



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

Marco Antonio Martinez Quintana

*Profesor:*

FUNDAMENTOS DE PROGRAMACION

*Asignatura:*

3

*Grupo:*

11

*No de Práctica(s):*

Andrik Uriel Reyes Roque

*Integrante(s):*

*No. de Equipo de  
cómputo empleado:*

No aplica

Fp03alu39

*No. de Lista o Brigada:*

1

*Semestre:*

11/01/2021

*Fecha de entrega:*

*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.

**Introducción**

Un arreglo almacena muchos elementos del mismo tipo, como 20 enteros, 50 números de coma flotante o 15 caracteres. El arreglo es muy trascendental por diversas razones, una operación muy importante es almacenar secuencias o cadenas de texto. Hasta el momento C proporciona datos de un solo carácter: utilizando el tipo arreglo, se puede crear una variable que contenga un grupo de caracteres.

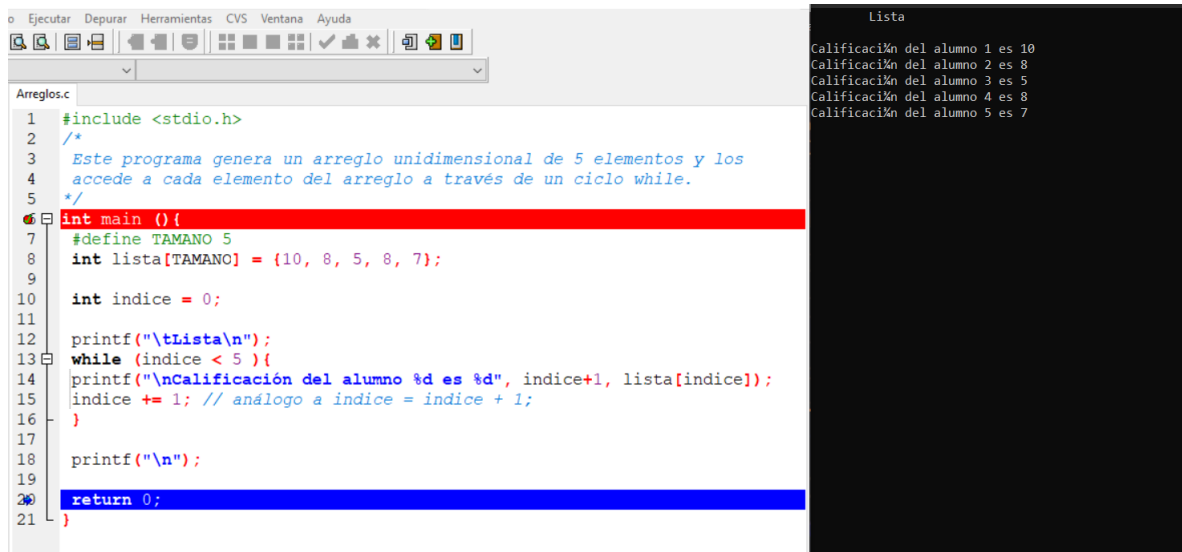
Un arreglo, lista o tabla es una secuencia de datos del mismo tipo. Los datos se llaman elementos del arreglo y se numeran consecutivamente 0. 1. 2. 3. etc. El tipo de elementos almacenados en el arreglo puede ser cualquier tipo de dato de C, incluyendo estructuras definidas por el usuario.

**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.

## Ejercicios propuestos

### Código (arreglo unidimensional while)



The screenshot shows a C program in a code editor. The program defines an array 'lista' of size 5 with values {10, 8, 5, 8, 7}. It uses a 'while' loop to iterate through the array, printing the index and the value. The output is shown in a terminal window on the right.

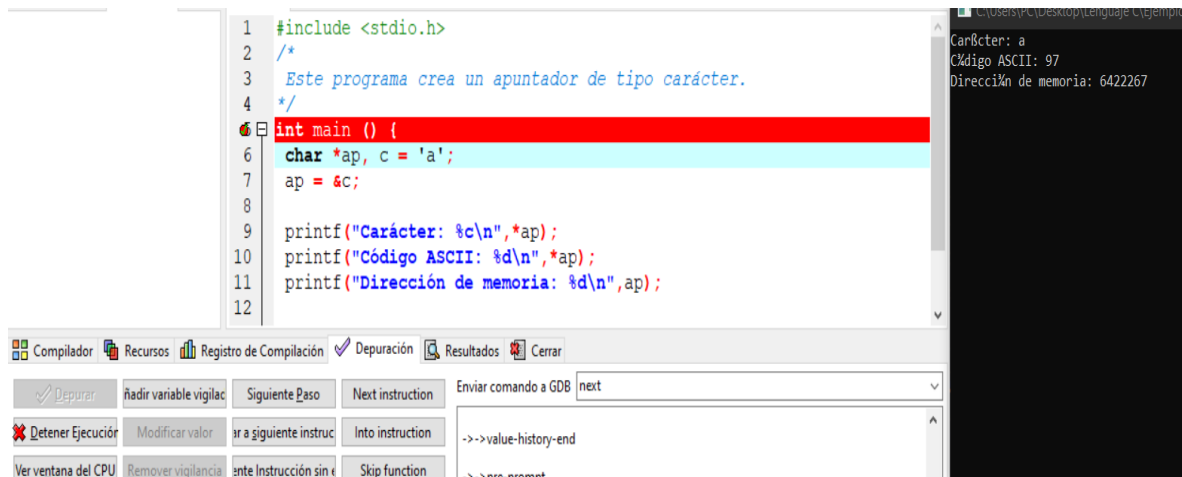
```
1 #include <stdio.h>
2 /*
3  Este programa genera un arreglo unidimensional de 5 elementos y los
4  accede a cada elemento del arreglo a través de un ciclo while.
5  */
6 int main () {
7     #define TAMANO 5
8     int lista[TAMANO] = {10, 8, 5, 8, 7};
9
10    int indice = 0;
11
12    printf("\tLista\n");
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

En la línea 7 indica el valor del “TAMANO”. Se definió en la línea 8 la lista en donde “TAMANO” iba a tener números almacenados, en la línea 10 “índice” tiene como valor 0 que será usado en el while donde el índice sea menor de 5 tendrá que repetirse hasta que llegue al 5, siendo así en la línea 14 va a ir sumando y poniendo el valor que le toque a cada índice el cual ira aumentando ya llegando al 5 el programa termina.

### Código (apuntadores)



The screenshot shows a C program in a code editor. The program declares a character variable 'c' with the value 'a' and a pointer 'ap' that points to 'c'. It then prints the character, its ASCII code, and its memory address. The output is shown in a terminal window on the right.

```
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 }
```

Carácter: a  
Código ASCII: 97  
Dirección de memoria: 6422267

En la línea 6 “\*ap” es el apuntador en donde contiene la dirección de la variable y en la línea 7 se le asigna la dirección de memoria de otra variable y para las líneas 9,10,11 va mostrando y sustituyendo los valores que se encuentran dentro.

## Código (apuntadores)

```
iglos.c
#include<stdio.h>
/*
Este programa accede a las localidades de memoria de distintas variables a
través de un apuntador.
*/
int main () {
    int a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0};
    int *apEnt;
    apEnt = &a;

    printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n");
    printf("apEnt = %a\n");

    b = *apEnt;
    printf("b = *apEnt \t-> b = %i\n", b);

    b = *apEnt + 1;
    printf("b = *apEnt + 1 \t-> b = %i\n", b);

    *apEnt = 0;
    printf("*apEnt = 0 \t-> a = %i\n", a);

    apEnt = &c[0];
    printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt);
}
```

```
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
```

En la línea 7 indica que dentro de las letras se contiene el número, pero en la letra “c” se da a entender que dentro habrá un total de 10 números contenidos, línea 8-9 sirven como apuntador. En la 11 y 12 solo muestra lo que se encuentra en el inicio.

Línea 14 `b = *apEnt` pero como `apEnt = &a` este muestra el número 5 que está guardado en `a` mas no en `b`, en la línea 17 indica la sumatoria de +1 y en la 18 imprime el resultado. Línea 20 señala que `apEnt` es igual a 0 por lo que en la línea 21 muestra que cambio a 0. En 23 y 24 muestra uno de los números contenidos dentro de “c”

## Código (apuntadores)

```
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
```

La letra “x” es una variable que se usa como sustituto de `*apArr` que es el apuntador del inicio el cual va con “arr”. En este caso en las líneas 15,18,21 como se empieza por el número

5 (debido a que no se especifica de donde empezar) y cada vez que suma se sigue con el orden establecido en “arr”.

### *Código (apuntadores en ciclo 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 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    printf("\n");
16    return 0;
17 }
18
19
```

```
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
C:\Users\PC\Desktop\Lenguaje C\Ejemplo>
```

Línea 8, 9 y 10 declaran que “TAMANO” , que contiene los números dentro y sirve como apuntador. En la 12 señala “índice” el cual empezara con un valor de cero, también indica que es menor a 5 y tendrá un aumento cada “++” cuando sea utilizado, esto se vera reflejado en la línea 13 donde muestra en pantalla la calificación de cada uno las cuales son ordenadas debido a lo que indica que va dentro de cada “%d”.

### *Código (apuntadores en cadenas)*

```
#include <stdio.h>
/*
 Este programa muestra el manejo de cadenas en lenguaje C.
 */
int main(){
    char palabra[20];
    int i=0;
    printf("Ingrese una palabra: ");
    scanf("%s", palabra);
    printf("La palabra ingresada es: %s\n", palabra);
    for (i = 0 ; i < 20 ; i++){
        printf("%c\n", palabra[i]);
    }
    return 0;
}
```

```
Ingrese una palabra: comida
La palabra ingresada es: comida
c
o
m
i
d
a
```

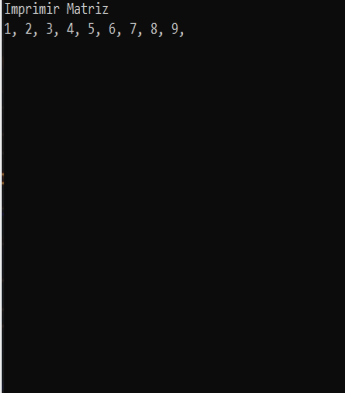
línea 6 el char sirve para guardar en este caso será una palabra la cual su máximo puede ser de 20 letras, línea 8 y 9 te piden una palabra la cual será registrada con el scan. En conjunto con la línea 7 y 11 se usó un for para ir haciendo un incremento de la variable “i” la cual con

la 12 estaría mostrando en pantalla letra por letra debido a lo que se realiza en la 11 don el for.

## Arreglos multidimensionales

### Código (arreglos multidimensionales)

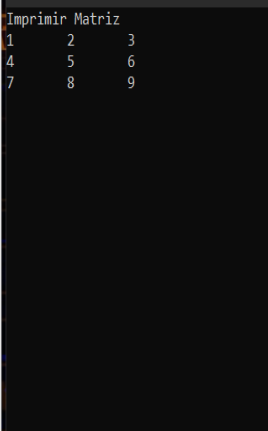
```
1 #include<stdio.h>
2 /* Este programa genera un arreglo de dos dimensiones (arreglo
3 multidimensional) y accede a sus elementos a través de dos ciclos
4 for, uno anidado dentro de otro.
5 */
6 int main(){
7     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
8     int i, j;
9     printf("Imprimir 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 }
```



Como explica en la practica el Lenguaje C te permite tener arreglos con varias dimensiones en donde la línea 7 contiene 2 después de la matriz conteniendo un orden de numeros. En la 8 se declaran “i” y “j” las cuales con for en 10 y 11 tendrá la función de en la línea 12 mostrar los valores de la Matriz ya que van incrementando.

### Código (arreglos multidimensionales con apuntadores)

```
1 int main(){
2     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
3     int i, cont=0, *ap;
4     ap = matriz;
5     printf("Imprimir Matriz\n");
6     for (i=0 ; i<9 ; i++){
7         if (cont == 3){
8             printf("\n");
9             cont = 0;
10        }
11        printf("%d\t",*(ap+i));
12        cont++;
13    }
14    printf("\n");
15    return 0;
16 }
```



Línea 7, 8 y 9 se usa un apuntador, se declaran variables y se define el tamaño de la dimensión, en la 11 el ciclo for hace que la variable “i” vaya incrementando. De la línea 12-19 hace una especie de repetición con ayuda del apuntador y del for debido a que van

añadiendo los números que se tenían en la matriz y el printf con \t los acomoda de tal manera que queden en orden

### **Conclusión:**

La practica estuvo un poco complicada ya que te puedes llegar a confundir en el estar buscando que variable se aloja cada termino y los valores que se le otorgan a dichas variables para que puedan hacer uso de lo que se contiene dentro de los arreglos. Observe la importancia el uso del apuntador si se usa de manera correcta puede ser de gran utilidad en conjunto con los arreglos multidimensionales ya que acortan y contienen dentro datos que puedes ir utilizando en menos líneas dando la posibilidad de tener un programa con un código más limpio.

### **Bibliografía**

Joyanes, Aguilar, Luis, and Martínez, Ignacio Zahonero. Programación en C, C++, Java y UML (2a. ed.), McGraw-Hill Interamericana, 2014. ProQuest Ebook Central, <https://ebookcentral.proquest.com/lib/bibliodgbmhe/detail.action?docID=3225314>.