 es: ",num[2];
19 Escribir "El dato en la posicion 3 es: ",num[3];
20 Escribir "El dato en la posicion 4 es: ",num[4];
21 Escribir "El dato en la posicion 5 es: ",num[5];
22 Escribir "El dato en la posicion 6 es: ",num[6];
23
24 FinProceso
25
```

Salida:

```
C:\Archivos de programa\PSeInt\pseint.exe
*** Ejecucion iniciada. ***
El dato en la posicion 0 es: 20
El dato en la posicion 1 es: 14
El dato en la posicion 2 es: 8
El dato en la posicion 3 es: 0
El dato en la posicion 4 es: 5
El dato en la posicion 5 es: 19
El dato en la posicion 6 es: 24
*** Ejecucion Finalizada. ***
```

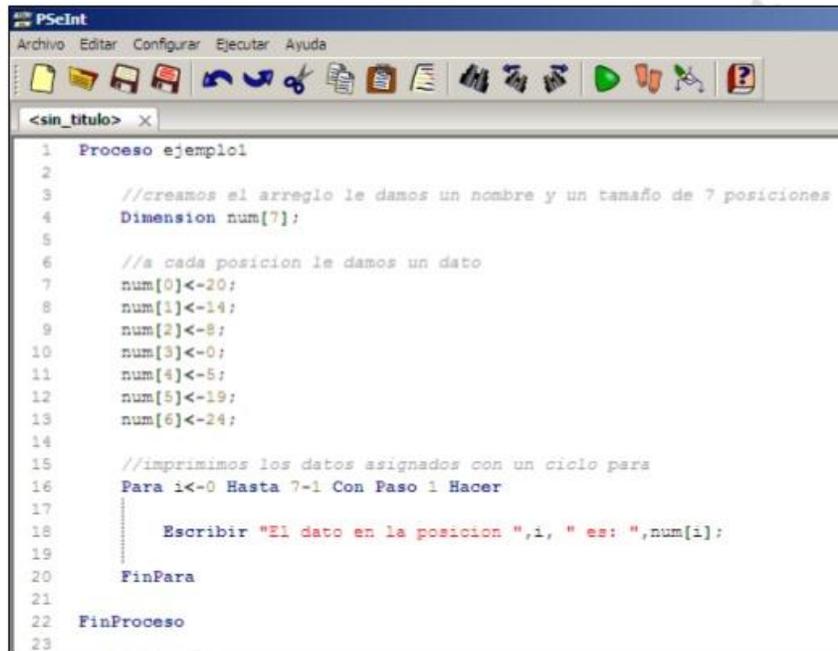
Representación grafica del anterior ejemplo:

	num						
Datos del arreglo	20	14	8	0	5	19	24
Posiciones	0	1	2	3	4	5	6

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:



```
1  Proceso ejemplo1
2
3      //creamos el arreglo le damos un nombre y un tamaño de 7 posiciones
4      Dimension num[7];
5
6      //a cada posicion le damos un dato
7      num[0]<-20;
8      num[1]<-14;
9      num[2]<-8;
10     num[3]<-0;
11     num[4]<-5;
12     num[5]<-19;
13     num[6]<-24;
14
15     //imprimimos los datos asignados con un ciclo para
16     Para i<-0 Hasta 7-1 Con Paso 1 Hacer
17
18         Escribir "El dato en la posicion ",i, " es: ",num[i];
19
20     FinPara
21
22 FinProceso
23
```

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:

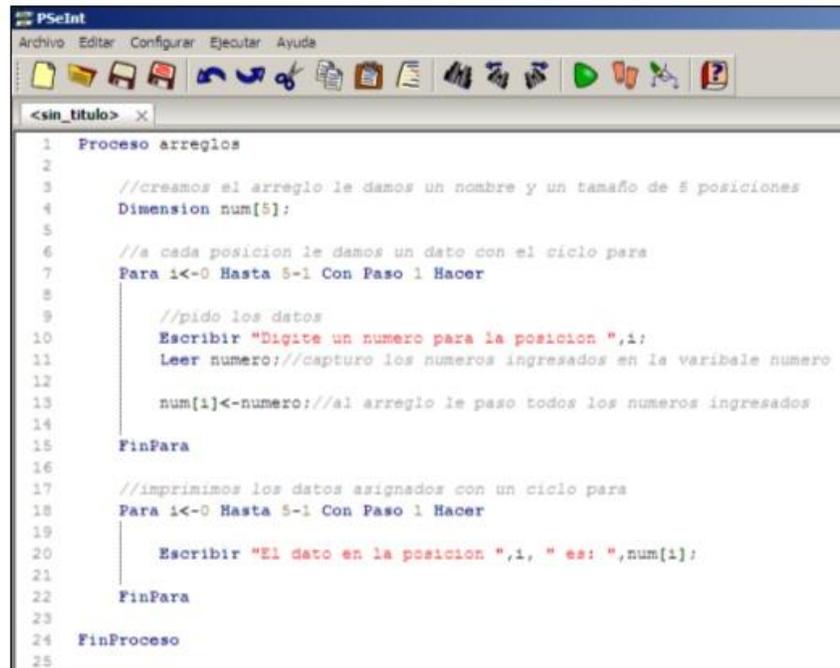


```
C:\Archivos de programa\PSeInt\pseint...
*** Ejecucion Iniciada. ***
El dato en la posicion 0 es: 20
El dato en la posicion 1 es: 14
El dato en la posicion 2 es: 8
El dato en la posicion 3 es: 0
El dato en la posicion 4 es: 5
El dato en la posicion 5 es: 19
El dato en la posicion 6 es: 24
*** Ejecucion Finalizada. ***
```

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.



```
1 Proceso arreglos
2
3 //creamos el arreglo le damos un nombre y un tamaño de 5 posiciones
4 Dimension num[5]:
5
6 //a cada posición le damos un dato con el ciclo para
7 Para i<-0 Hasta 5-1 Con Paso 1 Hacer
8
9     //pido los datos
10    Escribir "Digite un número para la posición ",i:
11    Leer numero://capturo los números ingresados en la variable numero
12
13    num[i]<-numero://al arreglo le paso todos los números ingresados
14
15 FinPara
16
17 //imprimimos los datos asignados con un ciclo para
18 Para i<-0 Hasta 5-1 Con Paso 1 Hacer
19
20     Escribir "El dato en la posición ",i, " es: ",num[i]:
21
22 FinPara
23
24 FinProceso
25
```

Salida:



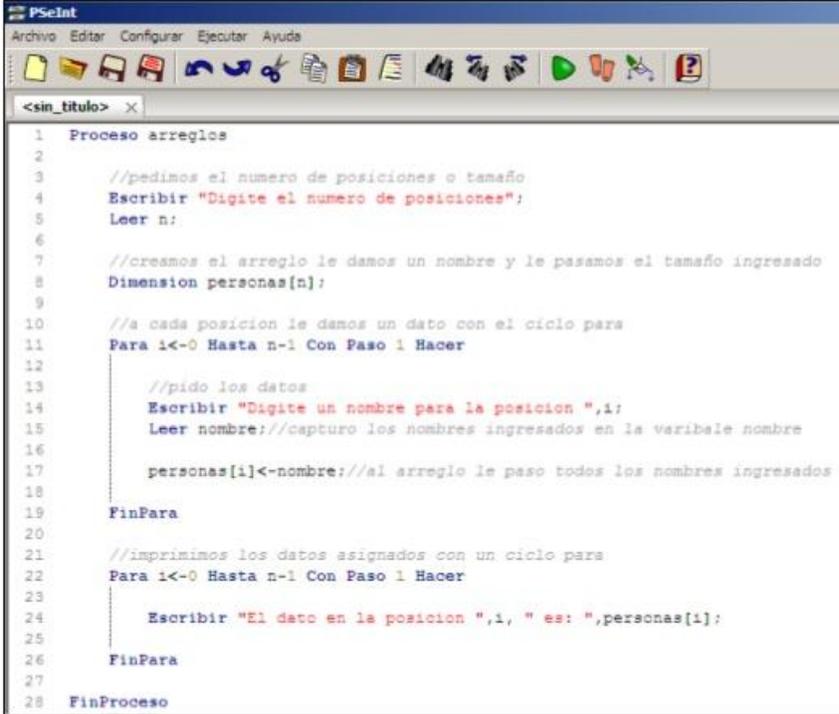
```
C:\Archivos de programa\PSeInt\pseint.exe
*** Ejecucion Iniciada. ***
Digite un número para la posición 0
10
Digite un número para la posición 1
20
Digite un número para la posición 2
30
Digite un número para la posición 3
40
Digite un número para la posición 4
50
El dato en la posición 0 es: 10
El dato en la posición 1 es: 20
El dato en la posición 2 es: 30
El dato en la posición 3 es: 40
El dato en la posición 4 es: 50
*** Ejecucion Finalizada. ***
```

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.**

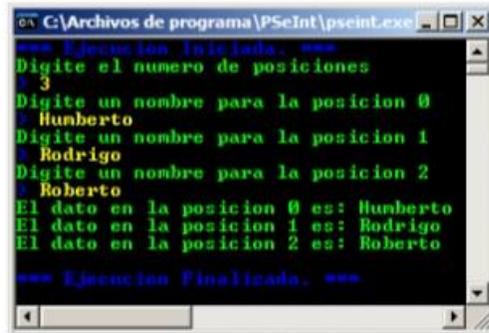
Ejemplo 4:

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



```
1  Proceso arreglos
2
3      //pedimos el numero de posiciones o tamaño
4  Escribir "Digite el numero de posiciones";
5  Leer n:
6
7  //creamos el arreglo le damos un nombre y le pasamos el tamaño ingresado
8  Dimension personas[n];
9
10 //a cada posición le damos un dato con el ciclo para
11 Para i<-0 Hasta n-1 Con Paso 1 Hacer
12
13     //pido los datos
14     Escribir "Digite un nombre para la posición ",i;
15     Leer nombre;//capturo los nombres ingresados en la variable nombre
16
17     personas[i]<-nombre;//al arreglo le paso todos los nombres ingresados
18
19 FinPara
20
21 //imprimimos los datos asignados con un ciclo para
22 Para i<-0 Hasta n-1 Con Paso 1 Hacer
23
24     Escribir "El dato en la posición ",i, " es: ",personas[i];
25
26 FinPara
27
28 FinProceso
```

Salida:



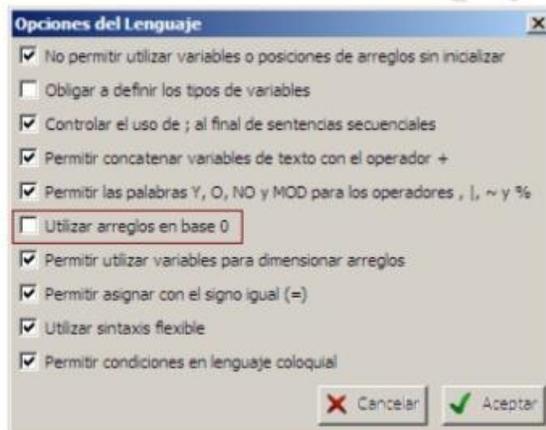
```
C:\Archivos de programa\PSeInt\pseint.exe
*** Ejecucion Iniciada. ***
Digite el numero de posiciones
3
Digite un nombre para la posicion 0
Humberto
Digite un nombre para la posicion 1
Rodrigo
Digite un nombre para la posicion 2
Roberto
El dato en la posicion 0 es: Humberto
El dato en la posicion 1 es: Rodrigo
El dato en la posicion 2 es: Roberto
*** Ejecucion Finalizada. ***
```

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

Arreglos en base 1

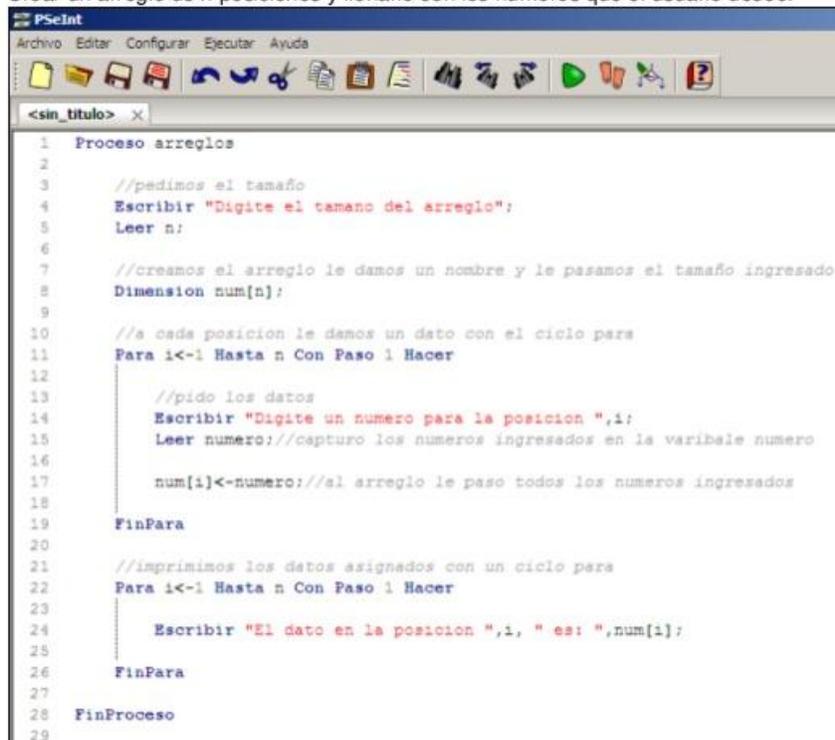
Comienzan desde **1** hasta **n**, donde **n** es el tamaño del arreglo.

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



Ejemplo 5:

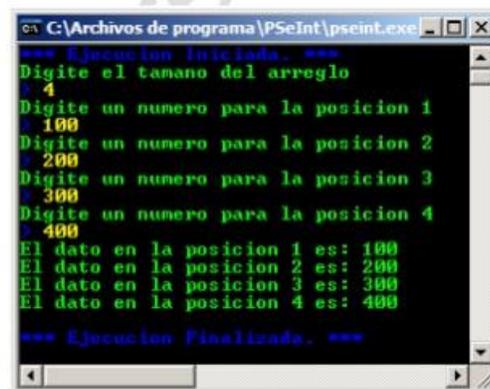
Crear un arreglo de n posiciones y llenarlo con los números que el usuario desee.



```
1 Proceso arreglos
2
3 //pedimos el tamaño
4 Escribir "Digite el tamaño del arreglo":
5 Leer n:
6
7 //creamos el arreglo le damos un nombre y le pasamos el tamaño ingresado
8 Dimension num[n]:
9
10 //a cada posición le damos un dato con el ciclo para
11 Para i<-1 Hasta n Con Paso 1 Hacer
12
13     //pido los datos
14     Escribir "Digite un número para la posición ",i:
15     Leer numero://capturo los números ingresados en la variable numero
16
17     num[i]<-numero://el arreglo le paso todos los números ingresados
18
19 FinPara
20
21 //imprimimos los datos asignados con un ciclo para
22 Para i<-1 Hasta n Con Paso 1 Hacer
23
24     Escribir "El dato en la posición ",i, " es: ",num[i]:
25
26 FinPara
27
28 FinProceso
29
```

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

Salida:



```
C:\Archivos de programa\PSeInt\pseint.exe
*** Ejecucion Iniciada. ***
Digite el tamaño del arreglo
4
Digite un número para la posición 1
100
Digite un número para la posición 2
200
Digite un número para la posición 3
300
Digite un número para la posición 4
400
El dato en la posición 1 es: 100
El dato en la posición 2 es: 200
El dato en la posición 3 es: 300
El dato en la posición 4 es: 400
*** Ejecucion Finalizada. ***
```

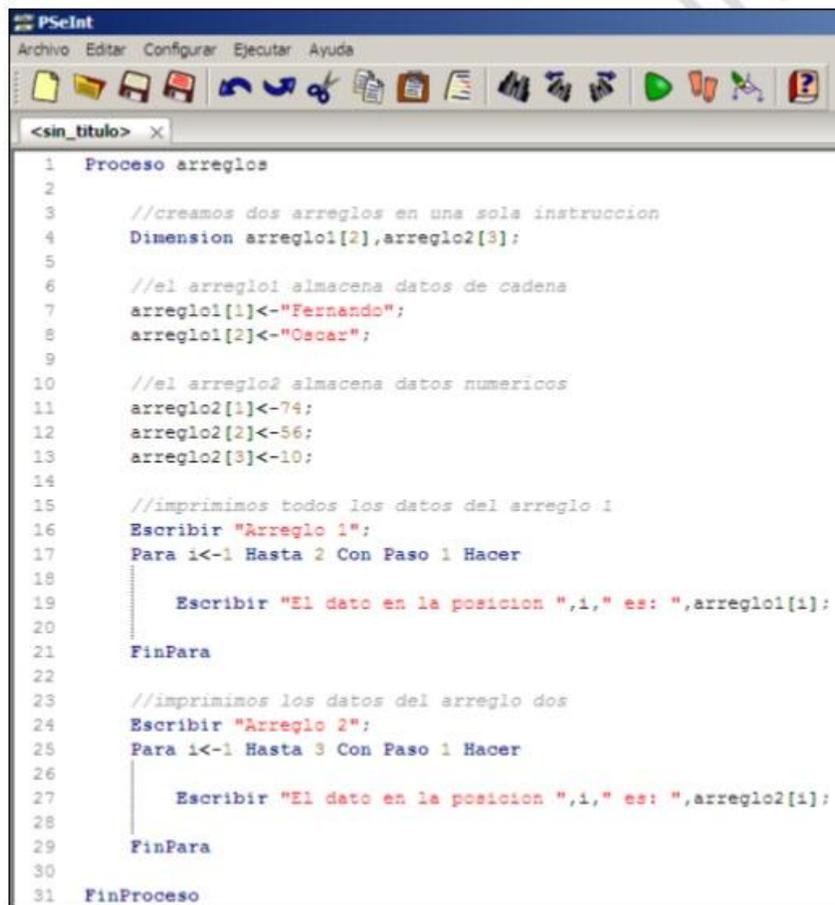
Consideraciones:

*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.



```
1  Proceso arreglos
2
3      //creamos dos arreglos en una sola instruccion
4      Dimension arreglo1[2],arreglo2[3];
5
6      //el arreglo1 almacena datos de cadena
7      arreglo1[1]<-"Fernando";
8      arreglo1[2]<-"Oscar";
9
10     //el arreglo2 almacena datos numericos
11     arreglo2[1]<-74;
12     arreglo2[2]<-56;
13     arreglo2[3]<-10;
14
15     //imprimimos todos los datos del arreglo 1
16     Escribir "Arreglo 1";
17     Para i<-1 Hasta 2 Con Paso 1 Hacer
18     |
19     |     Escribir "El dato en la posicion ",i," es: ",arreglo1[i];
20     |
21     FinPara
22
23     //imprimimos los datos del arreglo dos
24     Escribir "Arreglo 2";
25     Para i<-1 Hasta 3 Con Paso 1 Hacer
26     |
27     |     Escribir "El dato en la posicion ",i," es: ",arreglo2[i];
28     |
29     FinPara
30
31 FinProceso
```

Salida:



```
C:\Archivos de programa\PSeInt\pseint.exe
*** Ejecucion Inicializada. ***
Arreglo 1
El dato en la posicion 1 es: Fernando
El dato en la posicion 2 es: Oscar
Arreglo 2
El dato en la posicion 1 es: 74
El dato en la posicion 2 es: 56
El dato en la posicion 3 es: 10
*** Ejecucion Finalizada. ***
```

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.

Arreglos en PSeInt

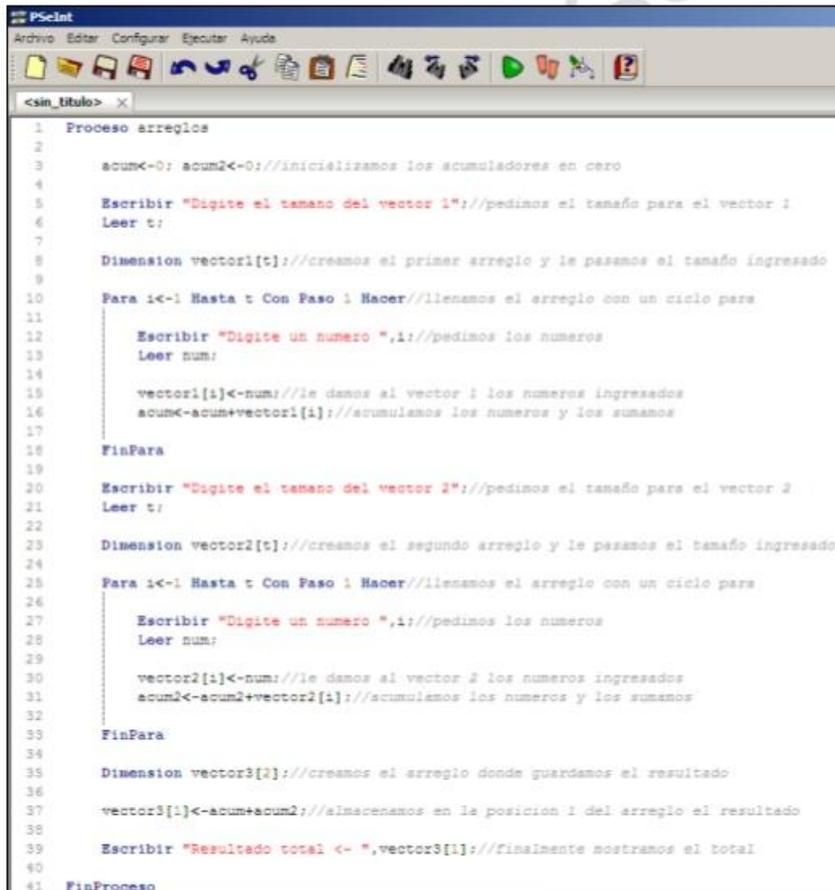
```
PSelnt
Archivo Editar Configurar Ejecutar Ayuda
<sin_titulo> X
1 Proceso arreglos
2
3 acum<-0;
4
5 //pedimos el tamaño
6 Escribir "Digite el tamaño del vector (arreglo)";
7 Leer t;
8
9 //creamos el arreglo y le pasamos el tamaño ingresado
10 Dimension vector[t];
11
12 //llenamos el arreglo con un ciclo para
13 Para i<-1 Hasta t Con Paso 1 Hacer
14
15     //pedimos los números
16     Escribir "Digite un número ";
17     Leer num;
18
19     vector[i]<-num;//le damos al vector los números ingresados
20     acum<-acum+vector[i];//acumulamos los números 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:



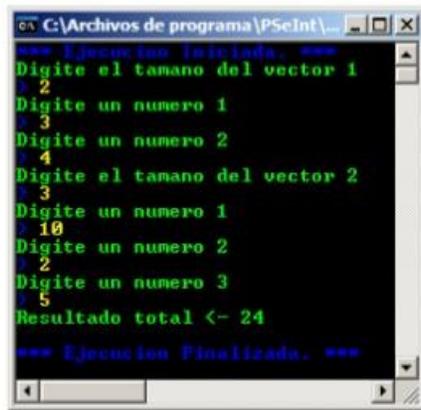
```
C:\Archivos de programa\PSeInt\pseint.exe
*** Ejecucion Iniciada. ***
Digite el tamaño del vector (arreglo)
: 3
Digite un número
: 12
Digite un número
: 1
Digite un número
: 3
La suma de: 12
La suma de: 1
La suma de: 3
Es: 16
*** Ejecucion Finalizada. ***
```

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



```
PSeInt
Archivo Editor Configurador Ejecutar Ayuda
<sin_titulo> x
1 Proceso arreglos
2
3 acum<-0; acum2<-0;//inicializamos los acumuladores en cero
4
5 Escribir "Digite el tamaño del vector 1";//pedimos el tamaño para el vector 1
6 Leer t;
7
8 Dimension vector1[t];//creamos el primer arreglo y le pasamos el tamaño ingresado
9
10 Para i<-1 Hasta t Con Paso 1 Hacer//llenamos el arreglo con un ciclo para
11
12 Escribir "Digite un número ",i;//pedimos los números
13 Leer num;
14
15 vector1[i]<-num;//le damos al vector 1 los números ingresados
16 acum<-acum+vector1[i];//acumulamos los números y los sumamos
17
18 FinPara
19
20 Escribir "Digite el tamaño del vector 2";//pedimos el tamaño para el vector 2
21 Leer t;
22
23 Dimension vector2[t];//creamos el segundo arreglo y le pasamos el tamaño ingresado
24
25 Para i<-1 Hasta t Con Paso 1 Hacer//llenamos el arreglo con un ciclo para
26
27 Escribir "Digite un número ",i;//pedimos los números
28 Leer num;
29
30 vector2[i]<-num;//le damos al vector 2 los números ingresados
31 acum2<-acum2+vector2[i];//acumulamos los números y los sumamos
32
33 FinPara
34
35 Dimension vector3[t];//creamos el arreglo donde guardamos el resultado
36
37 vector3[i]<-acum+acum2;//almacenamos en la posición i del arreglo el resultado
38
39 Escribir "Resultado total <- ",vector3[1];//finalmente mostramos el total
40
41 FinProceso
```

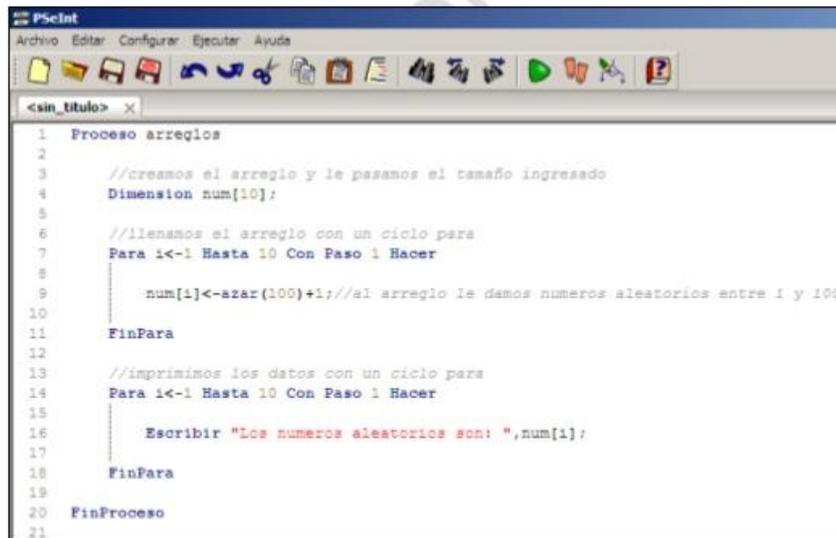
Salida:



```
C:\Archivos de programa\PSeInt\...
*** Ejecucion Iniciada. ***
Digite el tamaño del vector 1
2
Digite un número 1
3
Digite un número 2
4
Digite el tamaño del vector 2
3
Digite un número 1
10
Digite un número 2
2
Digite un número 3
5
Resultado total <- 24
*** Ejecucion Finalizada. ***
```

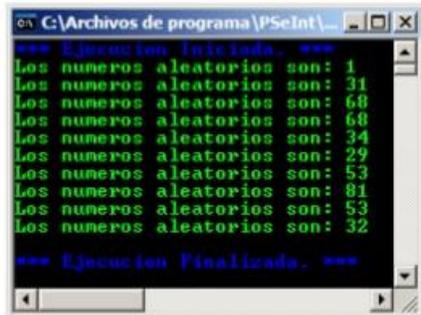
3. Llenar un vector de 10 posiciones con números aleatorios entre 1 y 100.

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



```
PSeInt
Archivo  Editar  Configurar  Ejecutar  Ayuda
<sin_titulo> x
1  Proceso arreglos
2
3  //creamos el arreglo y le pasamos el tamaño ingresado
4  Dimension num[10];
5
6  //llenamos el arreglo con un ciclo para
7  Para i<-1 Hasta 10 Con Paso 1 Hacer
8
9      num[i]<-azar(100)+1;//al arreglo le damos numeros aleatorios entre 1 y 100
10
11  FinPara
12
13  //imprimimos los datos con un ciclo para
14  Para i<-1 Hasta 10 Con Paso 1 Hacer
15
16      Escribir "Los numeros aleatorios son: ",num[i];
17
18  FinPara
19
20  FinProceso
21
```

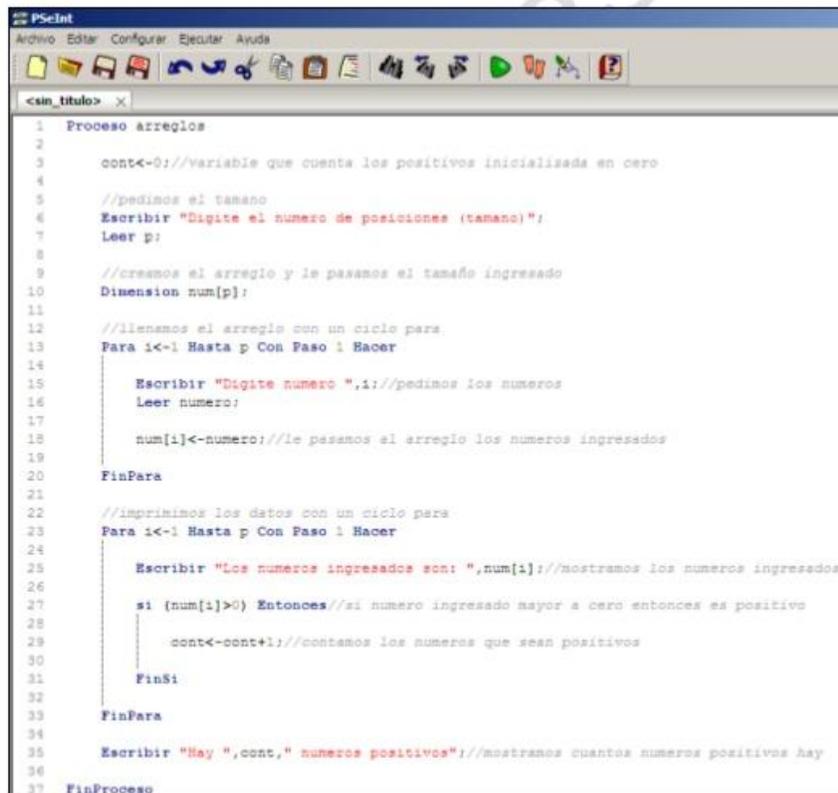
Salida:



```
*** Ejecucion Iniciada. ***
Los numeros aleatorios son: 1
Los numeros aleatorios son: 31
Los numeros aleatorios son: 68
Los numeros aleatorios son: 60
Los numeros aleatorios son: 34
Los numeros aleatorios son: 29
Los numeros aleatorios son: 53
Los numeros aleatorios son: 81
Los numeros aleatorios son: 53
Los numeros aleatorios son: 32
*** Ejecucion Finalizado. ***
```

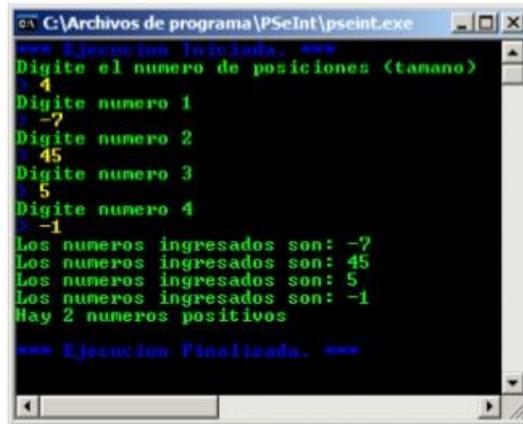
De esta manera cada vez que ejecutemos el algoritmo arrojará distintos números.

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



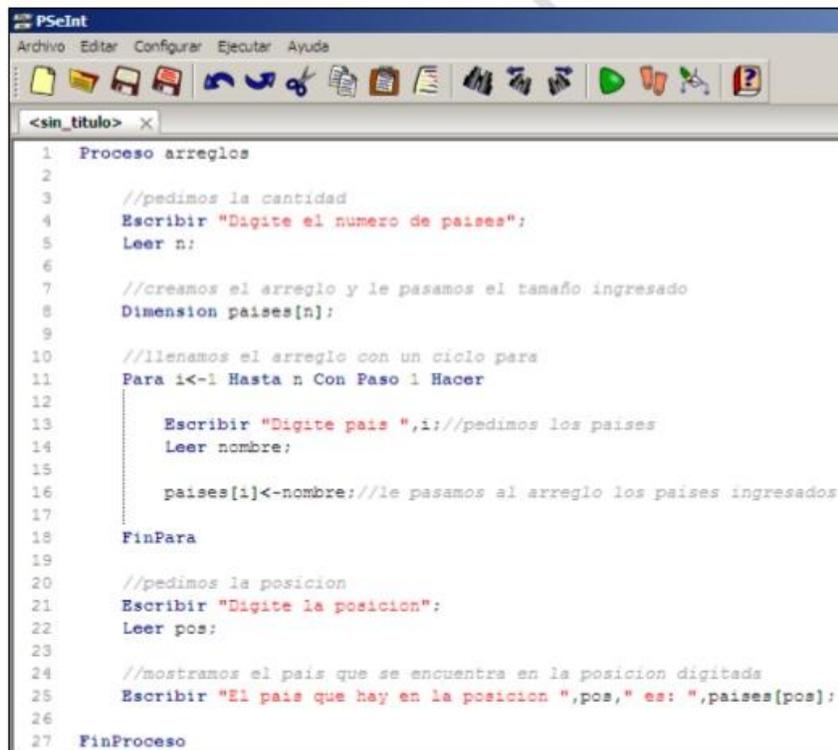
```
1 Proceso arreglos
2
3     cont<-0;//variable que cuenta los positivos inicializada en cero
4
5     //pedimos el tamaño
6     Escribir "Digite el numero de posiciones (tamaño)";
7     Leer p;
8
9     //creamos el arreglo y le pasamos el tamaño ingresado
10    Dimension num[p];
11
12    //llenamos el arreglo con un ciclo para
13    Para i<-1 Hasta p Con Paso 1 Hacer
14
15        Escribir "Digite numero ",i;//pedimos los numeros
16        Leer numero;
17
18        num[i]<-numero;//le pasamos al arreglo los numeros ingresados
19
20    FinPara
21
22    //imprimimos los datos con un ciclo para
23    Para i<-1 Hasta p Con Paso 1 Hacer
24
25        Escribir "Los numeros ingresados son: ",num[i];//mostramos los numeros ingresados
26
27        si (num[i]>0) Entonces//si numero ingresado mayor a cero entonces es positivo
28
29            cont<-cont+1;//contamos los numeros que sean positivos
30
31        FinSi
32
33    FinPara
34
35    Escribir "Hay ",cont," numeros positivos");//mostramos cuantos numeros positivos hay
36
37 FinProceso
```

Salida:



```
C:\Archivos de programa\PSeInt\pseint.exe
*** Ejecucion Iniciada. ***
Digite el numero de posiciones (tamano)
4
Digite numero 1
-2
Digite numero 2
45
Digite numero 3
5
Digite numero 4
-1
Los numeros ingresados son: -2
Los numeros ingresados son: 45
Los numeros ingresados son: 5
Los numeros ingresados son: -1
Hay 2 numeros positivos
*** Ejecucion Finalizada. ***
```

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



```
PSeInt
Archivo Editar Configurar Ejecutar Ayuda
<sin_titulo> x
1 Proceso arreglos
2
3 //pedimos la cantidad
4 Escribir "Digite el numero de paises":
5 Leer n:
6
7 //creamos el arreglo y le pasamos el tamaño ingresado
8 Dimension paises[n]:
9
10 //llenamos el arreglo con un ciclo para
11 Para i<-1 Hasta n Con Paso 1 Hacer
12
13     Escribir "Digite pais ",i;//pedimos los paises
14     Leer nombre:
15
16     paises[i]<-nombre;//le pasamos al arreglo los paises ingresados
17
18 FinPara
19
20 //pedimos la posicion
21 Escribir "Digite la posicion":
22 Leer pos:
23
24 //mostramos el pais que se encuentra en la posicion digitada
25 Escribir "El pais que hay en la posicion ",pos," es: ",paises[pos]:
26
27 FinProceso
```

Salida:

```
C:\Archivos de programa\PSeInt\pseint.exe
*** Ejecucion Iniciada. ***
Digite el numero de paises
5
Digite pais 1
Venezuela
Digite pais 2
Colombia
Digite pais 3
Ecuador
Digite pais 4
Peru
Digite pais 5
Bolivia
Digite la posicion
2
El pais que hay en la posicion 2 es: Colombia
*** Ejecucion Finalizada. ***
```

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.

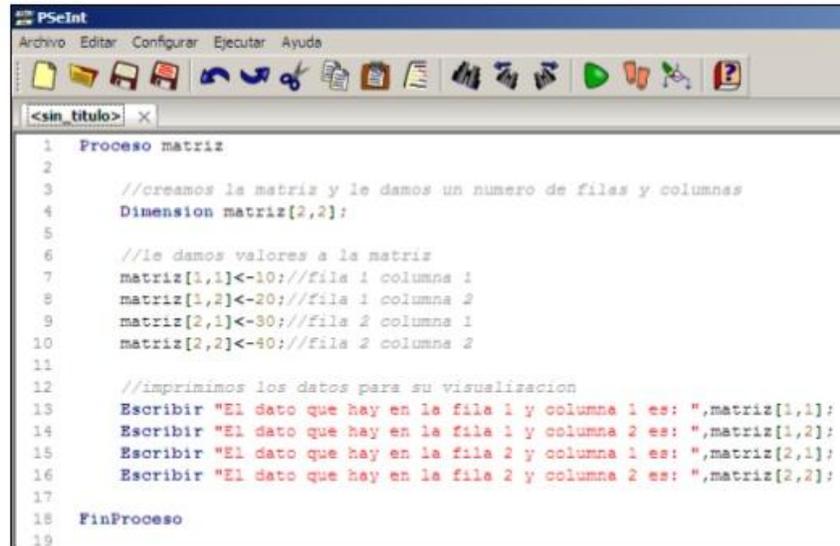
Sintaxis:

Dimension identificador [filas,columnas];

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

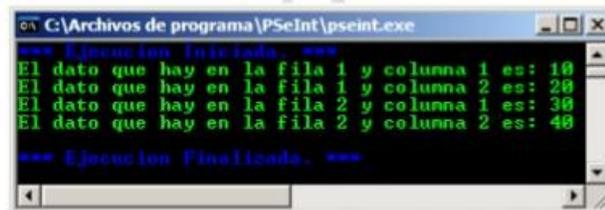
Ejemplo 1:

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



```
1  Proceso matriz
2
3      //creamos la matriz y le damos un numero de filas y columnas
4      Dimension matriz[2,2]:
5
6      //le damos valores a la matriz
7      matriz[1,1]<-10;//fila 1 columna 1
8      matriz[1,2]<-20;//fila 1 columna 2
9      matriz[2,1]<-30;//fila 2 columna 1
10     matriz[2,2]<-40;//fila 2 columna 2
11
12     //imprimimos los datos para su visualizacion
13     Escribir "El dato que hay en la fila 1 y columna 1 es: ",matriz[1,1];
14     Escribir "El dato que hay en la fila 1 y columna 2 es: ",matriz[1,2];
15     Escribir "El dato que hay en la fila 2 y columna 1 es: ",matriz[2,1];
16     Escribir "El dato que hay en la fila 2 y columna 2 es: ",matriz[2,2];
17
18     FinProceso
19
```

Salida:



```
C:\Archivos de programa\PSeInt\pseint.exe
*** Ejecucion Iniciada. ***
El dato que hay en la fila 1 y columna 1 es: 10
El dato que hay en la fila 1 y columna 2 es: 20
El dato que hay en la fila 2 y columna 1 es: 30
El dato que hay en la fila 2 y columna 2 es: 40
*** Ejecucion Finalizada. ***
```

Representación grafica del anterior ejemplo:

matriz

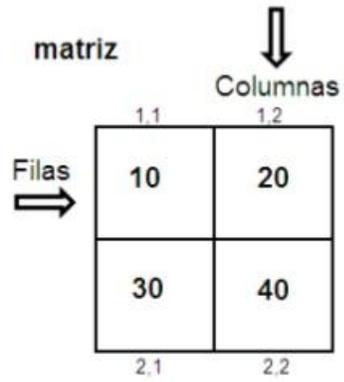
↓
Columnas

1.1 1.2

⇒ Filas

10	20
30	40

2.1 2.2

A 2x2 matrix is shown with a grid. The top-left cell contains '10', the top-right '20', the bottom-left '30', and the bottom-right '40'. Above the grid, the word 'matriz' is written. To its right, a downward-pointing arrow is labeled 'Columnas'. Below the grid, the word 'Filas' is written with a rightward-pointing arrow. The four cells of the grid are labeled with their respective row and column indices: '1.1' above the top-left, '1.2' above the top-right, '2.1' below the bottom-left, and '2.2' below the bottom-right.

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**

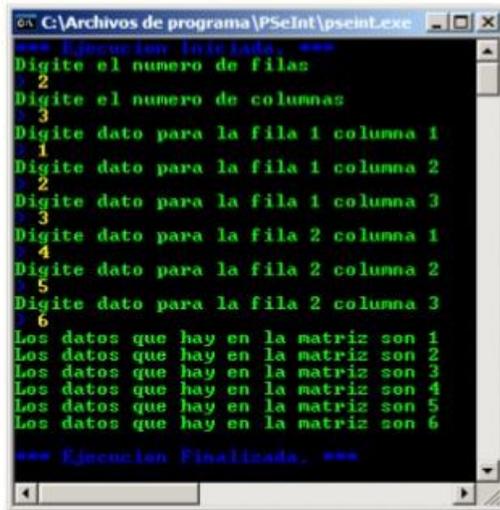
Así como en los arreglos unidimensionales llenábamos el vector con un ciclo **Para**, en las matrices también lo podemos hacer, sólo que ya no se utilizará un ciclo sino **dos**, uno para las filas y otro para las columnas. También los datos se muestran con dos ciclos.

Ejemplo 2:

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

```
PSInt
Archivo Editar Configurar Ejecutar Ayuda
matriz.psc x
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         //pedimos los datos
18         Escribir "Digite dato para la fila ",i," columna ",j:
19         Leer numero:
20         //llenamos la matriz con los numeros ingresados
21         matriz[i,j]<-numero:
22     FinPara
23 FinPara
24
25 //mostramos todos los datos que hay en la matriz con dos ciclos PARA
26 Para i<-1 Hasta f Con Paso 1 Hacer
27     Para j<-1 Hasta c Con Paso 1 Hacer
28         Escribir "Los datos quehay en la matriz son ",matriz[i,j]:
29     FinPara
30 FinPara
31
32 FinProceso
```

Salida:



```
C:\Archivos de programa\PSeInt\pseint.exe
*** Ejecucion Inicializada ***
Digite el numero de filas
2
Digite el numero de columnas
3
Digite dato para la fila 1 columna 1
1
Digite dato para la fila 1 columna 2
2
Digite dato para la fila 1 columna 3
3
Digite dato para la fila 2 columna 1
4
Digite dato para la fila 2 columna 2
5
Digite dato para la fila 2 columna 3
6
Los datos que hay en la matriz son 1
Los datos que hay en la matriz son 2
Los datos que hay en la matriz son 3
Los datos que hay en la matriz son 4
Los datos que hay en la matriz son 5
Los datos que hay en la matriz son 6
*** Ejecucion Finalizada ***
```

EJERCICIO CON MATRIZ

1. Crear una matriz $n \times n$ y llenarla con los números que el usuario desee. Suma todos los números que componga la columna 1.

```
Psiclat
Archivo Editar Configuración Ejecutar Ayuda
matriz.psc x
1 Proceso arreglos
2
3 acum<-0;//acumulador inicializado en cero para sumar los elementos de la columna 1
4
5 Escribir "Digite el numero de filas";//pedimos el numero de filas
6 Leer f:
7
8 Escribir "Digite el numero de columnas";//pedimos el numero de columnas
9 Leer c:
10
11 Dimension matriz[f,c];//creamos la matriz y le pasamos el numero de filas y columnas ingresadas
12
13 //llenamos la matriz con dos ciclos PARA, uno para las filas y otro para las columnas
14 Para i<-1 Hasta f Con Paso 1 Hacer
15     Para j<-1 Hasta c Con Paso 1 Hacer
16     |
17     |   Escribir "Digite dato para la fila ",i," columna ",j;//pedimos los datos
18     |   Leer numero:
19     |   |
20     |   |   matriz[i,j]<-numero;//llenamos la matriz con los numeros ingresados
21     |   |
22     |   |   FinPara
23     |   |
24     |   |   acum<-acum+matriz[i,1];//acumulamos y sumamos todos los numeros que componen la columna 1
25     |   |
26     |   |   FinPara
27     |   |
28     |   |   //mostramos todos los datos que hay en la matriz con dos ciclos PARA
29     |   |   Para i<-1 Hasta f Con Paso 1 Hacer
30     |   |   |   Para j<-1 Hasta c Con Paso 1 Hacer
31     |   |   |   |
32     |   |   |   |   Escribir matriz[i,j]:
33     |   |   |   |
34     |   |   |   |   FinPara
35     |   |   |   |
36     |   |   |   |   FinPara
37     |   |   |   |
38     |   |   |   |   Escribir "Todos los elementos de la columna 1 suman un total de: ",acum;//mostramos la suma
39     |   |   |   |
40     |   |   |   |   FinProceso
```

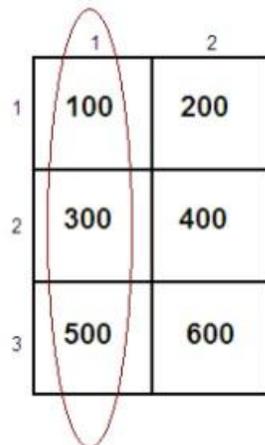
Salida:

```
C:\Archivos de programa\PScInt\pseint.exe
*** Ejecucion Iniciada. ***
Digite el numero de filas
3
Digite el numero de columnas
2
Digite dato para la fila 1 columna 1
100
Digite dato para la fila 1 columna 2
200
Digite dato para la fila 2 columna 1
300
Digite dato para la fila 2 columna 2
400
Digite dato para la fila 3 columna 1
500
Digite dato para la fila 3 columna 2
600
100
200
300
400
500
600
Todos los elementos de la columna 1 suman un total de: 900
*** Ejecucion Finalizada. ***
```

Representación grafica del anterior ejercicio:

matriz

	1	2
1	100	200
2	300	400
3	500	600



Total: 900

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

```
PSInt
Archivo  Editar  Configurar  Ejecutar  Ayuda
matriz.psc x
1  Proceso arreglos
2
3      //cremos 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     FinPara
15
16     //mostramos todos los datos que hay en la matriz
17     Para i<-1 Hasta 3 Con Paso 1 Hacer
18
19         Escribir matriz[i,1], " ",matriz[i,2], " ",matriz[i,3];
20
21     FinPara
22
23
24 FinProceso
```

Salida:

```
C:\Archivos de programa\PSInt...
*** Ejecucion Iniciada. ***
3 1 0
1 9 2
4 6 6
*** Ejecucion Finalizada. ***
```