



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:*

Marco Antonio Martinez Quintana

*Asignatura:*

Fundamentos de Programación

*Grupo:*

03

*No de Práctica(s):*

9

*Integrante(s):*

Lopez Martinez Diana

*No. de Equipo de  
cómputo empleado:*

*No. de Lista o Brigada:*

28

*Semestre:*

1

*Fecha de entrega:*

12/01/2021

*Observaciones:*

**CALIFICACIÓN:** \_\_\_\_\_

# Arreglos unidimensionales y multidimensionales

## 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 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.

## Código (arreglo unidimensional while)



\*arregloU: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
#include <stdio.h>

/*
    Este programa genera un arreglo unidimensional de 5 elementos y los
    accede a cada elemento del arreglo a través de un ciclo while.
*/

int main (){
    char ao= 162;

    #define TAMANO 5
    // Creamos la sintaxis del arreglo y especificamos sus valores
    int lista[TAMANO] = {10, 8, 5, 8, 7};

    // Especificamos la posicion del arreglo
    int indice = 0;

    // Imprimimos el titulo que tendra
    printf("\tlista\n");
    // Accede a cada elemento
    while (indice < 5 ){
        // Imprimimos un mensaje que nos dira el alumno y su calificacion
        printf("\nCalificaci%cn del alumno %d es %d", ao, indice+1, lista[indice]);
        indice += 1; // análogo a indice = indice + 1;
    }

    printf("\n");
    return 0;
}
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>gcc arregloU.c -o arregloU.exe
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>arregloU.exe
```

```
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
```

## Código (arreglo unidimensional for)

 \*uniFor: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
#include <stdio.h>
```

```
/*
```

```
Este programa genera un arreglo unidimensional de 5 elementos y  
accede a cada elemento del arreglo a través de un ciclo for.
```

```
*/
```

```
int main (){
```

```
    #define TAMANO 5
```

```
    // Declaramos el tamaño del arreglo y le damos valor a las variables
```

```
    int lista[TAMANO] = {10, 8, 5, 8, 7};
```

```
    // Escribimos un mensaje el cual va a ser el titulo
```

```
    printf("\tLista\n");
```

```
    // Utilizamos el ciclo for, indice va a comenzar desde la posicion 0 y tiene que ser menor a 5 tambien aumentara de 1 en 1
```

```
    for (int indice = 0 ; indice < 5 ; indice++){
```

```
        // Escribimos un mensaje que nos muestre el alumno y su calificacion
```

```
        printf("\nCalificaci%cn del alumno %d es %d", 162, indice+1, lista[indice]);
```

```
    }
```

```
    // Salto de linea
```

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

```
    return 0;
```

```
}
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>gcc uniFor.c -o uniFor.exe
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>uniFor.exe
```

```
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
```

## Código (apuntadores)



caracter: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
#include <stdio.h>
/*
    Este programa crea un apuntador de tipo carácter.
*/
int main () {
    // Inicializamos los valores con tipo de dato char
    char *ap, c = 'a', ad=160, ao=162;
    // la variable ap estara guardada en la direccion de memoria c
    ap = &c;
    // Mandamos el mensaje con el formato de caracter
    printf("Carácter: %c\n",ad,*ap);
    printf("Código ASCII: %d\n",ao,*ap);
    printf("Dirección de memoria: %d\n",ao,ap);

    return 0;
}
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>gcc caracter.c -o caracter.exe
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>caracter.exe
```

```
Carácter: a
```

```
Código ASCII: 97
```

```
Dirección de memoria: 6422295
```

## Código (apuntadores)



\*apB: Bloc de notas

Archivo Edición Formato Ver Ayuda

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

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

    // Calculamos las localidades |
    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);

    return 0;
}
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>gcc apB.c -o apB.exe
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>apB.exe
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
```

```
#include <stdio.h>
/*
    Este programa trabaja con aritmética de apuntadores para acceder a los
    valores de un arreglo.
*/
int main () {
    // Damos valores a los apuntadores
    int arr[] = {5, 4, 3, 2, 1};
    // Tipo de dato entero
    int *apArr;
    apArr = arr;

    printf("int arr[] = {5, 4, 3, 2, 1};\n");
    printf("apArr = &arr[0]\n");

    // Mandamos mensaje de los valores del arreglo
    int x = *apArr;
    printf("x = *apArr \t -> x = %d\n", x);

    x = *(apArr+1);
    printf("x = *(apArr+1) \t -> x = %d\n", x);

    x = *(apArr+2);
    printf("x = *(apArr+1) \t -> x = %d\n", x);

    return 0;
}
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>gcc apC.c -o apC.exe
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>apC.exe
```

```
int arr[] = {5, 4, 3, 2, 1};
```

```
apArr = &arr[0]
```

```
x = *apArr          -> x = 5
```

```
x = *(apArr+1)      -> x = 4
```

```
x = *(apArr+1)      -> x = 3
```

## Código (apuntadores en ciclo for)



aFor: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
#include <stdio.h>
/*
    Este programa genera un arreglo unidimensional de 5 elementos y
    accede a cada elemento del arreglo a través de un apuntador
    utilizando un ciclo for.
*/
int main (){
    #define TAMANO 5
    // creamos el arreglo y le damos valores
    int lista[TAMANO] = {10, 8, 5, 8, 7};
    int *ap = lista;
    printf("\tLista\n");
    // la posicion empieza desde cero menos a 5 y va de 1 en 1
    for (int indice = 0 ; indice < 5 ; indice++){
        // Mandamos mensaje que nos diga el alumno y su la calificacion
        printf("\nCalificaci%cn del alumno %d es %d", 162, indice+1, *(ap+indice));
    }
    // salto de linea
    printf("\n");

    return 0;
}
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>gcc aFor.c -o aFor.exe
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>aFor.exe
```

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



## Código (apuntadores en cadenas)



\*aP: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
#include <stdio.h>
/*
    Este programa muestra el manejo de cadenas en lenguaje C.
*/
int main(){
    // inicializo palabras
    char palabra[20];
    int i=0;

    // mandamos un mensaje donde pedimos el dato de la palabra
    printf("Ingrese una palabra: ");
    // Lo guardamos en el arreglo
    scanf("%s", palabra);
    printf("La palabra ingresada es: %s\n", palabra);

    // la posicion inicia en cero es menor a 20 y se va sumando de 1 en 1
    for (i = 0 ; i < 20 ; i++){

        printf("%c\n", palabra[i]);
    }
    return 0;
}
```

```
Ingrese una palabra: programacion
La palabra ingresada es: programacion
```

```
p
r
o
g
r
a
m
a
c
i
o
n
```

```
á
6
```

## *Código (arreglos multidimensionales)*



matriz: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
#include<stdio.h>
```

```
/* Este programa genera un arreglo de dos dimensiones (arreglo multidimensional) y accede a sus elementos a través de dos ciclos for, uno anidado dentro de otro. */
```

```
int main(){
    // Sera una matriz con 3 filas y 3 columnas
    int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};

    //Inicializamos las variables
    int i, j;

    printf("Imprimir Matriz\n");

    // la posicion inicia en cero es menos a 3 y aumentara de 1 en 1
    for (i=0 ; i<3 ; i++){
        for (j=0 ; j<3 ; j++){
            printf("%d, ",matriz[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>gcc matriz.c -o matriz.exe
```

```
C:\Users\paust\Downloads\Lenguaje c\Ejemplos>matriz.exe
```

```
Imprimir Matriz
```

```
1, 2, 3,
```

```
4, 5, 6,
```

```
7, 8, 9,
```

# Conclusión

Los arreglos nos permiten realizar cualquier programa más eficiente, nosotros podemos almacenar datos en una posición específica con valores específicos estos son almacenados en cada arreglo.

## Bibliografía

Solano Gálvez, García Cano, Sandoval Montaña, Nakayama Cervantes, Arteaga Ricci, Castañeda Perdomo, I., 2020. Laboratorio Salas A Y B. [online] Lcp02.fi-b.unam.mx. Available at: <<http://lcp02.fi-b.unam.mx/>> [Accessed 6 April 2020].