



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* M.I. Marco Antonio Martínez Quintana

*Asignatura:* Fundamentos de Programación

*Grupo:* 3 Bloque: 136

*No de Práctica(s):* 11: Arreglos unidimensionales y multidimensionales

*Integrante(s):* Carranza Ochoa José David

*No. de Equipo de  
cómputo empleado:* No aplica

*No. de Lista o Brigada:* 6

*Semestre:* 2021-1

*Fecha de entrega:* 11/01/2021

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

## Objetivo:

Reconocer la importancia y utilidad de los arreglos, en la elaboración de programas que resuelvan problemas que requieran agrupar datos del mismo tipo, así como trabajar con arreglos tanto unidimensionales como multidimensionales.

## Actividades:

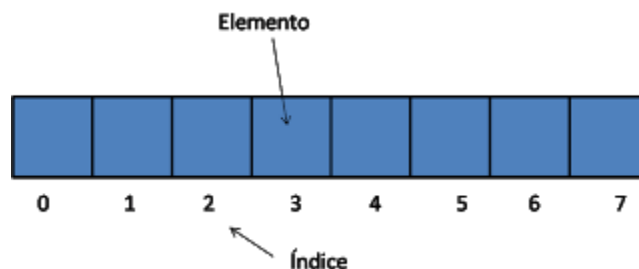
- ♣ Elaborar un programa en lenguaje C que emplee arreglos de una dimensión.
- ♣ Resolver un problema que requiera el uso de un arreglo de dos dimensiones, a través de un programa en lenguaje C.
- ♣ Manipular arreglos a través de índices y apuntadores.

## Introducción

Un arreglo es un conjunto de datos contiguos del mismo tipo con un tamaño fijo definido al momento de crearse. A cada elemento (dato) del arreglo se le asocia una posición particular, el cual se requiere indicar para acceder a un elemento en específico. Esto se logra a través del uso de índices. Los arreglos pueden ser unidimensionales o multidimensionales. Los arreglos se utilizan para hacer más eficiente el código de un programa.

### Arreglos unidimensionales

Un arreglo unidimensional guarda información en cada bloque de memoria comenzando desde el 0 hasta  $n-1$ , donde  $n$  es el tamaño del arreglo



Su sintaxis del arreglo es:

`tipoDeDato nombre[tamaño]`

Cabe destacar que solo puede tomar valores del mismo tipo de dato empleado, siendo enteros, reales, caracteres o estructurados.

## Código (arreglo unidimensional while)

Comenzamos con el primer ejemplo de arreglo observando su comportamiento

```
1 #include <stdio.h>
2 /*
3  Este programa genera un arreglo unidimensional de 5 elementos y los
4  accede a cada elemento del arreglo a través de un ciclo while.
5  */
6 int main (){
7     #define TAMANO 5
8     int lista[TAMANO] = {10, 8, 5, 8, 7};
9
10    int indice = 0;
11
12    printf("\tLista\n");
13    while (indice < 5 ){
14        printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
15        indice += 1; // análogo a indice = indice + 1;
16    }
17
18    printf("\n");
19
20    return 0;
21 }
```

Lista

Calificación del alumno 1 es 10  
Calificación del alumno 2 es 8  
Calificación del alumno 3 es 5  
Calificación del alumno 4 es 8  
Calificación del alumno 5 es 7

...Program finished with exit code 0  
Press ENTER to exit console.

## Código (arreglo unidimensional for)

```
1 #include <stdio.h>
2 /*
3  Este programa genera un arreglo unidimensional de 5 elementos y
4  accede a cada elemento del arreglo a través de un ciclo for.
5  */
6 int main (){
7     #define TAMANO 5
8     int lista[TAMANO] = {10, 8, 5, 8, 7};
9
10    printf("\tLista\n");
11    for (int indice = 0 ; indice < 5 ; indice++){
12        printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
13    }
14
15    printf("\n");
16
17    return 0;
18 }
```

Lista

Calificación del alumno 1 es 10  
Calificación del alumno 2 es 8  
Calificación del alumno 3 es 5  
Calificación del alumno 4 es 8  
Calificación del alumno 5 es 7

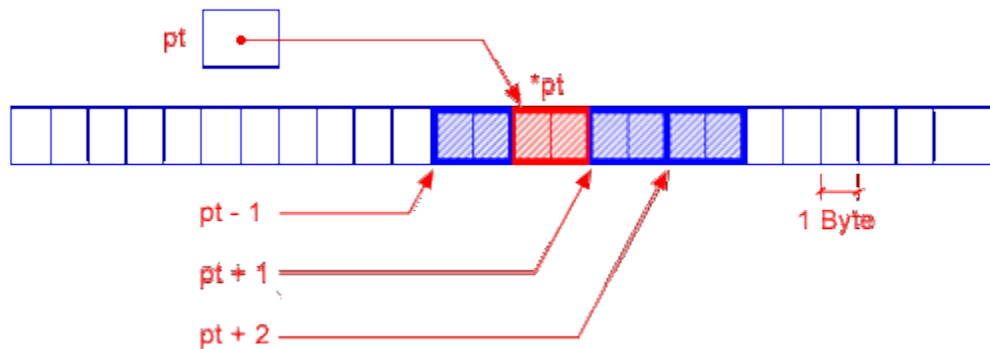
...Program finished with exit code 0  
Press ENTER to exit console.

## Apuntadores

Los apuntadores en C se refieren a la dirección de una variable descrita anteriormente, gracias a ellos es posible ingresar a la variable de forma más rápida debido a que trabajan con memoria. Su sintaxis se define como:

TipoDeDato \*apuntador, variable;

apuntador = &variable;



Con esto se tienen los siguientes códigos ejemplificando su uso, observemos cada caso

```
1 #include <stdio.h>
2 /*
3  Este programa crea un apuntador de tipo carácter.
4  */
5 int main () {
6     char *ap, c = 'a';
7     ap = &c;
8
9     printf("Carácter: %c\n", *ap);
10    printf("Código ASCII: %d\n", *ap);
11    printf("Dirección de memoria: %d\n", ap);
12
13    return 0;
14 }
```

input

```
main.c:11:34: warning: format '%d' expects argument of type 'int', but argument 2 has type 'char *' [-Wformat=]
Carácter: a
Código ASCII: 97
Dirección de memoria: -120685865

...Program finished with exit code 0
Press ENTER to exit console.
```

## Código (apuntadores)

```
1 #include<stdio.h>
2 /*
3  Este programa accede a las Localidades de memoria de distintas variables a
4  través de un apuntador.
5  */
6 int main () {
7     int a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0};
8     int *apEnt;
9     apEnt = &a;
10
11     printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n");
12     printf("apEnt = &a\n");
13
14     b = *apEnt;
15     printf("b = *apEnt \t-> b = %i\n", b);
16
17     b = *apEnt + 1;
18     printf("b = *apEnt + 1 \t-> b = %i\n", b);
19
20     *apEnt = 0;
21     printf("*apEnt = 0 \t-> a = %i\n", a);
22
23     apEnt = &c[0];
24     printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt);
25
26     return 0;
27 }
28
```

```
a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}
apEnt = &a
b = *apEnt      -> b = 5
b = *apEnt + 1  -> b = 6
*apEnt = 0      -> a = 0
apEnt = &c[0]   -> apEnt = 5
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

```
1 #include <stdio.h>
2 /*
3  Este programa trabaja con aritmética de apuntadores para acceder a los
4  valores de un arreglo.
5  */
6 int main () {
7     int arr[] = {5, 4, 3, 2, 1};
8     int *apArr;
9     apArr = arr;
10
11     printf("int arr[] = {5, 4, 3, 2, 1};\n");
12     printf("apArr = &arr[0]\n");
13
14     int x = *apArr;
15     printf("x = *apArr \t -> x = %d\n", x);
16
17     x = *(apArr+1);
18     printf("x = *(apArr+1) \t -> x = %d\n", x);
19
20     x = *(apArr+2);
21     printf("x = *(apArr+1) \t -> x = %d\n", x);
22
23     return 0;
24 }
```

```
int arr[] = {5, 4, 3, 2, 1};
apArr = &arr[0]
x = *apArr      -> x = 5
x = *(apArr+1)  -> x = 4
x = *(apArr+1)  -> x = 3
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

### Código (apuntadores en ciclo for)

De la misma forma se puede emplear el uso de ciclos como vimos anteriormente, siguiendo una secuencia de pasos hasta que se cumpla la condición establecida.

```
1  #include <stdio.h>
2  /*
3   Este programa genera un arreglo unidimensional de 5 elementos y
4   accede a cada elemento del arreglo a través de un apuntador
5   utilizando un ciclo for.
6   */
7  int main (){
8   #define TAMANO 5
9   int lista[TAMANO] = {10, 8, 5, 8, 7};
10  int *ap = lista;
11  printf("\tLista\n");
12  for (int indice = 0 ; indice < 5 ; indice++){
13   printf("\nCalificación del alumno %d es %d", indice+1, *(ap+indice));
14  }
15
16  printf("\n");
17
18  return 0;
19  }
20
```

Lista

Calificación del alumno 1 es 10  
Calificación del alumno 2 es 8  
Calificación del alumno 3 es 5  
Calificación del alumno 4 es 8  
Calificación del alumno 5 es 7

...Program finished with exit code 0  
Press ENTER to exit console.

## Código (apuntadores en cadenas)

Recordando que se puede trabajar con caracteres al igual que números, el siguiente código ejemplifica dicha función utilizando las variables de tipo "char" en su estructura, la condición for imprime letra por letra dando un salto de línea.

```
1  #include <stdio.h>
2  /*
3   Este programa muestra el manejo de cadenas en lenguaje C.
4   */
5  int main(){
6      char palabra[20];
7      int i=0;
8      printf("Ingrese una palabra: ");
9      scanf("%s", palabra);
10     printf("La palabra ingresada es: %s\n", palabra);
11     for (i = 0 ; i < 20 ; i++){
12         printf("%c\n", palabra[i]);
13     }
14     return 0;
15 }
```

▼ ↗ 🖨

```
Ingrese una palabra: dos
La palabra ingresada es: dos
d
o
s

@

❖
x
[
❖

...Program finished with exit code 0
Press ENTER to exit console.
```

## Arreglos multidimensionales

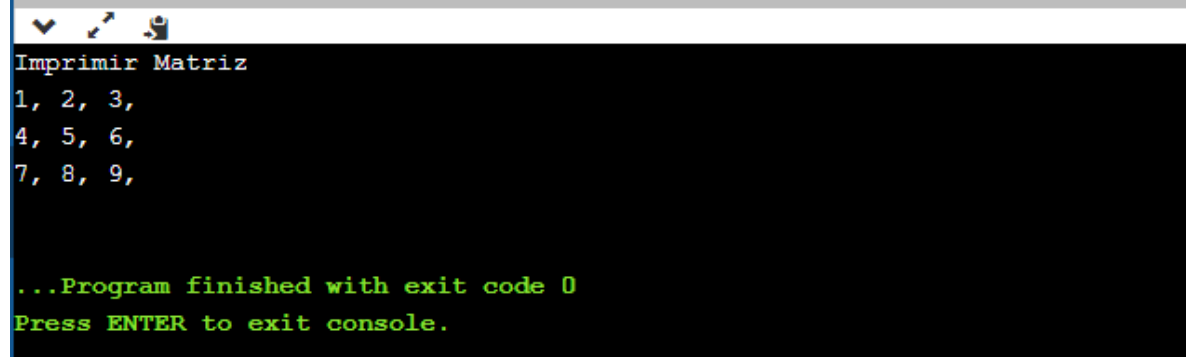
Como vimos anteriormente los arreglos no solo son de una dimensión si no que pueden ser multidimensionales como en el caso de matrices u otros objetos de estudio, siguiendo la siguiente sintaxis:

`tipoDato nombre[ tamaño ][ tamaño ]...[tamaño];`

La primera dimensión expresa a los renglones del arreglo, mientras que la segunda la de las columnas y la tercera de los planos si se observa geométricamente, y de la misma forma, se puede hacer uso de apuntadores dentro de ella.

Para ello se tienen los siguientes ejemplos de manera clara

```
1  #include<stdio.h>
2  /* Este programa genera un arreglo de dos dimensiones (arreglo
3  multidimensional) y accede a sus elementos a través de dos ciclos
4  for, uno anidado dentro de otro.
5  */
6  int main(){
7      int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
8      int i, j;
9      printf("Imprimir Matriz\n");
10     for (i=0 ; i<3 ; i++){
11         for (j=0 ; j<3 ; j++){
12             printf("%d, ",matriz[i][j]);
13         }
14         printf("\n");
15     }
16     return 0;
17 }
18
```



The screenshot shows the output of the C program. It displays the text "Imprimir Matriz" followed by a 3x3 matrix of numbers: 1, 2, 3, 4, 5, 6, 7, 8, 9. The output is formatted with commas and newlines. At the bottom, it shows "...Program finished with exit code 0" and "Press ENTER to exit console."



## Código (arreglos multidimensionales con punteros)

Como antes se mencionaba, en el siguiente código se hace uso de arreglos cuyos códigos cuentan con punteros incluidos, notemos las diferencias a la anterior:

```
1  #include<stdio.h>
2  /* Este programa genera un arreglo de dos dimensiones (arreglo
3  multidimensional) y accede a sus elementos a través de un puntero utilizando
4  un ciclo for.
5  */
6  int main(){
7      int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
8      int i, cont=0, *ap;
9      ap = matriz;
10     printf("Imprimir Matriz\n");
11     for (i=0 ; i<9 ; i++){
12         if (cont == 3){
13             printf("\n");
14             cont = 0;
15         }
16         printf("%d\t",*(ap+i));
17         cont++;
18     }
19     printf("\n");
20     return 0;
21 }
22
```

input

main.c:9:5: warning: assignment from incompatible pointer type [-Wincompatible-pointer-types]

Imprimir Matriz

1	2	3
4	5	6
7	8	9

...Program finished with exit code 0  
Press ENTER to exit console.

## Conclusiones

Gracias a esta práctica pude comprender la importancia de los arreglos tanto unidimensionales como bidimensionales para la generación de grupos de datos siempre y cuando sean del mismo tipo; igualmente aprendí a generar matrices con los mismos arreglos, los ciclos for fueron de mucha utilidad para emplear arreglos bidimensionales con datos.

Finalmente corregí algunos errores de sintaxis que frecuentemente realizaba como olvidar el punto y coma al final de las sentencias y el uso de coma en vez de punto y coma en los ciclos "for".

## Bibliografía

El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.

