
	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	1/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Guía práctica de estudio 09: Arreglos unidimensionales



Elaborado por	Actualizado por:	Revisado por:
M.C. Edgar E. García Cano Ing. Jorge A. Solano Gálvez	M.C. Cintia Quezada Reyes Ing. María Guadalupe Morales Nava	M.C. Laura Sandoval Montaño

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	2/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Guía práctica de estudio 09: Arreglos unidimensionales

Objetivo:

El alumno utilizará arreglos de una dimensión en la elaboración de programas que resuelvan problemas que requieran agrupar datos del mismo tipo, alineados en un vector o lista.

Actividades:

- Elaborar programas en lenguaje C que empleen arreglos de una dimensión.
- Manipular este tipo de arreglos a través de índices.


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. La dimensión del arreglo va de acuerdo con el número de índices que se requiere emplear para acceder a un elemento del arreglo. Así, si se requiere ubicar a un elemento en un arreglo de una dimensión (unidimensional), se requiere de un índice, para un arreglo de dos dimensiones se requieren dos índices y así sucesivamente. Los arreglos se utilizan para hacer más eficiente el código de un programa, así como manipular datos del mismo tipo con un significado común.

En esta práctica nos enfocaremos a trabajar con los arreglos unidimensionales.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	3/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			


Licencia GPL de GNU

El software presente en esta práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.

```

/*
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 *
 * Authors: María Guadalupe Morales, Cintia Quezada and Jorge A. Solano
 */

```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	4/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Arreglos unidimensionales

Un arreglo unidimensional de n elementos se almacena en la memoria de la siguiente manera (Figura 1):

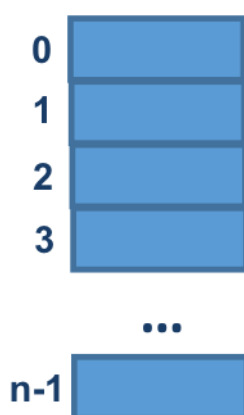


Figura 1. Almacenamiento en memoria de un arreglo unidimensional

La primera localidad del arreglo corresponde al índice 0 y la última corresponde al índice $n-1$, donde n es el tamaño del arreglo.

La sintaxis para definir un arreglo unidimensional en lenguaje C es la siguiente:


`tipoDeDato nombre[tamaño]`

Donde *nombre* se refiere al identificador del arreglo, *tamaño* es un número entero y define el número máximo de elementos que puede contener el arreglo. El *tipoDeDato* es el tipo de dato de los elementos del arreglo, el cual puede ser tipo entero, real, carácter o estructura.

NOTA: El tipo de datos *estructura* no se abordará en esta práctica.

Código (arreglo unidimensional empleando la estructura while)

A continuación, se muestra el código de un programa que genera un arreglo unidimensional de 5 elementos y que para poder acceder, recorrer y mostrar cada elemento del arreglo se usa la variable *indice* desde 0 hasta 4 haciendo uso de un ciclo *while*.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	5/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Programa1a.c

```
#include <stdio.h>
int main ()
{
    int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional
    int indice = 0;
    printf("\tLista\n");
    while (indice < 5 ) // Acceso a cada elemento del arreglo unidimensional usando while
    {
        printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
        indice += 1;      // Sentencia análoga a indice = indice + 1;
    }


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

Código (arreglo unidimensional empleando la estructura do-while)

En el siguiente código se genera un arreglo unidimensional de 5 elementos y que para poder acceder, recorrer y mostrar cada elemento del arreglo se usa la variable *indice* desde 0 hasta 4 haciendo uso de un ciclo *do-while*.

Programa1b.c

```
#include <stdio.h>
int main ()
{
    int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional
    int indice = 0;
    printf("\tLista\n");
    do // Acceso a cada elemento del arreglo unidimensional usando do-while
    {
        printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
        indice += 1;      // Sentencia análoga a indice = indice + 1;
    }
    while (indice < 5 );
    printf("\n");
    return 0;
}
```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	6/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código (arreglo unidimensional empleando la estructura for)

A continuación, se muestra el código que genera un arreglo unidimensional de 5 elementos y que para poder acceder, recorrer y mostrar cada elemento del arreglo se usa la variable *indice* desde 0 hasta 4 haciendo uso de un ciclo *for*.


Programa1c.c

```
#include <stdio.h>
int main ()
{
    int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional
    int indice=0;
    printf("\tLista\n");
    // Acceso a cada elemento del arreglo unidimensional usando for
    for (indice = 0 ; indice < 5 ; indice++)
    {
        printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
    }

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

Código (arreglo unidimensional empleando la estructura for)

A continuación, se muestra un programa que genera un arreglo unidimensional de máximo 10 elementos. Para poder leer y almacenar datos en cada elemento y posteriormente mostrar el contenido de estos elementos se hace uso de un ciclo *for* respectivamente.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	7/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Programa2.c


```
#include <stdio.h>
int main ()
{
    int lista[10]; // Se declara el arreglo unidimensional
    int indice=0;
    int numeroElementos=0;
    printf("\nDa un número entre 1 y 10 para indicar la cantidad de elementos que tiene el arreglo\n");
    scanf("%d",&numeroElementos);
    if((numeroElementos>=1) && (numeroElementos<=10))
    {
        // Se almacena un número en cada elemento del arreglo unidimensional usando for
        for (indice = 0 ; indice <= numeroElementos-1 ; indice++)
        {
            printf("\nDar un número entero para el elemento %d del arreglo", indice );
            scanf("%d",&lista[indice]);
        }
        printf("\nLos valores dados son: \n");
        // Se muestra el número almacenado en cada elemento del arreglo unidimensional usando for
        for (indice = 0 ; indice <= numeroElementos-1 ; indice++)
        {
            printf("%d  ", lista[indice] );
        }
    }
    else printf("el valor dado no es válido");
    printf("\n");
    return 0;
}
```

El código anterior también puede realizarse utilizando los ciclos while y do-while; se invita al lector a que pruebe la implementación de ambos casos.

Apuntadores

Un apuntador es una variable que contiene la dirección de una variable, es decir, hace referencia a la localidad de memoria de otra variable. Debido a que los apuntadores trabajan directamente con la memoria, a través de ellos se accede con rapidez a un dato.

La sintaxis para declarar un apuntador y para asignarle la dirección de memoria de otra variable es, respectivamente:

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	8/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

TipoDeDato *apuntador, variable;
apuntador = &variable;

La declaración de una variable apuntador inicia con el carácter *. Cuando a una variable le antecede un ampersand (&), lo que se hace es referirse a la dirección de memoria donde se ubica el valor de dicha variable (es lo que pasa cuando se lee un dato con scanf).

Los apuntadores solo pueden apuntar a direcciones de memoria del mismo tipo de dato con el que fueron declarados; para referirse al contenido de dicha dirección, a la variable apuntador se le antepone * (Figura 2).

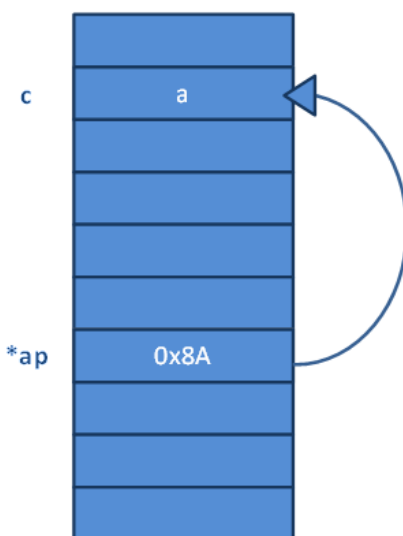



Figura 2. Un apuntador almacena la dirección de memoria de la variable a la que apunta

Código (apuntadores)

En el siguiente código se aprecia la declaración de un apuntador de tipo carácter y se muestra en pantalla, haciendo uso de apuntadores, el contenido de la variable *c*, así como el código ASCII del carácter 'a' que tiene almacenado dicha variable.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	9/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Programa3.c

```
#include <stdio.h>
int main ()
{
    char *ap, c = 'a'; // Se declara el apuntador ap de tipo alfanumérico
    ap = &c; //Se le asigna al apuntador la dirección de memoria de la variable c
    printf("Carácter: %c\n",*ap); /* Se imprime el contenido de la variable a la
                                que apunta el apuntador ap */
    printf("Código ASCII: %d\n",*ap); /*Se imprime el código ASCII del carácter
                                'a' */
    printf("Dirección de memoria: %d\n",ap);/*Se imprime la dirección de
                                memoria que almacena el apuntador*/
    return 0;
}
```


Código (apuntadores)

Enseguida se observa el código de un programa que permite acceder a las localidades de memoria de distintas variables a través de un apuntador.

Programa4.c

```
#include<stdio.h>
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");
    /*A la variable b se le asigna el contenido de la variable a la que apunta
    apEnt*/
    b = *apEnt;
    printf("b = *apEnt \t-> b = %i\n", b);
    /*A la variable b se le asigna el contenido de la variable a la que apunta
    apEnt y se le suma uno*/
    b = *apEnt +1;
    printf("b = *apEnt + 1 \t-> b = %i\n", b);
    //La variable a la que apunta apEnt se le asigna el valor cero
    *apEnt = 0;
    printf("*apEnt = 0 \t-> a = %i\n", a);
    /*A apEnt se le asigna la dirección de memoria que tiene el elemento 0 del
    arreglo c*/

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

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	10/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Apuntadores y su relación con los arreglos.

Cabe mencionar que el nombre de un arreglo es un apuntador fijo al primero de sus elementos; por lo que las siguientes instrucciones, para el código de arriba, son equivalentes:


```
apEnt = &c[0];
apEnt = c;
```

Código (apuntadores)

El programa que se observa a continuación trabaja con aritmética de apuntadores para acceder a todos los valores que se encuentran almacenados en cada uno de los elementos de un arreglo.

Programa5.c

```
#include <stdio.h>
int main ()
{
    int arr[] = {5, 4, 3, 2, 1};
    int *apArr; //Se declara el apuntador apArr
    int x;
    apArr = arr;
    printf("int arr[] = {5, 4, 3, 2, 1};\n");
    printf("apArr = &arr[0]\n");
    x = *apArr; /*A la variable x se le asigna el contenido del arreglo arr en su
    elemento 0*/
    printf("x = *apArr \t -> x = %d\n", x);
    x = *(apArr+1); /*A la variable x se le asigna el contenido del arreglo arr
    en su elemento 1*/
    printf("x = *(apArr+1) \t -> x = %d\n", x);
    x = *(apArr+2); /*A la variable x se le asigna el contenido del arreglo arr
    en su elemento 2*/
    printf("x = *(apArr+2) \t -> x = %d\n", x);
    x = *(apArr+3); /*A la variable x se le asigna el contenido del arreglo arr
    en su elemento 3*/
    printf("x = *(apArr+3) \t -> x = %d\n", x);
    x = *(apArr+4); /*A la variable x se le asigna el contenido del arreglo arr
    en su elemento 4*/
    printf("x = *(apArr+2) \t -> x = %d\n", x);
    return 0;
}
```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	11/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código (apuntadores en ciclo for)


El siguiente programa genera un arreglo unidimensional de 5 elementos y accede a cada uno de los elementos del arreglo haciendo uso de un apuntador, para ello se utiliza un ciclo *for*.

Programa6a.c

```
#include <stdio.h>
int main ()
{
    int lista[5] = {10, 8, 5, 8, 7};
    int *ap = lista; //Se declara el apuntador ap
    int indice;
    printf("\tLista\n");
    //Se accede a cada elemento del arreglo haciendo uso del ciclo for
    for (indice = 0 ; indice < 5 ; indice++)
    {
        printf("\nCalificación del alumno %d es %d", indice+1, *(ap+indice));
    }
    printf("\n");
    return 0;
}
```

Código (apuntadores en ciclo while)

A continuación, se muestra el código de un programa que genera un arreglo unidimensional de 5 elementos y accede a cada elemento del arreglo a través de un apuntador utilizando un ciclo *while*.

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	12/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Programa6b.c

```
#include <stdio.h>
int main ()
{
    int lista[5] = {10, 8, 5, 8, 7};
    int *ap = lista; //Se declara el apuntador ap
    int indice = 0;


    printf("\tLista\n");
    //Se accede a cada elemento del arreglo haciendo uso del ciclo while
    while (indice < 5);
    {
        printf("\nCalificación del alumno %d es %d", indice+1, *(ap+indice));
        indice++;
    }
    printf("\n");
    return 0;
}
```

Código (apuntadores en ciclo do-while)

El programa que a continuación se observa genera un arreglo unidimensional de 5 elementos y accede a cada elemento del arreglo a través de un apuntador utilizando un ciclo *do while*.

Programa6c.c

```
#include <stdio.h>
int main ()
{
    int lista[5] = {10, 8, 5, 8, 7};
    int *ap = lista; //Se declara el apuntador ap
    int indice = 0;
    printf("\tLista\n");
    //Se accede a cada elemento del arreglo haciendo uso del ciclo do-while
    do
    {
        printf("\nCalificación del alumno %d es %d", indice+1, *(ap+indice));
        indice++;
    }
    while (indice < 5)
    printf("\n");
    return 0;
}
```


	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	13/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Código (apuntadores en cadenas)

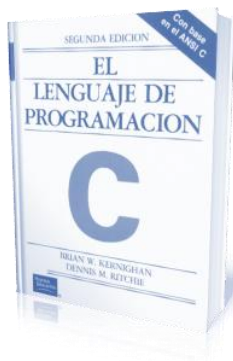
El siguiente programa muestra el manejo básico de cadenas en lenguaje C.

Programa7.c

```
#include <stdio.h>
int main()
{
    char palabra[20];
    int i=0;
    printf("Ingrese una palabra: ");
    scanf("%s", palabra); /* Se omite & porque el propio nombre del arreglo de
    tipo cadena apunta, es decir, es equivalente a la dirección de comienzo del
    propio arreglo*/
    printf("La palabra ingresada es: %s\n", palabra);
    for (i = 0 ; i < 20 ; i++)
    {
        printf("%c\n", palabra[i]);
    }
    return 0;
}
```

	Manual de prácticas del Laboratorio de Fundamentos de programación	Código:	MADO-17
		Versión:	03
		Página	14/14
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Bibliografía



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