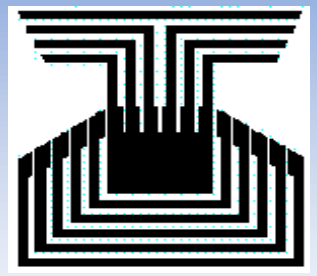


UNIVERSIDAD NACIONAL EXPERIMENTAL DEL TÁCHIRA
VICERRECTORADO ACADÉMICO
DECANATO DE DOCENCIA
DEPARTAMENTO DE ING. EN INFORMÁTICA

# UNIDAD CURRICULAR: COMPUTACIÓN I

(Código: 0415102T)

## Clase No: 7

## Unidad IV: Arreglos en Lenguaje C (Unidimensionales y Bidimensionales)

Profesor: Armando Carrero

# UNIDAD No. IV

# ARREGLOS EN LENGUAJE C

## ¿Qué es un Arreglo?

Es una variable, con las siguientes características:

1.- Permite almacenar varios valores homogéneos, es decir, del mismo tipo.

2.- A la variable se le asigna un único nombre.

3.- Los valores se almacenan en la memoria interna (RAM).

4.- Los valores se pueden acceder en forma directa o secuencial.

5.- Los valores se almacenan en forma contigua.

*La primera característica es una ventaja, por cuanto las variables simples, solo permiten almacenar un único valor; sin embargo deben ser del mismo tipo, limitación que será superada con otro tipo de variable que se estudiaran en temas posteriores, de esta unidad curricular.*

*La segunda característica es una ventaja, por cuanto con una única variable se puede acceder a varios valores.*

*La tercera característica es una desventaja, por cuanto se sabe que la memoria RAM es volátil, es decir, los valores se pierden a l suspenderse la energía eléctrica, esta desventaja se subsana mediante el uso de otra forma de almacenar los datos, que se estudian en unidades curriculares posteriores.*

*La cuarta y quinta características son una ventaja, que permite acceder a los datos almacenados, de una forma fácil y favorable.*

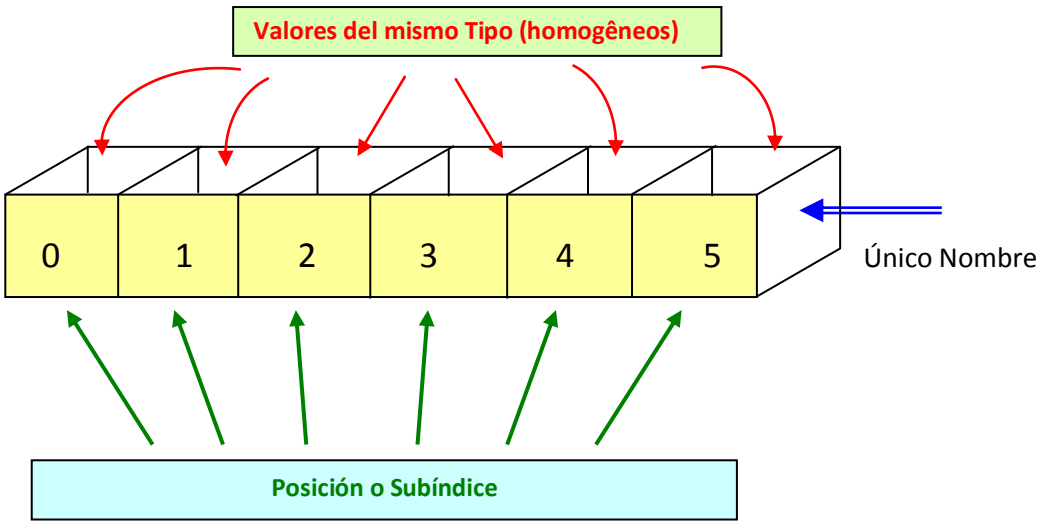
## Tipos de Arreglos:

Tipos : { Arreglos Unidimensionales, Arreglos Bidimensionales, Arreglos Multidimensionales

# Arreglos Unidimensionales

También se conoce como **Lista** o **Vector.**

Representación Grafica de un Arreglo Unidimensional




## DECLARACIÓN DE UNA VARIABLE TIPO ARREGLO UNIDIMENSIONAL

Formato para declarar arreglos unidimensionales (Vectores) de:
Enteros, Reales ó Caracteres.

tipo de dato nombre de la variable [ Dimensión, Tamaño ó Longitud del Arreglo ] ;

La dimensión, tamaño o longitud, se refiere a la cantidad máxima de valores que se pueden almacenar en el arreglo

Para declarar un arreglo (vector) de cadena de caracteres:

char nombre de la variable [ Cantidad de Cadenas ] [ Cantidad Máx. caracteres por cadena + 1 ] ;

Ejemplo: Declarar variables de tipo arreglo unidimensional para almacenar:

El código (*cadena*), cantidad (*entero*), peso (*real*) y calidad (*caracter*) de 50 diferentes herramientas usadas en la construcción civil.

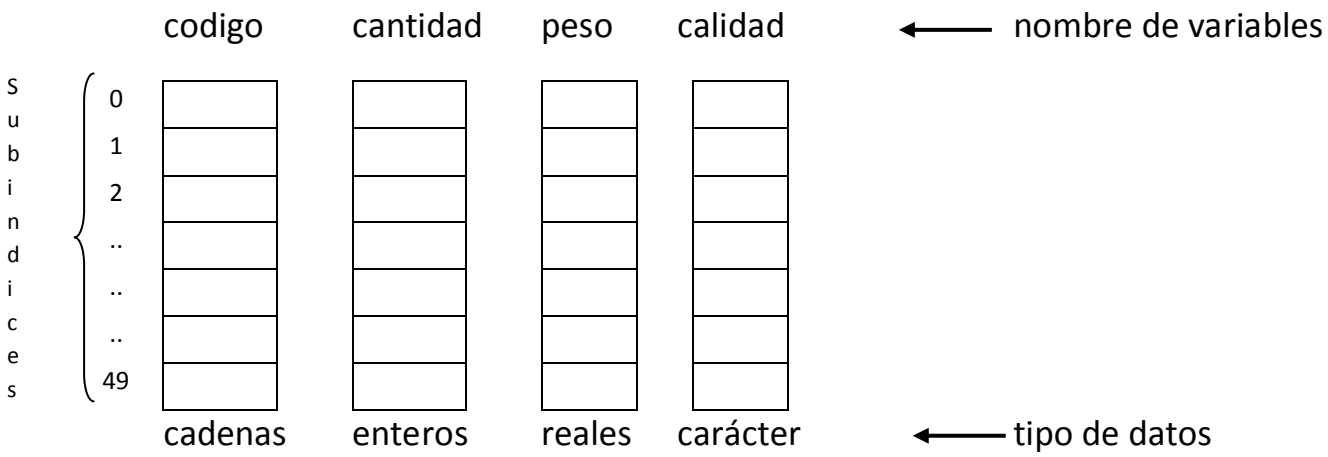
```
char  codigo [ 50  ][ 25  ] ;
int   cantidad [ 50 ] ;
float peso [ 50 ] ;
char  calidad [  51  ] ;
```

Observaciones:

La declaración de la variable codigo, indica que puede almacenar 50 cadenas, cada una con un máximo de 24 caracteres ya que el último (el numero 25) se reserva para que el compilador coloque el carácter nulo "\0", que se necesita para indicar el fin de la cadena.

La variable calidad, es declarada con una dimensión de 51, debido a que un vector de caracteres equivale a una cadena de caracteres, por lo tanto debe tener reservado una última posición para la ubicación del caracter nulo.

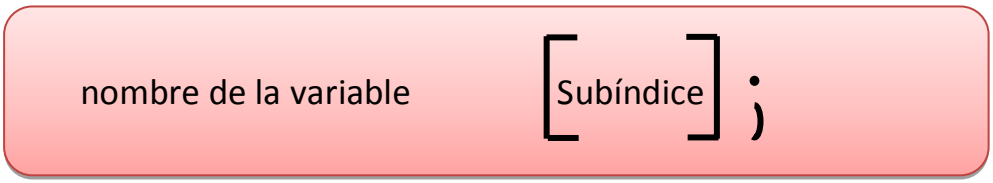
Representación grafica de las variables (arreglos) declaradas:




Nota: En lenguaje C, el primer subíndice de un arreglo es cero (0), y en éste caso el último es 49, ya que en la declaración, el tamaño de los arreglos es 50. Si se empieza con 1, la primera posición del arreglo queda vacía y es un error referirse a la posición 50, ya que ésta no existe.

## ACCESO A UNA VARIABLE TIPO ARREGLO UNIDIMENSIONAL

Para acceder a una variable de tipo vector, debe usarse la siguiente forma:




Ejemplo. Para leer los valores de la primera posición de las variables declaradas, se hace de la siguiente forma:

```
printf ( " Ingrese el código de la herramienta");
scanf("%s", código [ 0 ]) ;
printf ( " Ingrese la cantidad en existencia de la herramienta");
scanf("%d",  &cantidad [ 0 ]) ;
printf ( " Ingrese el peso de la herramienta");
scanf("%f",  &peso [ 0 ]) ;
printf ( " Ingrese la calidad de la herramienta");
scanf("%c", &calidad [ 0 ]) ;
```

Observación: Al leer las variables, en lenguaje C, debe anteponerse el símbolo "&"(*ampersan*), previo al nombre de la variable, excepto para las cadenas. Para leer cadenas se recomienda usar la función gets, en vez de scanf.

Las lecturas de los datos anteriores se pueden hacer simultáneamente, excepto las cadenas de caracteres que se recomienda hacerla por separado, así:

```
printf ( " \n Ingrese el código de la herramienta :  ");
gets( codigo [ 0 ]) ;
printf ( " \n Ingrese la cantidad, el peso y la calidad de la herramienta");
scanf("%d%f%c",  &cantidad [ 0 ] ,   &peso [ 0 ], &calidad [ 0 ]) ;
```

Con las sentencias anteriores, se almacenan los datos en la primera posición de los arreglos. Es de preguntarse:  y para las restantes 49 herramientas ¿Hay que repetir el proceso anterior, usando los subíndices 1, 2, 3, 4, 5……49? , la respuesta es: se puede hacer de esa forma pero el número de sentencias se multiplicaría y se necesitarían 200 sentencias para solamente cargar los datos. Observando, se determina que solo cambia el subíndice, proceso que se puede hacer usando una estructura de repetición, en donde el ciclo genere el subíndice. A continuación se explica tal proceso.

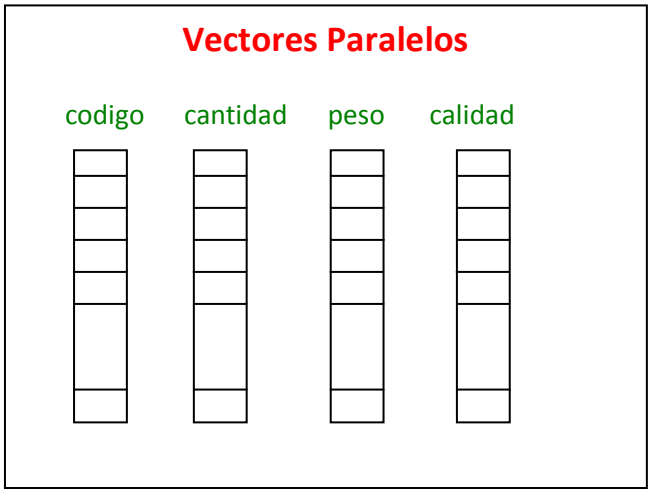
## CARGA O LECTURA DE ARREGLOS UNIDIMENSIONALES

**CASO 1**.- Se conoce de antemano el número exacto de valores a almacenar.

Ejemplo: Efectuar la declaración de variables y el proceso de almacenamiento en arreglos unidimensionales, de los códigos (*cadena*), cantidad (*entero*), peso (*real*) y calidad (*caracter*) de 50 diferentes herramientas usadas en la construcción civil.

Se conoce el numero de herramientas = 50.

Se denominan vectores paralelos, por cuanto tiene el mismo tamaño y los datos ubicados en la misma posición, están relacionados, es decir, el código de la herramienta ubicada en la posición 0, se corresponde con la cantidad, el peso y la calidad, en la misma posición.




Porción del programa en lenguaje C, para leer los datos

```
char      codigo [ 50 ][ 25  ]  ;
int       cantidad [ 50 ]  , k ;
float     peso [ 50 ]   ;
char      calidad [ 51 ]  ;

// Lectura o carga de los datos :

for (  k = 0  ;  k<  50  ;  k ++ )
    {
       printf ( " \n Ingrese el código de la herramienta Numero:  %d " , k + 1);
       gets( codigo [ k ]) ;
       printf ( " \n Ingrese la cantidad, el peso y la calidad de la herramienta");
       scanf("%d%f%c",  &cantidad [ k ] ,   &peso [ k ],  &calidad [ k ]) ;
    }
```

**CASO 2.-** Se desconoce el número exacto de valores a almacenar.

Ejemplo: Tómese el mismo ejemplo anterior, considerando que el numero de herramientas se desconoce.

Para este caso el programador debe asumir un máximo de herramientas, por cuanto es necesario realizar la declaración de las variables antes de efectuar la carga de los datos.

**Forma A**: Haciendo uso de un ciclo condicional( do-while), en donde para repetir el proceso se debe cumplir dos condiciones:

1.- Que el usuario confirme que quedan datos por almacenar.
2.- Que la variable tipo arreglo aun disponga de espacio.

Considerando un máximo de 40 herramientas.

```
char      codigo [ 40 ][ 25  ]  ;
int       cantidad [ 40 ]  , z = 0  ;
float     peso [ 40 ]   ;
char      calidad [ 41 ]  , resp ;

// Lectura o carga de los datos

do
{
        printf ( " \n Ingrese el código de la herramienta Numero:  %d " , z + 1);
        gets( codigo [ z ]) ;
        printf ( " \n Ingrese la cantidad, el peso y la calidad de la herramienta");
        scanf("%d%f%c",  &cantidad [ z ] ,   &peso [ z ],  &calidad [ z ]) ;
        z ++ ;
        printf( "\n ¿ Otra herramienta por almacenar sus datos ? S o N  " ) ;
        scanf ( " %c " ,  &resp ) ;
}    while (  tolower ( resp ) == 's '   &&  z < 40  ) ;
```

Respuesta dada por el usuario, confirmando que hay más datos.

Verificación que aun hay espacio en el arreglo.

**Nota:** La variable ( **z** ), al finalizar el ciclo, contiene el número real de valores almacenados, lo que permite que a continuación de la carga, se pueden acceder los arreglos con ciclos automáticos, usando a ( **z** ) número exacto de herramientas.

**Forma B:** Solicitando al usuario el numero de valores a almacenar, validando que el numero dado no supere la cantidad declarada y luego usar un ciclo automático.

```
char      codigo [ 40 ][ 25  ]  ;
int       cantidad [ 40 ]   k , nh ;
float     peso [ 40 ]   ;
char      calidad [ 41 ]  ;

do        // Proceso de Validación de la cantidad dada por el usuario
    {
       printf( "\n Indique el número exacto de herramientas, máximo = 40 " ) ;
       scanf( "%d " ,  &nh ) ;
       if ( nh <  1  || nh > 40 )
            printf ("\n Disculpe su cantidad debe ser como máximo 40 ");
    }  while (nh <  1  || nh >  40) ;


       // Lectura o carga de los arreglos

for (  k = 0  ;  k < nh  ;  k ++ )
    {
      printf ( " \n Ingrese el código de la herramienta Numero:  %d " , k + 1);
      gets( codigo [ k ]) ;
      printf ( " \n Ingrese la cantidad, el peso y la calidad de la herramienta");
      scanf("%d%f%c",  &cantidad [ k ] ,   &peso [ k ],  &calidad [ k ]) ;
    }
```

Las funciones: printf, scanf, gets, tolower, toupper y otras requieren incorporar en los archivos de cabecera la librería correspondiente, que contiene tal función, para poder usarlas en un programa. En el cuadro siguiente se indican algunas funciones y su correspondiente librería.

## LIBRERÍAS y FUNCIONES

| Librería o Biblioteca | Función |
|---|---|
| | |
| #include<stdio.h> | printf |
| | scanf |
| | getchar |
| | putchar |
| | gets |
| | puts |
| | fflush |
| | |
| #include<ctype.h> | tolower |
| | toupper |
| #include<conio.h> | clrscr |
| | gotoxy |
| | textbackground |
| | textcolor |
| | getch |
| | getche |
| #include<string.h> | strlen |
| | strcpy |
| | strcmp |
| | strcmpi |
| #include<stdlib.h> | system |

# PROBLEMA

En una planta de ensamblaje de automóviles, se labora en varios puestos de trabajo. El departamento de Producción ha tomado lectura de: el consumo de aire comprimido y de la temperatura en grados centígrados, en cada uno de los puestos de trabajo, para un día de trabajo. Ejemplo:

| **PUESTO DE TRABAJO** | **CONSUMO DE AIRE COMPRIMIDO ( Valor entero en Libras)** | **TEMPERATURA ( en $^{o}$C )** |
|---|---|---|
| Calderas | 3500 | 40,5 |
| Vestido | 6400 | 30,5 |
| Pintura | 5125 | 38,0 |
| -------- | ------- | --------- |
| -------- | ------- | --------- |
| -------- | ------- | --------- |

Diseñe un programa en lenguaje C, para procesar la información tomada durante el día de trabajo, que utilice arreglos unidimensionales y que permita:

a) Almacenar los datos recopilados en el día de ayer.(Considere máximo 15 puestos de trabajo)

b) Mostrar la información almacenada, en tres columnas organizadas, con su respectivo encabezamiento.

c) Convertir la temperatura a grados Fahrenheit ( $^{o}$F = 9/5 $^{o}$C + 32 ) y guardar en otro arreglo.

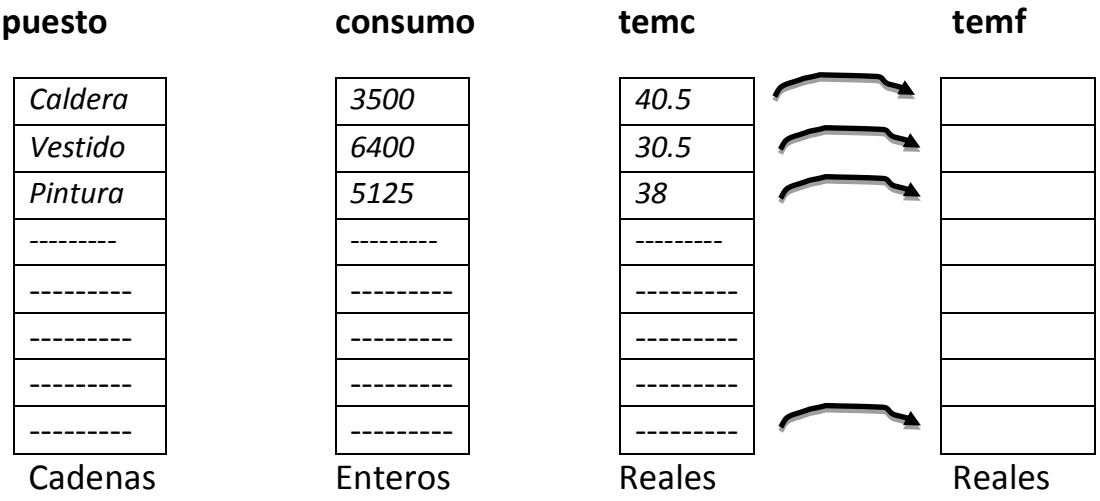
d) Calcular la temperatura promedio en grados Fahrenheit respecto a los puestos de trabajo, solo para aquellos donde el consumo de aire fue mayor a 3300 libras pero menor a 4500 libras.

e) Mencionar el puesto o puestos de trabajo, en donde se registro la menor temperatura.

f) Calcular y mostrar el porcentaje de consumo de aire comprimido de cada puesto de trabajo.

g) Mostrar la temperatura registrada en el puesto de Pintura. (Búsqueda)

h) Mostrar la temperatura en $^{o}$C y el consumo de aire, para un puesto de trabajo suministrado por teclado. (Consulta)




```
//          Programa Ensambladora de Vehículos
//          Autor: Armando Carrero      Fecha:      Lugar:

//          Archivos de Cabecera

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#define  MAX   15

//  Código para usar la función gotoxy en Dev C
#include<windows.h>
void gotoxy(int x, int y)
 {
        HANDLE hCon;
        COORD dwPos;
        dwPos.X = x;
        dwPos.Y = y;
        hCon = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleCursorPosition(hCon,dwPos);
 }

//Declaración de variables

char        puesto [ MAX ] [ 17 ] , resp ;
int         consumo [MAX] , w = 0 , i , k   ,  testigo , contf  = 0;
float       temc [MAX] ,  temf [MAX]  , acumf = 0 , prome, h,   suma_consumo = 0;
```

```
int main ( )     // Funcion Principal
  {

    // a.- Carga de datos

    do
       {
          printf ("\n Ingrese el nombre del puesto de trabajo No. %d ", w + 1 );
          gets  (puesto [ w ] ) ;
          printf( "\n Indique el consumo de aire, en libras, y la temperatura en grados
                    centígrados ") ;
          scanf ( " %d%f " ,  &consumo [ w ]   ,    &temc [ w ]  ) ;
          w ++ ;
          printf( "\n ¿ Otro Puesto por almacenar sus datos ? S o N  " ) ;
          fflush(stdin ) ;
          scanf ( " %c " ,  &resp ) ;
       }  while (tolower ( resp ) == 's'   &&w<MAX  ) ;


    //b.- Mostrar datos en Columnas organizadas

    system ("cls" ) ; // función para limpiar la pantalla y ubicar el cursor en la fila 1 columna 1

    gotoxy( 10 , 5 ) ;         // función que ubica el cursor en la columna 10 , fila 5 de la pantalla
    printf ( "Puesto" ) ;
    gotoxy( 22 , 5 ) ; printf ( "Consumo" ) ;
    gotoxy( 35 , 5 ) ; printf ( "Temperatura en C" ) ;

    for (  i = 0   ;  i <  w  ;  i ++ )
        {
              gotoxy ( 10 , 8 + i ) ; printf ( "%s" , puesto[ i ] ) ;
              gotoxy ( 22 , 8 + i ) ; printf ( "%d" , consumo [ i ] ) ;
              gotoxy ( 35 , 8 + i ) ; printf ( "%.2f" , temc [ i ] ) ;
        }

printf ( "\n\n" ) ;
system ("pause" ) ;
```

```
//c.- Crear un nuevo arreglo con las temperaturas en grados Fahrenheit

for (k = 0 ;  k<  w  ;  k ++ )
       temf [k ]   = (float)  9 / 5 *  temc [ k ]  + 32 ;



/*d.- Temperatura promedio en grados Fahrenheit respecto a los puestos de trabajo,
solo para aquellos donde el consumo de aire fue mayor a 3300 libras pero menor a
4500 libras*/


for (k = 0 ;  k<  w  ;  k ++ )
   if  (consumo [ k]  >= 3300   && consumo [ k] <= 4500 )
       {
            acumf  = acumf + temf [k ]  ;
            contf ++ ;
       }

if   (contf  !=  0  )
     {
       prome   = acumf  /contf ;
       printf( "\n\n\t\t La Temperatura promedio en grados Fahrenheit para los
       puestos donde el consumo de aire fue mayor a 3300 libras pero
       menor a 4500 libras fue de = %.2f" ,prome ) ;
     }
   else
       printf( "\n\n\t\t No hubo puestos donde el consumo de aire fue mayor a 3300
                  libras pero menor a 4500 libras" ) ;

system ("pause" ) ;          // función que detiene la ejecución del programa,
                             // haciendo una pausa para que el usuario observe los
                             // resultados, al pulsar ENTER se continua
```

```
//e.- Puesto o puestos de trabajo, en donde se registró la menor temperatura.

h = temc [ 0 ]  ;
for (k = 0 ;  k<  w  ;  k ++ )
       if  (temc [ k ]  < h )
            h = temc [ k ]  ;


system (“cls” ) ;


printf( “\n\t Puesto(s) de trabajo, en donde se registró la menor temperatura” ) ;

for (k = 0 ;  k<  w  ;  k ++ )
     if  (temc [ k ]  == h )
       printf( “\n\t\t  %s ” , puesto [ k ]   ) ;

system (“pause” ) ;


// f.- Calcular y mostrar el porcentaje de consumo de aire comprimido de cada puesto
//      de trabajo.

system (“cls” ) ;
gotoxy( 15 , 7 ) ; printf ( “Puesto” ) ;
gotoxy( 35 , 7 ) ; printf ( “Consumo en %%” ) ;

for (  i = 0   ;  i<  w  ;  i ++ )
            suma_consumo+= consumo [ i ] ;

for (  i = 0   ;  i<  w  ;  i ++ )
   {
            gotoxy ( 15 , 9 + i ) ; printf ( “%s” , puesto[ i ] ) ;
            gotoxy ( 35 , 9 + i ) ; printf ( “%.2f” , consumo [ i ] / suma_consumo * 100) ;
            }

system (“pause” ) ;
```

```
//g.- Mostrar la temperatura registrada en el puesto de Pintura. (Búsqueda)

testigo = 0 ;
k = 0;
do
   {
            h = strcmpi( puesto [ k  ] ,  "Pintura" )  ;
           if( h == 0)
              {
                  testigo = 1 ;
                  printf( "\n\t En el puesto de Pintura la temperatura fue de %.3f °C", temc [ k ]
           ) ;
              }
           k++ ;
    } while (  k<w   &&  testigo == 0 ) ;


if (testigo != 1 )
   printf( "\n\t El puesto de Pintura no fue registrado " ) ;

system ("pause" ) ;

return (0 );

}    // fin de la función main, fin del programa.
```