Tema: “Vectores en C#”

Objetivos

- Introducir al estudiante a los conceptos de estructura de datos.
- Conocer la sintaxis de creación y definición de vectores.
- Conocer los tipos de vectores que se utilizan en la programación.
- Resolver problemas aplicando vectores.

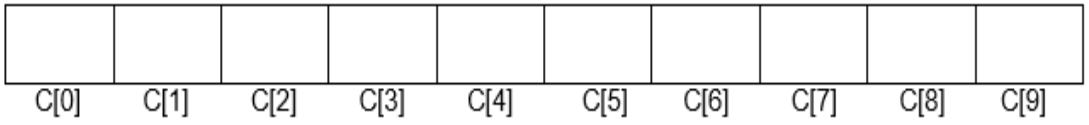
Introducción

Los arreglos son estructuras de datos que consisten en elementos de datos del mismo tipo relacionados. Los arreglos son entidades de longitud fija; conservan la misma longitud una vez que se crean, aunque puede reasignarse una variable tipo arreglo de tal forma que haga referencia a un nuevo arreglo de distinta longitud.

Arreglos

Un arreglo es un grupo de variables (llamadas elementos) que contienen valores y todos son del mismo tipo. Recordemos que los tipos se dividen en dos categorías: por valor y por referencia.

Los elementos de un arreglo pueden ser tipos por valor o referencia. Para referirnos a un elemento en especial dentro de un arreglo, especificamos el nombre de la referencia al arreglo y el número de la posición de este elemento en el arreglo. Al número de la posición se le conoce como índice del elemento. Una aplicación hace referencia a cualquier elemento del arreglo mediante una expresión de acceso al arreglo, la cual incluye el nombre del arreglo, seguido el índice del elemento específico entre corchetes ([ ]). El primer elemento en cualquier arreglo tiene el índice cero, el cual se conoce como elemento cero. En la figura siguiente, se muestra una ilustración de un arreglo:



- El nombre del arreglo anterior es C.
- El índice del elemento en el arreglo C se denota por: C[0], C[1], …, C[9].
- Si un arreglo tiene n posiciones, la última posición tiene el índice o subíndice n-1.

Las instancias de los arreglos ocupan espacio en memoria. Al igual que los objetos, los arreglos se crean con la palabra clave new. Para crear una instancia de un arreglo, se especifica el tipo y el número de elementos del arreglo, y el número de elementos como parte de una expresión de creación de arreglos, que utiliza la palabra clave new. Dicha expresión devuelve una referencia que puede almacenarse en una variable tipo arreglo. La siguiente expresión de declaración y creación de arreglos crea un objeto que contiene 12 elementos int, y almacena la referencia al arreglo en la variable C.

```
int[ ] C = new int[12];
```

Cabe mencionar, que la expresión anterior puede dividirse en dos partes, así:

```
int [ ] C;
C = new int[12];
```

Material y Equipo

- Guía de laboratorio No. 9.
- Computadora con Visual Studio 2013 o superior.
- Dispositivo de almacenamiento (USB).

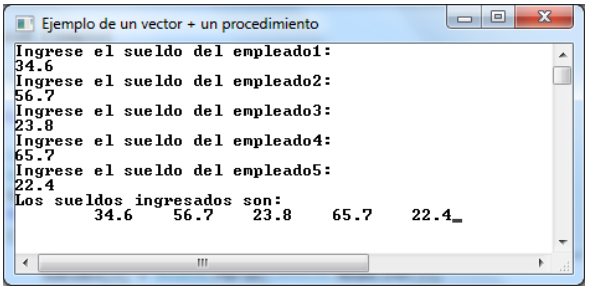
Procedimiento

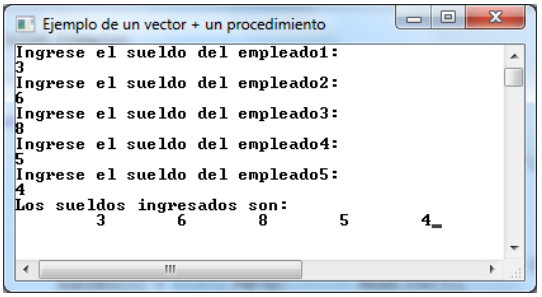
Ejemplo1

Se desea guardar los sueldos de 5 empleados de una fábrica X. Según lo que hemos estudiado hasta el momento, deberíamos definir 5 variables si queremos tener en un cierto momento los 5 sueldos almacenados en memoria. Empleando un vector, solo se requiere definir un único nombre y accedemos a cada elemento por medio del subíndice.

```
1   {
2     Double[] Sueldos = new Double[5];
3     for (int i = 0; i < 5; i++)
4     {
5     Console.WriteLine("Ingrese el sueldo del empleado" + (i + 1) + ":");
6     Sueldos[i] = Double.Parse(Console.ReadLine());
7     }
8     Mostrar(Sueldos); // utilizando un procedimiento
9     Console.ReadKey();
10  }
11  static void Mostrar (Double[] Sueldos)
12     {
13        Console.WriteLine("Los sueldos ingresados son:");
14        for (int i=0;i<5;i++)
15          {
16              Console.Write("\t" + Sueldos[i]);
```

```
17          }
18      }
```

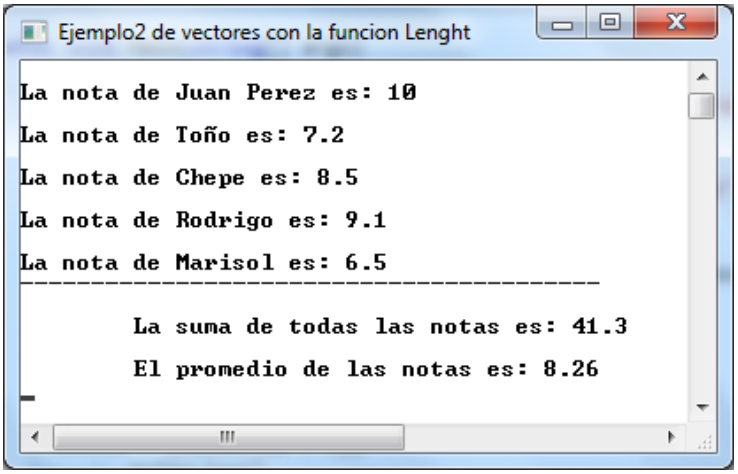




Ejemplo2

En este ejemplo veremos cómo trabajar con un arreglo unidimensional que ha sido previamente inicializado. Vamos a conocer la forma de especificar los elementos que contiene la estructura de datos.

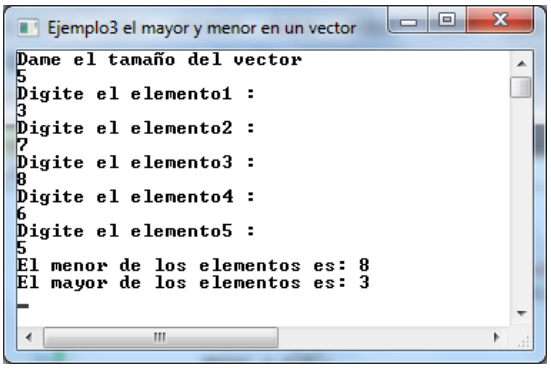
```
1    {
2     Console.Title = "Ejemplo2 de vectores con la funcion Lenght";
3     Double SumaNotas = 0, Promedio;
4     Double[] Notas = { 10, 7.2, 8.5, 9.1, 6.5 };
5     String[] Alumnos = {"Juan Perez","Toño","Chepe","Rodrigo", "Marisol"};
6     for ( int i=0; i<Notas.Length; i++)
7       {
8            Console.WriteLine("\nLa nota de {0} es: {1}", Alumnos[i], Notas[i]);
9       }
10    for ( int i=0; i<Notas.Length; i++)
11      {
12          SumaNotas = SumaNotas + Notas[i];
13      }
14    Promedio = (SumaNotas / 5);
15    Console.WriteLine("----------------------------------------");
16    Console.WriteLine("\n\tLa suma de todas las notas es: " + SumaNotas);
17    Console.WriteLine("\n\tEl promedio de las notas es: " + Math.Round(Promedio,2));
18    Console.ReadKey();
19   }
20
```

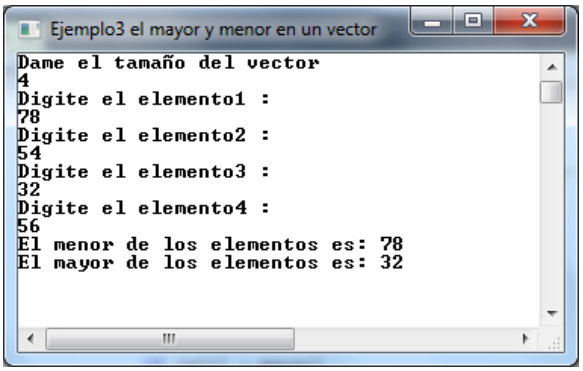


Ejemplo3

En este ejemplo, aprenderemos a encontrar el mayor elemento de un arreglo así como también el menor elemento del mismo.

```
{
 Console.Title = "Ejemplo3 el mayor y menor en un vector";
 int[] n;
 int tam, menor, mayor;
 Console.WriteLine("Dame el tamaño del vector");
 tam = int.Parse(Console.ReadLine());
 n = new int[tam];
 for (int i = 0; i < tam; i++)
  {
     Console.WriteLine("Digite el elemento" + (i+1) + " : ");
     n[i] = int.Parse(Console.ReadLine());
  }
 mayor = n[0];
 menor = n[0];
 for (int i = 0; i < tam; i++)
 {
      if (n[i] > mayor)
       {
         mayor = n[i];
        }
  else if (n[i] < menor)
       {
          menor = n[i];
        }
 }
 Console.WriteLine("El menor de los elementos es: " + mayor );
 Console.WriteLine("El mayor de los elementos es: " + menor);
 Console.ReadKey();
}
```





La instrucción foreach

En ejemplos anteriores, demostramos cómo utilizar las instrucciones for controladas por un contador para iterar a través de los elementos de un arreglo.

Ahora comenzamos el estudio del foreach, que itera a través de los elementos de un arreglo o colección completa.

La sintaxis de una instrucción foreach es la siguiente:

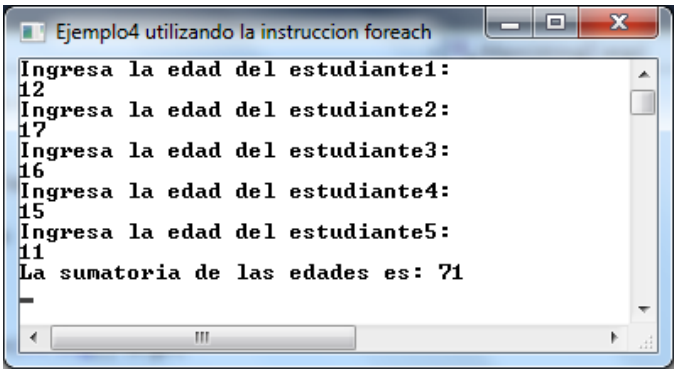
```
foreach (<tipoElemento> <Elemento> in <NombreArreglo>)
{
<Instrucciones>
}
```

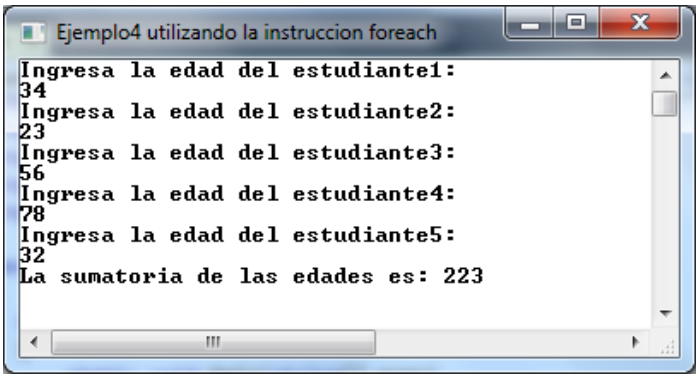
Donde, el tipo y el identificador son el tipo y el nombre (por ejemplo int numero) de la variable de iteración y NombreArreglo es al arreglo a través del cual se va a iterar. El tipo de la variable de iteración debe concordar con el tipo de los elementos del arreglo.

La instrucción foreach es una variante de la instrucción for, pensada principalmente para compactar la escritura de códigos donde se realiza un tratamiento a todos los elementos de un arreglo o lista.

En este ejemplo, haremos uso de la instrucción foreach para realizar una sumatoria de edades de estudiantes. Se crea un arreglo de 5 posiciones de memoria para almacenar datos de tipo entero.

```
1     {
2     Console.Title = "Ejemplo4 utilizando la instruccion foreach";
3     int[] Edades = new int[5];
4     Array(Edades);
5     Console.ReadKey();
6     }
7     static void Array(int[] Edades)
8     {
9       int Total = 0;
10      Edades = new int[5];
11      for (int x = 0; x < Edades.Length; x++)
12      {
13         Console.WriteLine("Ingresa la edad del estudiante" + (x + 1) + ":");
14         Edades[x] = int.Parse(Console.ReadLine());
15      }
16      foreach (int element in Edades)
17      {
18         Total = Total + element;
19      }
20     Console.WriteLine("La sumatoria de las edades es: " + Total);
21     }
22
```





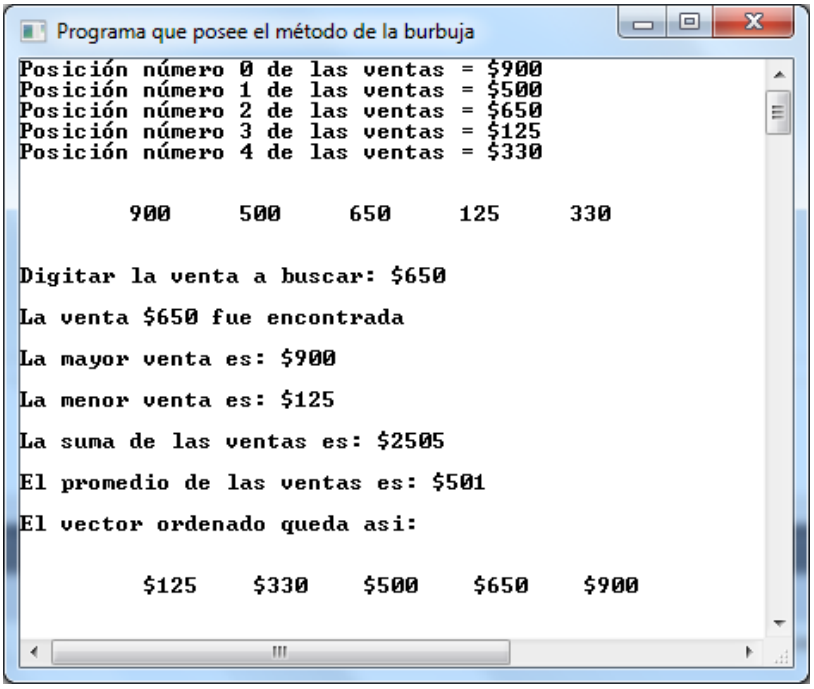
Ejemplo5

A continuación presentamos varias operaciones que podemos realizar con un vector de datos numéricos, como lo es: llenado de datos, buscar datos, mayor y menor, sumatoria y promedio y sin olvidar lo más importante es utilizar el método de la burbuja para el ordenamiento de los datos.

```
1    static void Main(string[] args)
2     {
3      Console.ForegroundColor = ConsoleColor.Black;
4      Console.BackgroundColor = ConsoleColor.White;
5      Console.Clear();
6      Console.Title = "Programa que posee el método de la burbuja";
7      Double[] ventas = new Double[5];
8      int i,j;
9      Double resp, encontrado, promedio, suma, buscar, mayor, menor, aux;
10     //LLENADO DE VENTAS
11     for (i=0;i<5;i++)
12     {
13      Console.Write("Posición número " + i +" de las ventas = $");
14      ventas[i] = Double.Parse(Console.ReadLine());
15     }
16     Console.WriteLine("\n");
17     //IMPRIME LOS DATOS AÑADIDOS
18     for(i=0;i<5;i++)
19     {
20     Console.Write("\t" + ventas[i]);
```

```
 }
Console.WriteLine("\n");
//BUSCAR UNA VENTA
encontrado = 1;
resp = 0;
Console.Write("\nDigitar la venta a buscar: $");
buscar = Double.Parse(Console.ReadLine());
for(i=0;i<5;i++)
{
 if (buscar == ventas[i])
  resp = 1;
}
 if (encontrado == resp)
 {
 Console.WriteLine("\nLa venta ${0} fue encontrada", buscar);
 }
 else
 {
  Console.WriteLine("\nLa venta ${0} no fue encontrada", buscar);
 }
//MOSTRAR LA MAYOR Y LA MENOR VENTA
mayor=0;
menor=1000;
for(i=0;i<5;i++)
{
 if (mayor < ventas[i])
 mayor = ventas[i];
 if (menor > ventas[i])
 menor = ventas[i];
}
Console.WriteLine("\nLa mayor venta es: $" + mayor);
Console.WriteLine("\nLa menor venta es: $" + menor);
//LA SUMATORIA DE LAS VENTAS
suma = 0;
for(i=0;i<5;i++)
{
 suma = suma + ventas[i];
}
Console.WriteLine("\nLa suma de las ventas es: $" + suma);
promedio=suma/5;
Console.WriteLine("\nEl promedio de las ventas es: $" + Math.Round(promedio,3));
//REALIZA LA ORDENACION POR MEDIO DE LA BURBUJA
for(i=0;i<5-1;i++)
 {//inicio de primer for
for(j=0;j<5-1;j++)
 { //inicio del segundo for
  if(ventas[j]>ventas[j+1])
  { // inicio del if
   aux=ventas[j];
   ventas[j]=ventas[j+1];
```

```
71        ventas[j+1]=aux;
72       }// fin del if
73      }//fin del segundo for, el más interno, el de la j
74     }//fin del primer for
75     //IMPRIME LAS VENTAS ORDENADAS
76     Console.WriteLine("\nEl vector ordenado queda asi:");
77     Console.WriteLine("\n");
78     for(i=0;i<5;i++)
79     {
80      Console.Write("\t $" + ventas[i]);
81     }
82     Console.WriteLine("\n");
83     Console.ReadKey();
84    }
```

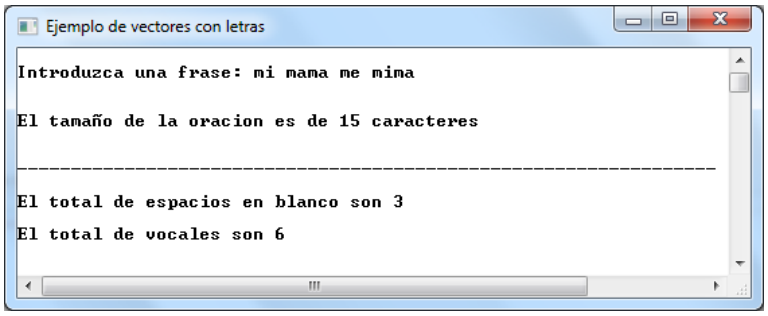


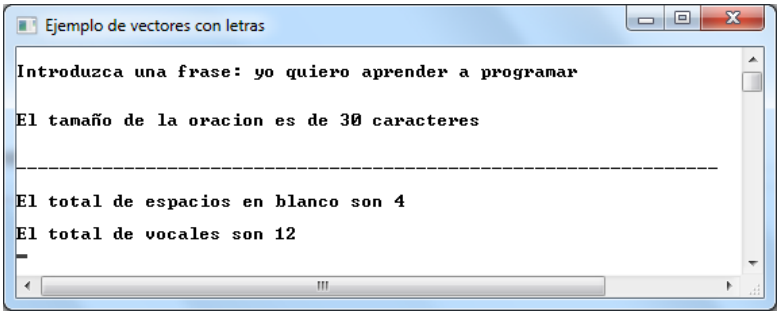
Ejemplo6

Programa que permite digitar una oración y me muestra cuántas vocales y espacios en blanco posee.

```
1     static void Main(string[] args)
2      {
3       Console.ForegroundColor = ConsoleColor.Black;
4       Console.BackgroundColor = ConsoleColor.White;
5       Console.Clear();
6       Console.Title = "Ejemplo de vectores con letras";
7       //Escriba un programa que lea una cadena de texto  y como resultado que muestre el
      total
8       // de vocales y espacios en blanco que tiene la frase.
9       String nombre;
10      char caracter;
```

```
11      int i, vocales, espacios;
12      vocales = 0;
13      espacios = 0;
14      Console.Write("\nIntroduzca una frase: ");
15      nombre = Console.ReadLine();
16      Console.WriteLine("\n");
17      Console.WriteLine("El tamaño de la oracion es de " + nombre.Length + " caracteres");
18      Console.WriteLine("\n");
19      Console.WriteLine("------------------------------------------------------------");
20      for (i = 0; i < nombre.Length; i++)
21      {
22       caracter = char.Parse(nombre.Substring(i, 1));
23       //búsqueda de espacios en blanco
24       if (caracter == ' ')
25       // al poner un espacio en blanco entre apostrofe indicamos que es espacio en blanco
26         espacios++;
27           //busqueda de vocales
28         if ((caracter == 'A') || (caracter == 'a'))
29             vocales++;
30         if ((caracter == 'E') || (caracter == 'e'))
31             vocales++;
32         if ((caracter == 'I') || (caracter == 'i'))
33             vocales++;
34         if ((caracter == 'O') || (caracter == 'o'))
35             vocales++;
36         if ((caracter == 'U') || (caracter == 'u'))
37             vocales++;
38      }
39      Console.WriteLine("\nEl total de espacios en blanco son " + espacios);
40      Console.WriteLine("\nEl total de vocales son " + vocales);
41      Console.ReadKey();
42     }
```

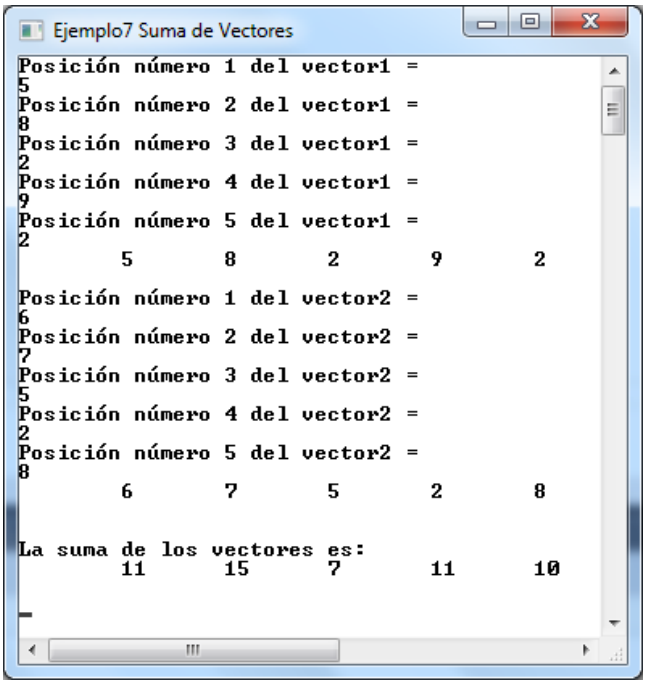




Ejemplo7

Programa que permite digitar los datos de 2 vectores y realiza la suma de ellos.

```
1    static void Main(string[] args)
2    {
3    Console.ForegroundColor = ConsoleColor.Black;
4    Console.BackgroundColor = ConsoleColor.White;
5    Console.Clear();
6    Console.Title = "Ejemplo7 Suma de Vectores";
7    //Programa que suma los elementos de 2 vectores
8    Double[] vector1 = new Double[5];
9    Double[] vector2 = new Double[5];
10   Double[] suma = new Double[5];
11   int i;
12   //LLENADO DEL VECTOR1
13      for (i = 0; i < 5; i++)
14      {
15      Console.WriteLine("Posición número " + (i+1) + " del vector1 = ");
16      vector1[i] = Double.Parse(Console.ReadLine());
17      }
18      for (i = 0; i < 5; i++)
19      {
20      Console.Write("\t" + vector1[i]);
21      }
22      Console.WriteLine("\n");
23   //LLENADO DEL VECTOR2
24      for (i = 0; i < 5; i++)
25      {
26      Console.WriteLine("Posición número " + (i+1) + " del vector2 = ");
27      vector2[i] = Double.Parse(Console.ReadLine());
28      }
29      for (i = 0; i < 5; i++)
30      {
31      Console.Write("\t" + vector2[i]);
32      }
33      Console.WriteLine("\n");
34   //REALIZA LA SUMA DE LOS VECTORES E IMPRIME EL RESULTADO
35      for (i = 0; i < 5; i++)
36      {
37      suma[i] = vector1[i] + vector2[i];
38      }
39      Console.WriteLine("\nLa suma de los vectores es:");
40      for (i = 0; i < 5; i++)
41      {
42      Console.Write("\t" + suma[i]);
43      }
44   Console.WriteLine("\n");
45   Console.ReadKey();
46   }
```



Análisis de Resultados

1. De una lista de 9 notas almacenadas en un vector llamado Grado, se necesita saber cuántas notas son igual a 8.5
2. Una planta que fabrica perfiles de hierro posee un lote de “n” piezas. Diseñar un programa que pida ingresar la cantidad de piezas a procesar y luego ingresar la longitud de cada perfil de hierro; sabiendo que la pieza cuya longitud este comprendida en el rango de 1.20 y 1.40 son aptas. Imprimir por pantalla la cantidad de piezas aptas que hay en el lote.
3. Cargar un vector y solicitarle al usuario un valor que quiera buscar en un arreglo de una dimensión, luego el programa debe decirle si se encuentra o no.
   Si se encuentra, debe mostrarse la posición del elemento.
4. Realizar un programa para la empresa “T-Comunico” que solicite las ventas de recargas mensuales de la tienda (durante 9 meses), luego muestre el mes en el que se realizo la venta mayor y la venta menor, además del promedio y los meses que tienen ventas menores al promedio.
5. En una empresa trabajan “x” empleados cuyos sueldos oscilan entre $100 y $500, realizar un programa que lea los sueldos que cobra cada empleado e informe cuantos empleados cobran entre $100 y $300 y cuantos cobran mas de $350.
6. Desarrollar un programa que permita cargar 5 nombres de personas y sus edades respectivas. Luego de digitar la información de todas las personas, se necesita imprimir los nombres de las personas mayores de edad (mayores o iguales a 18).

Investigación Complementaria

1. Desarrollar una aplicación que ordene los elementos de un vector de 7 números enteros (ascendente y descendente).
2. Se dispone de una lista de las temperaturas de una ciudad en cada uno de los días de una determinada semana.
   Se desea calcular la temperatura mayor y menor de la semana.
3. De una lista de 6 precios almacenados en un vector llamado **Mercadito**, se necesita saber cuántos precios son mayores a $25.
4. Definir un vector de 5 componentes de tipo Double que representen las alturas de 6 personas. Obtener el promedio de las mismas. Contar cuantas personas son mas altas que el promedio y cuantas mas bajas.
5. Una empresa tiene dos turnos (mañana y tarde) en los que trabajan 8 empleados (4 por la mañana y 4 por la tarde). Diseñar un programa que permita almacenar los sueldos de los empelados agrupados por turno. Imprimir el total de sueldos por turno.
6. Se tienen las notas del primer parcial de los alumnos de dos cursos, el curso A y el curso B, cada curso cuenta con 5 alumnos. Realizar un programa que muestre el curso que obtuvo el mayor promedio general.

Bibliografía

- Deitel, Harvey M. y Paul J. Deitel, Cómo Programar en C#, Segunda Edición, México, 2007.