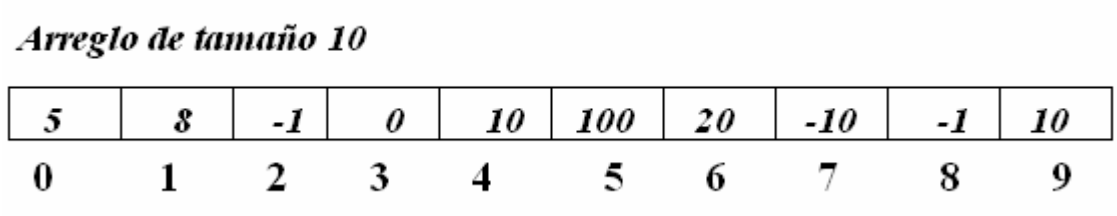
ÍNDICE

5 Estructura de Datos Básicas

5.1 Arreglos Unidimensionales

Un arreglo es una secuencia de datos del mismo tipo. Cada uno de los datos es considerado como un elemento del arreglo y estos están numerados consecutivamente, como por ejemplo:



Para identificar cada uno de los datos que se encuentran dentro del arreglo, se hace uso de un índice que indica la posición dentro del arreglo. Del ejemplo anterior, se determina que el valor 5 se encuentra en la posición 0, el valor 8 en la posición 1,…. el valor -1 en la posición 8 y el valor 10 en la posición 9.

Definición 9: Arreglos unidimensionales

Pseudocódigo
//**Declaración** de un arreglo especifico arregloEntero: 5, 8, -1, 0, 10, 100, 20, -10, -1, 10 //**Asignación** de valor en la posición 2 arregloEntero[2] ← 25 //**Lectura** de un valor en la posición 8 Valor ← arregloEntero [8]
En Código C
//**Declaración** de un arreglo especifico int arregloEntero [**10**] = { 5, 8, -1, 0, 10, 100, 20, -10, -1, 10 } //**Asignación** de valor en la posición 2 arregloEntero[2] = 25 //**Lectura** de un valor en la posición 8 Valor = arregloEntero [8]

Ejemplo 1 "Declaración y acceso a los valores de un arreglo"

Dado un arreglo de un tamaño fijo, se inicializa de manera estática, es decir, con valores que se colocan dentro del código fuente. El programa desplegara cada uno de los valores contenidos del arreglo, después se efectuaran algunas modificaciones y se volverá a desplegar el contenido.

Código en C

```
/*
  Archivo: ArreglosUni.c
  Descripcion: Este programa declara un arreglo de tamanio fijo a 5 elementos
               después el programa mandará a imprimir el contenido de cada una
               de las casillas del arreglo.
               Se efectuaran modificaciones en algunos valores y después
               se manda a imprimir nuevamente para ver las modificaciones
               efectuadas.
  Autor: Basurto Páez Gustavo
  Fecha: 21 de mayo de 2008
*/
```

```
/*Incluir librerias*/
#include <stdio.h>

/**Definicion de constantes*/
#define TAMANIO 5  //Tamanio del arreglo

int main ()
{
   //Declaración del arreglo de tipo entero, con valores inicializados
   int arreglo[TAMANIO]= {-1, 3, 32, -20, 22 };

   //Imprimir el contenido de cada una de las casillas del arreglo
   printf(" \n\n El contenido del arreglo es: ");
   printf(" \n El valor en la posicion 0 es: %d", arreglo[0]);
   printf(" \n El valor en la posicion 1 es: %d", arreglo[1]);
   printf(" \n El valor en la posicion 2 es: %d", arreglo[2]);
   printf(" \n El valor en la posicion 3 es: %d", arreglo[3]);
   printf(" \n El valor en la posicion 4 es: %d", arreglo[4]);

   //Se modifican los valores de las posiciones 0, 2 y 4
   arreglo[0] = 100;
   arreglo[2] = 0;
   arreglo[4] = -100;

   printf(" \n\n El contenido del arreglo es: ");
   //Se manda a imprimir nuevamente el contenido del arreglo
   printf(" \n El valor en la posicion 0 es: %d", arreglo[0]);
   printf(" \n El valor en la posicion 1 es: %d", arreglo[1]);
   printf(" \n El valor en la posicion 2 es: %d", arreglo[2]);
   printf(" \n El valor en la posicion 3 es: %d", arreglo[3]);
   printf(" \n El valor en la posicion 4 es: %d", arreglo[4]);

   getch();
   return (0);
}
```

Al momento de ejecutarlo, se manda a desplegar el contenido siguiente:

El contenido del arreglo es:
El valor en la posicion 0 es: -1
El valor en la posicion 1 es: 3
El valor en la posicion 2 es: 32
El valor en la posicion 3 es: -20
El valor en la posicion 4 es: 22

El contenido del arreglo es:
El valor en la posicion 0 es: **100**
El valor en la posicion 1 es: 3
El valor en la posicion 2 es: **0**
El valor en la posicion 3 es: -20
El valor en la posicion 4 es: **-100**

Ejercicios de Arreglos

Ejercicio 1 "***Aceptar enteros y almacenar en un arreglo***"

Entrada: 10 números enteros que se almacenarán en un arreglo de enteros.
Salida: ninguna

Pseudocódigo "Acepta Entero"

Comienza
cont ← 0
valor ← 0
Escribir (" Teclee los numeros enteros: ")
mientras (cont < 10) haz
comienza
Escribir (" Teclee el valor " (cont +1) ":")
Leer (valor)
enteros [cont] = valor;
cont ← cont + 1;
termina
Termina

Código en C

```
/*
  Archivo: AceptaEntero.c
  Descripción: Este programa recibe numeros enteros de la entrada estandar,
               tantos como lo indique la constante TAMANIO. Se hace uso de
               la iteración condicional While.
  Autor: Basurto Páez Gustavo
  Fecha: 21 de mayo de 2008
*/

/*Incluir librerias*/
#include <stdio.h>

/**Definición de constantes*/
#define TAMANIO 10 //Tamanio del arreglo

int main()
{
    int enteros[TAMANIO];
    int cont=0; /**indice del arreglo*/
    int valor=0;


    printf(" Teclee %d enteros.\n", TAMANIO);
    cont = 0;
    while (cont<TAMANIO)
    {
          printf("\nTeclee el %d valor: ", cont+1);
          scanf("%d", &valor);
          enteros[cont] = valor;
          cont++;
    }
    getch();
    return (0);
}
```

Ejercicio 2 "***Aceptar enteros y almacenar en un arreglo utilizando For***"

Resuelva el ejercicio 1 considerando la estructura de iteración **For**.

Ejercicio 3 "***Imprimir el contenido de un arreglo***"

Entrada: 8 valores introducidos por el usuario.

Salida: los 8 valores almacenados en un arreglo.

Código en C

```
/*
  Archivo: ImprimeArreglo.c
  Descripcion: Este programa recibe numeros enteros de la entrada estandar,
               se encarga de leertantos valores enteros como lo indique la constante TAMANIO.
               Posteriormente se manda a imprimir en pantalla
               el contenido del arreglo. En ambos se hace uso de la iteracion condicional
               For.
  Autor: Basurto Páez Gustavo
  Fecha: 20 de mayo de 2008
*/

/*Incluir librerias*/
#include <stdio.h>

/**Definición de constantes*/
#define TAMANIO 10 //Tamanio del arreglo

int main()
{
    int enteros[TAMANIO];
    int cont=0; /**indice del arreglo*/
    int valor=0;


    printf(" Teclee %d enteros.\n", TAMANIO);

   //Se leen los valores tecleados por el usuario y se almacenan
    for (cont = 0; cont<TAMANIO; cont++)
    {
          printf("\nTeclee el %d valor: ", cont+1);
          scanf("%d", &valor);
          enteros[cont] = valor;
    }

//Se mandan a escribir a pantalla
//Nota:  Se  hace  uso  nuevamente  de  la  estructura  de  control  iterativa  para
// recorrer el arreglo.

     printf(" Lista de Tamaño: %d .\n", TAMANIO);
     for(cont=0; cont < TAMANIO ; cont++)
     {
          printf("%d, ", enteros[cont]);
     }
     printf(" \n\n");


    getch();
    return (0);
}
```

Ejercicio 4 "***Buscar un elemento en un arreglo***"

Declarado un arreglo de manera estática, es decir, dentro del código del programa fuente; el usuario introducirá un número entero, le programa se encargara de buscar si se encuentra el valor dentro del arreglo y desplegara la posición en donde se encuentra.

Entrada: Valor a buscar dentro del arreglo.

Salida: Mensaje indicando la posición dentro del arreglo en donde se encuentra el valor o un mensaje indicando que no se encontró dicho valor.

Código en C

```
/*
  Archivo: BuscarElemento.c
  Descripcion: Este programa recibe un número entero de la entrada estándar,
               y el programa determina si dicho entero se encuentra en un
               arreglo de tamaño TAMANIO y que contiene valores inicializados.
               dentro del código fuente. Se hace uso de la iteración
               Condicional While.
```

```
  Autor: Basurto Páez Gustavo
  Fecha: 20 de mayo de 2008
*/
/*Incluir librerias*/
#include <stdio.h>

/**Definicion de constantes*/
#define TAMANIO 10  //Tamanio del arreglo

int main ()
{
    //Declaración del arreglo de tipo entero, con valores inicializados
    int arreglo[TAMANIO]= {-1, 3, 32, -20, 22, 49, 7, 9, 11, 101 };
    int entBuscar=0;
    int cont=0;
    int posicion = -1;
    int exito = -1;
    //exito = -1 NO se encontro, 'otro valor' SI se encontro y esta en la
    //la posicion del valor contenido en la variable


     printf("\n\n Teclee un numero entero: ");
     scanf("%d", &entBuscar);


     cont=0;
     //No es necesario recorrer todo
     while( cont < TAMANIO && exito == -1)
     {
            if(entBuscar == arreglo [cont])
            {
                 bandera = 1; //encontrado
                 posicion = cont; //en la posicion dada
            }
            //No es necesario el else

            cont++;
     }

    if(posicion  == -1 )
    {
       printf("\n\n El Entero %d NO se encuentra en el arreglo...\n",entBuscar );
    }
    else
    {
        printf("\n\n  El  Entero  %d  se  encuentra  en  la  posicion  %d  del  arreglo.
               ...\n",entBuscar, posicion );
    }
    getch();
    return (0);
}
```

5.2 Arreglos Multidimencionales – bidimensionales

Los arreglos multidimensionales son aquellos que tienen más de una dimensión y se hace uso de un índice que indique la poción que se encuentra dentro del arreglo. A los arreglos de dos dimensiones son conocidos como tablas o matrices, divididos en columnas y arreglos que para tener acceso a uno de los elementos de la matriz se utilizan dos índices, uno para colocarse en la columna y otro para colocarse en el renglón indicado.

En la siguiente matriz se tienen 4 renglones y 4 columnas, de la misma forma como los arreglos se utilicen índices para indicar la posición de algún elemento en particular dentro de la matriz. Los índices para columnas como para renglones empiezan en 0.

	0	1	2	3
0				
1				
2				
3				

Por ejemplo en el siguiente arreglo se accede a su contenido de la siguiente forma:

Arreglo en el renglón 0 y la columna 0 hay un 5 → arreglo [0][0]
Arreglo en el renglón 0 y la columna 1 hay un 10 → arreglo [0][1]
Arreglo en el renglón 0 y la columna 2 hay un -9 → arreglo [0][2]
Arreglo en el renglón 0 y la columna 3 hay un 8 → arreglo [0][3]
Arreglo en el renglón 1 y la columna 0 hay un 0 → arreglo [1][0]
Arreglo en el renglón 1 y la columna 1 hay un -1 → arreglo [1][1]

	0	1	2	3
0	5	10	-9	8
1	0	-1	-3	0
2	20	18	-6	-10
3	-5	-3	0	-1

Definición 10: Arreglos bidimensionales

Pseudocódigo

```
//Declaración de una matriz especifico
matrizEntero: { {5,      8,   -1,   0,  10},
                {100, 20,  -10,  -1,  10}   }

//Asignación de valor en el renglón 1 y columna 3
matrizEntero[1][3] ← 25

//Lectura de un valor en la posición de renglón 0 y columna 2
Valor ← matrizEntero [0][2]
```

En Código C

```
//Declaración de una matriz especifico
int matrizEntero [ 2 ][ 5 ] =  { {5,      8,   -1,   0,  10},
                                 {100, 20,  -10,  -1,  10}   } ;

//Asignación de valor en el renglón 1 y columna 3
matrizEntero[1] [3]  =  25

//Lectura de un valor en la posición del renglón 0 y columna 2
Valor = matrizEntero [0][2]
```

Ejemplo 1 "Declaración y acceso a los valores de un arreglo"

Dado una matriz de un tamaño fijo, se inicializa de manera estática, es decir, con valores que se colocan dentro del código fuente. El programa desplegara cada uno de los valores contenidos de la matriz, después se efectuaran algunas modificaciones y se volverá a desplegar el contenido.

Código en C

```
/*
  Archivo: Matrices.c
  Descripción:  Este programa hace uso de matrices de una dimensión de 2
                renglones por 3 columnas. Se manda a imprimir, después se
                efectúan algunas modificaciones y finalmente se imprime nuevamente
  Autor: Gustavo Basurto Páez.
  Fecha: 21 de mayo de 2008
*/
#include <stdio.h>

//Declaracion de las constantes para determinar el tamanio de la matriz
#define MAXCOLUMNAS  3
#define MAXRENGLONES 2

int main()
{

  int matriz[MAXRENGLONES][MAXCOLUMNAS] ={
                  {1,2,3},
```

```
                {5,4,3},
                };

   //Se manda a imprimir el contenido de la matriz
   printf(" \n\nEl contenido de la matriz es: ");
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 0,0,
               matriz[0][0]);
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 0,1,
               matriz[0][1]);
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 0,2,
               matriz[0][2]);
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 1,0,
               matriz[1][0]);
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 1,1,
               matriz[1][1]);
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 1,2,
               matriz[1][2]);

   //Se efectuan algunas modificaciones
   matriz [ 0 ] [ 0 ] = -10;
   matriz [ 0 ] [ 2 ] = 100;
   matriz [ 1 ] [ 0 ] = 0;
   matriz [ 1 ] [ 1 ] = -1;

   printf(" \n\nEl contenido de la matriz es: ");
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 0,0,
               matriz[0][0]);
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 0,1,
               matriz[0][1]);
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 0,2,
               matriz[0][2]);
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 1,0,
               matriz[1][0]);
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 1,1,
               matriz[1][1]);
   printf(" \nContenido en la Matriz en renglon: %d y Columna: %d es : %d", 1,2,
               matriz[1][2]);

  getch();
  return 0;

}
```

La evidencia de ejecución es la siguiente:

```
El contenido de la matriz es:
Contenido en la Matriz en renglon: 0 y Columna: 0 es : 1
Contenido en la Matriz en renglon: 0 y Columna: 1 es : 2
Contenido en la Matriz en renglon: 0 y Columna: 2 es : 3
Contenido en la Matriz en renglon: 1 y Columna: 0 es : 5
Contenido en la Matriz en renglon: 1 y Columna: 1 es : 4
Contenido en la Matriz en renglon: 1 y Columna: 2 es : 3

El contenido de la matriz es:
Contenido en la Matriz en renglon: 0 y Columna: 0 es : -10
Contenido en la Matriz en renglon: 0 y Columna: 1 es : 2
Contenido en la Matriz en renglon: 0 y Columna: 2 es : 100
Contenido en la Matriz en renglon: 1 y Columna: 0 es : 0
Contenido en la Matriz en renglon: 1 y Columna: 1 es : -1
Contenido en la Matriz en renglon: 1 y Columna: 2 es : 3
```

Ejercicios de Matrices

Ejercicio 1 "Imprimir el contenido de una matriz"

Se desarrolla un programa que mande a imprimir el contenido de una matriz. Para la manipulación de las matrices se requiere del uso de dos índices, uno para recorrer los renglones y otro para las columnas.

Código en C

```
/*
  Archivo: ImprimirMatriz.c
  Descripcion:  Este programa hace uso de matrices de una dimencion de 2
                renglones por 3 columnas. Se recorre mara mandar a imprimir
                su contenido, se hace uso de dos indices para recorr los
                renglones y otro para recorrer las columnas.
  Autor: Gustavo Basurto Páez.
  Fecha: 21 de mayo de 2008
*/

#include <stdio.h>

//Declaracion de las constantes para determinar el tamanio de la matriz
#define MAXCOLUMNAS  3
#define MAXRENGLONES 2


int main()
{
  int contReng, contCol; //indices para acceder al contenido de la matriz
  int matriz[MAXRENGLONES][MAXCOLUMNAS] ={
                  {1,2,3},
                  {5,4,3},
                  };

 //El primer for es para recorrer los englones
  for(contReng =0; contReng<MAXRENGLONES; contReng++ )
  {
         printf("\n\t [ ");

         //El segundo for es para recorrer las columnas
         for(contCol =0; contCol<MAXCOLUMNAS; contCol++ )
         {
              //se manda a impirmir accediendo al renglon y columna dado
              printf(" %d, ", matriz[contReng][contCol]);
         }
         printf(" ] ");
  }

  getch();
  return 0;

}
```

Evidencia del funcionamiento del programa.

```
        [  1,  2,  3,  ]
        [  5,  4,  3,  ]
```

Ejercicio 2 "Almacenar la información en una matriz"

Se desarrollará un programa que solicite al usuario que ingrese un valor de acuerdo a la posición en donde se encuentre en la matriz. Con esto se completará el contenido de la matriz con valores introducidos por el usuario. Después se envía a imprimir el contenido con el objetivo verificar que se haya almacenado la información correctamente.

Código en C

```
/*
  Archivo: LeerMatriz.c
  Descripcion:  Este programa hace uso de matrices de una dimensión de 3
                renglones por 4 columnas.
                Se solicita al usuario que ingrese un valor de acuerdo
                a la posición en donde se encuentre en la matriz.
                Con esto se completará el contenido de la matriz con
                valores introducidos por el usuario.
                Después se enviará a imprimir el contenido para verificar
                que se haya almacenado la información correctamente.
  Autor: Gustavo Basurto Páez.
  Fecha: 21 de mayo de 2008
*/

#include <stdio.h>

//Declaracion de las constantes para determinar el tamanio de la matriz
#define MAXCOLUMNAS  4
#define MAXRENGLONES 3

int main()
{
  int contReng, contCol; //indices para acceder al contenido de la matriz
  int matriz[MAXRENGLONES][MAXCOLUMNAS] ;

  //Se manda a solicitar que el usuario teclee la informacion
  printf(" \n\n Debera de proporcionar los valores a ingresar a la matriz \n");
 //El primer for es para recorrer los englones
  for(contReng =0; contReng<MAXRENGLONES; contReng++ )
  {
         //El segundo for es para recorrer las columnas
         for(contCol =0; contCol<MAXCOLUMNAS; contCol++ )
         {
              //se manda a solicitar un valor para almacenarlo
              //en renglon y columna dado
              printf(" \n Teclee el valor Matriz [%d][%d]: ", contReng, contCol);
              scanf ("%d", &matriz[contReng][contCol]);
         }
  }

  //Se manda a imprimir el contenido de la matriz
  printf(" \n\n El contenido de la matriz es el siguiente: \n");

 //El primer for es para recorrer los englones
  for(contReng =0; contReng<MAXRENGLONES; contReng++ )
  {
         printf("\n\t [ ");

         //El segundo for es para recorrer las columnas
         for(contCol =0; contCol<MAXCOLUMNAS; contCol++ )
         {
              //se manda a impirmir accediendo al renglon y columna dado
              printf(" %d, ", matriz[contReng][contCol]);
         }
         printf(" ] ");
  }
  getch();
  return 0;
}
```

Evidencia de ejecución del programa:

```
Deberá de proporcionar los valores a ingresar a la matriz
Teclee el valor Matriz [0][0]: 1
Teclee el valor Matriz [0][1]: 2
Teclee el valor Matriz [0][2]: 3
Teclee el valor Matriz [0][3]: 4
Teclee el valor Matriz [1][0]: 5
Teclee el valor Matriz [1][1]: 6
Teclee el valor Matriz [1][2]: 7
Teclee el valor Matriz [1][3]: 8
Teclee el valor Matriz [2][0]: 9
Teclee el valor Matriz [2][1]: 10
Teclee el valor Matriz [2][2]: 11
Teclee el valor Matriz [2][3]: 12

El contenido de la matriz es el siguiente:
        [  1,  2,  3,  4,  ]
        [  5,  6,  7,  8,  ]
        [  9,  10,  11,  12,  ]
```

5.3 Cadenas

Una cadena es considerada como un conjunto de caracteres que se encuentran agrupados entre comillas dobles, como por ejemplo: “Esto es una cadena”, “1233445677”.

En C es posible utilizar un arreglo de caracteres como si fuera una cadena, solo hay una diferencia importante. Las cadenas a diferencia de los arreglos de caracteres siempre deben de tener al final el carácter nulo (‘\0’).

Ejemplo de arreglo de caracteres: “arreglo de caracteres”
Ejemplo de cadena: “Cadena de caracteres**0**”

De la misma manera que los arreglos de enteros o flotantes, el contenido de los arreglos de caracteres pueden ser accedidos mediante el uso de un índice.

C	a	d	e	n	a		h	o	l	a	\0
0	1	2	3	4	5	6	7	8	9	10	11

Observe que el carácter nulo (‘\0’) se encuentra en la posición 11, indicando así que se esta representando a una cadena. Si no tuviese el carácter nulo al final se trataría de un arreglo de caracteres.

Definición 11: Cadenas

Pseudocódigo
//**Declaración** de una cadena cadena : {‘H’, ‘o’, ‘l’ , ‘a’ , ‘ ‘ , ‘a’ , ‘d’, ‘i’, ‘o’, ‘s’, ‘\0’ } //**Asignación** de un nuevo carácter cadena [1] ← ‘O’ //**Lectura** de un carácter de la cadena Valor ← cadena [0] //Escribir una cadena Escribir (cadena)
En Código C
//**Declaración** de una cadena char cadena [ ] = {‘H’, ‘o’, ‘l’ , ‘a’ , ‘ ‘ , ‘a’ , ‘d’, ‘i’, ‘o’, ‘s’, ‘**\0**’ }; char cadena2 [] = “Hola y Adios**\0**”

```
//Asignación de un nuevo carácter en la posición 1 de cadena
cadena [1]  =  'O'

//Lectura de un carácter en la posición 3 de la cadena
valor = cadena  [3]

//Escribir una cadena a pantalla
printf ( "%s", cadena);
```

Ejemplo 1 "Declarar, modificar y escribir una cadena a pantalla"

Desarrollar un programa que haga uso de cadenas, tanto en la declaración, manipulación y escritura a pantalla.

Código en C

```
/*
  Archivo: DecCadenas.c
  Descripcion: Este programa muestra como se hace uso de las cadenas.
  Autor: Basurto Páez Gustavo
  Fecha: 21 de mayo de 2008
*/
/*Incluir librerias*/
#include <stdio.h>

int main()
{
    //Declaracion de una cadena.
    char cadena1[]="C1: Hola.\0";

    //Declaracion de una cadena. Note en donde se encuentra el caracter nulo
    char cadena2[]="C2: Hola.\0 como estas";
    char cadena3[]= {  'C', '3', ':', ' ', 'H' , 'O' , 'L' , 'A',
                       '.' , '.' , '\0'
                       };

    //Se envia a imprimir el contenido de las cadenas
    printf("\n\n Cadena 1: %s", cadena1);
    printf("\n\n Cadena 2: %s", cadena2);
    printf("\n\n Cadena 3: %s", cadena1);

    printf("\n\n Se efectuan modificaciones ");
    //Se modifican la informacion de la cadena 1
    cadena1 [6] = '\0';

    printf("\n\n Cadena 1: %s", cadena1);

    //Se modifican la informacion de la cadena 2
    cadena2 [4] = '@';
    cadena2 [9] = '.'; //Se reemplaza el caracte nulo
    cadena3 [20] = '\0';

    printf("\n\n Cadena 2: %s", cadena2);

    getch();
    return 0;
}
```

Evidencia del funcionamiento del programa

```
Cadena 1: C1: Hola.
Cadena 2: C2: Hola.
Cadena 3: C1: Hola.
Se efectúan modificaciones
Cadena 1: C1: Ho
Cadena 2: C2:
```

Ejercicios con el manejo de cadenas

Ejercicio 1 "Lectura de caracteres y el manejo de cadenas"

Se desarrollará un programa en donde se solicite al usuario que teclee una serie de caracteres hasta que se teclee el 'enter', cada uno de los caracteres será almacenado en un arreglo de caracteres el cual después será considerado como una cadena.

Código en C

```
/*
  Archivo: LecCadena.c
  Descripción: Este programa muestra como se lee desde teclado una serie
               de caracteres hasta que el usuario teclee enter y
               se almacena cada uno de los caracteres en el arreglo, para
               que al final se considere como cadena.
  Autor: Basurto Páez Gustavo
  Fecha: 21 de mayo de 2008
*/

#include <stdio.h>

//Simpre considerar el caracter nulo '\0'
//TAmanio de la cadena es de 20
#define TAMCADENA  21

int main()
{
    char Cadena [TAMCADENA+1];
    char letra;
    int cont =0;


    printf("\nTeclee una cadena de caracteres (finalice con enter):");
    //El usuario teclea hasta un número limite o bien hasta q presiona enter
    while(letra !='\n' && cont < TAMCADENA)
    {
      //Se lee un caracter
      letra = getchar();
      Cadena[cont]= letra;
      cont++;
    }
    cont--; //se incremento en 1
    Cadena [cont]='\0'; //fin de cadena

    printf("\n\nLa cadena es: ");
    puts(Cadena);

    printf("\n\La cadena (tamanio %d) es: %s", cont, Cadena);
    printf("\nAdios!!..");
    getch();
    return 0;
}
```

Evidencia de ejecución del programa:

```
Teclee una cadena de caracteres (finalice con enter):Hola como estas?
La cadena es: Hola como estas?
La cadena (tamanio 16) es: Hola como estas?
Adios!!..
```

Ejercicio 2 "Copia de una cadena en otra"

Se desarrollará un programa en donde se solicite al usuario que teclee una serie de caracteres hasta que se teclee el 'enter', cada uno de los caracteres será almacenado en un arreglo de caracteres el cual después será considerado como una cadena. Posteriormente se efectuara la copa de la cadena origen a una cadena destino utilizado la función **strcpy()**.

Código en C

```
/*
  Archivo: CopiaCadena.c
  Descripcion: Este programa muestra como se efectua la copia de una
               cadena a otra utilizando la funcion strcpy().
  Autor: Basurto Páez Gustavo
  Fecha: 21 de mayo de 2008
*/


#include <stdio.h>
//Simpre considerar el caracter nulo '\0'
//TAmanio de la cadena es de 20
#define TAMCADENA  21

int main()
{
    char CadenaOrigen [TAMCADENA];
    char CadenaDestino [TAMCADENA];
    char letra;
    int cont =0;

    printf("\nTeclee una cadena de caracteres (finalice con enter):");
    //El usuario teclea hasta un numero limite o bien hasta q presiona enter
    while(letra !='\n' && cont < TAMCADENA)
    {
      //Se lee un caracter
      letra = getchar();
      CadenaOrigen[cont]= letra;
      cont++;
    }
    cont--; //se incremento en 1
    CadenaOrigen [cont]='\0'; //fin de cadena

    printf("\n\nLa cadena Origen  es: ");
    puts(CadenaOrigen);

    printf("\n\nLa cadena Destino es: ");
    puts(CadenaDestino);

    printf("\n\n Se copia la cadena Origen a Destino...");

    strcpy(CadenaDestino, CadenaOrigen);

    printf("\n\nLa cadena Destino es: ");
    puts(CadenaDestino);

    printf("\nAdios!!..");
    getch();
    return 0;
}
```

Evidencia de la ejecución del programa:

```
Teclee una cadena de caracteres (finalice con enter):Hola que tal?...
La cadena Origen  es: Hola que tal?...
La cadena Destino es: ╬╠┐w↑ "

Se copia la cadena Origen a Destino...
La cadena Destino es: Hola que tal?...
Adios!!..
```

5.4 Ejercicios de Unidad 5

Los siguientes ejercicios permiten ejercitar aspectos prácticos de la unidad 5. Algunos ejercicios son propuestas el desarrollo del Algoritmo, Diagrama de Flujo y/o Pseudocódigo, antes de comenzar a efectuar el Código en C.

Para el desarrollo de los siguientes ejercicios se requiere efectuar previamente las fases de Análisis y Diseño, ya que se contempla desarrollar las siguientes fases: Codificación, Compilación, Ejecución y Pruebas, Documentación y Mantenimiento.

I. Ejercicios que hacen uso de arreglos unidimensionales.

1. "***Buscar el elemento mayor de un arreglo utilizando while***"

Dado un arreglo de tamaño fijo, elaborar un programa que busque el elemento más grande dentro del arreglo. El programa se encargara de buscar el elemento mayor de todos, debe de desplegarlo en pantalla incluyendo la posición que se encuentra dentro del arreglo.

2. "***Buscar el elemento mayor y menor de un arreglo For***"

El programa debe de solicitar que el usuario teclee los valores del arreglo, para posteriormente buscar el elemento mayor y menor dentro del arreglo, se debe desplegar tanto el valor mayor y menor, como las posiciones en donde se encuentra.

3. "***Suma de dos vectores***"

El programa debe de solicitar que el usuario teclee los valores dos arreglos del mismo tamaño, posteriormente se efectuara la suma de cada uno de los elementos del arreglo para guardar el resultado de la suma en un tercer arreglo. Al finalizar el programa deberá de desplegar el arreglo con el resultado de la operación.

4. "***Ordenar en forma creciente un arreglo***"

El programa debe de solicitar que el usuario teclee los valores de un arreglo, posteriormente se efectuaré el ordenamiento de forma creciente del contenido del arreglo. Al finalizar el programa deberá de desplegar el arreglo de manera ordenada.

II. Ejercicios que hacen uso de arreglos multidimencionales.

1. "***Buscar un elemento en una matriz***"

Dada una matriz de tamaño fijo, elaborar un programa que lee un número y se encargue de buscar la primer ocurrencia del número en la matriz. El programa deberá de desplegar la posición en donde se encuentra dicho valor en la matriz o un mensaje indicando que no se encuentra dicho valor.

2. "***Buscar el mayor y menor de los elemento en una matriz***"

Desarrolle un programa que solicite al usuario cada uno de los valores que se almacenarán en una matriz de tamaño determinado. Posteriormente el programa se encargara de recorrer la matriz en busca del elemento mayor y menor de toda la matriz. El programa deberá de desplegar la posición en donde se encuentra el mayor y el menor valor en la matriz.

3. "***Suma de dos matrices***"

Investigar como se lleva a cabo el proceso de sumar dos matrices y desarrolle un programa que reciba dos matrices y posteriormente efectué la suma de ambas. Al finalizar se debe de desplegar la matriz resultante de la operación.

4. "***Buscar el mayor y menor de los elemento en una matriz***"

Investigar como se lleva a cabo el proceso de multiplicar dos matrices y desarrolle un programa que reciba dos matrices y posteriormente efectué la multiplicación de ambas. Al finalizar se debe de desplegar la matriz resultante de la operación.

III. Ejercicios que hacen uso de cadenas.

1. "***Convierta en mayúsculas***"

Desarrolle un programa que dada una cadena proporcionada por el usuario, el programa efectué la conversión de todos las letras minúsculas a mayúsculas.

2. "***Contador de ocurrencias de un símbolo***"

Desarrolle un programa que dada una cadena proporcionada por el usuario y un carácter, el programa cuente las ocurrencias de dicho carácter dentro de la cadena.

3. "***Contador de vocales***"

Desarrolle un programa que dada una cadena proporcionada por el usuario, el programa deberá contar el numero de 'a', 'e', 'i', 'o' y 'u' que se encuentren dentro de la cadena tecleada. Debe de considerar las vocales mayúsculas y minúsculas.

4. "***Comparador de cadenas***"

Desarrolle un programa que dada dos cadenas proporcionadas por el usuario, el programa deberá determinar si dichas cadenas son iguales o no, además deberá de desplegar la longitud de cada una de ellas..

5. "***Buscador de cadenas***"

Desarrolle un programa que dada dos cadenas proporcionadas por el usuario, con la condicion que la segunda cadena proporcionada sea más corta que la primera. El programa deberá determinar si la segunda cadena se encuentra dentro de la primera cadena.