	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor(a): Karina García Morales

Asignatura: Fundamentos de Programación

Grupo: 22

No de Práctica(s): 09

Integrante(s): Edilberto Vicente Martínez

No. De lista o brigada: 50

Semestre: 2026-1

Fecha de entrega: 04 de noviembre de 2025

Observaciones:

CALIFICACIÓN: _____

Práctica 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.

Desarrollo:

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.

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 tamaño del arreglo el cual es asignado por el usuario. La sintaxis para definir al arreglo en lenguaje c es la siguiente:

tipoDeDato nombre[tamaño]

En donde:

Nombre = Es el identificador del arreglo.

Tamaño = Es un numero entero y es el que define la cantidad de elementos (tamaño) del arreglo.

Tipodedato = Es el tipo que puede haber dentro del arreglo, puede ser entero, real, carácter o una estructura.

Apuntadores

El apuntador es una variable que almacena la dirección de una variable, es decir, hace referencia a la localidad de memoria de otra variable. Gracias a que los apuntadores trabajan directamente con la memoria, es más sencillo trabajar con datos.

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

TipoDeDato *apuntador, variable; apuntador = &variable;

Para declarar una variable apuntador, primero se inicia con el carácter *. Una variable al estar acompañada con el ampersand (&) hace referencia a la dirección de la memoria donde se ubica la 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).

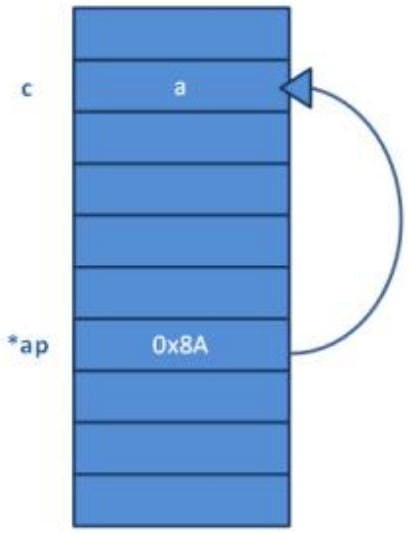

Figura 2. Un apuntador almacena la dirección de memoria de la variable a la que apunta

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
```

1.¿Qué es un arreglo?

Esuna secuencia de datos que va almacenada en un orden el cual va desde el índice 0 hasta el índice n-1, en donde el valor de n es un valor que define el tamaño del arreglo y el cual este se encuentra definido por el usuario.

2. ¿Qué es un apuntador?

Es una variable que almacena la dirección de memoria de otra variable del mismo tipo.

3. Ejecutamos programa 1a.c y lo modificamos:

Se revisa el cambio del primer programa con do-while y for

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>
int main ()
{
 int *apuntador,lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional
 apuntador=lista;
 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,*(apuntador+indice));
 indice += 1; // Sentencia análoga a indice = indice + 1;
 }

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

input

```
        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
```

4. ¿Que hace el programa 2.c?

Primeramente, muestra una pantalla donde tienes que escoger un numero entre 1 y 10 para indicar la cantidad de elemtos del arreglo (tamaño del arreglo), después de esto dependiendo de el numero ingresado te pide el valor de un numero entero cualquiera el cual va a ir dentro de nuestro arreglo, esto se repetira las veces que este definido el tamaño del arreglo. Terminado esto, mostrará una pantalla con nuestro arreglo ya hecho con los numeros proporcionados por el usuario.

5. Al programa2.c le vamos a aplicar apuntadores

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>
int main ()
{

    int *apuntador,lista[10];
    apuntador=lista;
    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))
    {
        for (indice = 0 ; indice <= numeroElementos-1 ; indice++)
    {
        printf("\nDar un número entero para el elemento %d del arreglo ", indice );
        scanf("%d",&*(apuntador+indice));
    }
        printf("\nLos valores dados son: \n");
        for (indice = 0 ; indice <= numeroElementos-1 ; indice++)
    {
        printf("%d ",*(apuntador+indice));
    }
    }
    else printf("el valor dado no es válido");
    printf("\n");
return 0;
}
```

input

```
3

Dar un número entero para el elemento 0 del arreglo 6

Dar un número entero para el elemento 1 del arreglo 9

Dar un número entero para el elemento 2 del arreglo 4

Los valores dados son:
6 9 4
```

*Arreglo 2.c con apuntadores.

6. Describe la sintaxis del arreglo

tipo_dato nombre_arreglo[numero_elementos];

7. Dar un ejemplo de arreglo unidimensional(explicito o implicito)

tipo nombre_arreglo[tamaño]; Explicito

tipo nombre_arreglo[] Implicito

8. ¿Qué estructura requieres para manipular un arreglo por medio de sus índices?

Las estructuras iterativas (while, do while y for).

9. Ejecución de programa 4.c con modificaciones

main.c

```c
//Edilberto Vicente Martínez
#include<stdio.h>
int main ()
{
 int a = 5, b = 10, c[10] = {300, 4, 3, 2, 1, 9, 8, 7, 6, 80};// cambiamos dos valores
 int *apEnt;
 apEnt = &a;
 printf("a = 5, b = 10, c[10] = {300, 4, 3, 2, 1, 9, 8, 7, 6, 80}\n");// cambiamos dos valores
 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);
 a=*apEnt;//cambiamos el valor final de a
 printf("a = %i\n", a);//comprobamos el valor final de a
 return 0;
}
```

input

```
a = 5, b = 10, c[10] = {300, 4, 3, 2, 1, 9, 8, 7, 6, 80}
apEnt = &a
b = *apEnt      -> b = 5
b = *apEnt + 1  -> b = 6
*apEnt = 0      -> a = 0
apEnt = &c[0]   -> apEnt = 300
a = 300
```

*El programa muestra el contenido de la variable, la variable apEnt y la dirección de memoria.

10. Programa5.c que indica como trabaja un apuntador recorriendo un arreglo

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>
int main ()
{
      int arr[] = {75, 4, 3, 2, 1};//tamaño 5 de forma implicita porque esta incializado
      int *apArr;//Se declara el apuntador apArr
      int x;
      apArr = arr; //apArr = &arr[0];
      printf("int arr[] = {75, 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*/
```

input

```
int arr[] = {75, 4, 3, 2, 1};
apArr = &arr[0]
x = *apArr        -> x = 75
x = *(apArr+1)    -> x = 4
x = *(apArr+2)    -> x = 3
x = *(apArr+3)    -> x = 2
x = *(apArr+4)    -> x = 1
```

*Recorre un arreglo usando un apuntador (indica como trabaja un apuntador recorriend un arreglo) solamente en el último x cambiamos el aptr+2 por aptr+4.

11. Programa7.c (aplicamos el uso de Strlen)

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>
#include <string.h>
int main()
{
 char palabra[20]; //arreglo tamaño 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);// si fuera caracter sería %c
 // este for me sirve para imprimir la palabra que ingresó el usuario
 for (i = 0 ; i<strlen(palabra) ; i++) // i < 20      i<=strlen(palabra-1)   (n-1)
  {
    printf("%c", palabra[i]); // %c porque va a recorrer letra por letra
  }
 return 0;
}

//    &lista[indice]  =>   &*(apuntador+indice)  implementación de apuntadores
```

input

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

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

El uso de strlen ayuda a leer una cadena de carácter, en este caso, nos permite establecer como condición a nuestra palabra en este caso de que i sea menor a la palabra ingresada.

Tarea:

1.-Indica que realiza el siguiente programa:

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>

int main() {
    int arreglo[] = {16, 32, 42, 67, 25, 20, 73, 28, 17, 45};
    int i, j, n, aux;

    n = 10;
    for (i = 1; i < n; i++)
    {
        j = i;
        aux = arreglo[i];
        while (j > 0 && aux < arreglo[j - 1])
        {
            arreglo[j] = arreglo[j - 1];
            j = j - 1;
        }
        arreglo[j] = aux;
    }
    printf("\n\nLos elementos obtenidos del arreglo son: \n");
    for (i = 0; i < n; i++) // <-- FIX: Use i < n
```

input

```
Elemento [0]: 16
Elemento [1]: 17
Elemento [2]: 20
Elemento [3]: 25
Elemento [4]: 28
Elemento [5]: 32
Elemento [6]: 42
```

*Muestra cada elemento de arreglo mediante saltos de línea mostrando su posición en el arreglo de cada uno.

2.- Genera un programa que le solicite una cadena de letras y números al usuario(emplear arreglo) e imprima solo letras. Este arreglo debe ser de caracteres, recuerda que un caracter puede almacenar números pero un arreglo de números no puede almacenar caracteres.

Análisis:

DE: Imprimir solo las letras de un arreglo con una combinación de letras y números.

RE: Imprimir solo letras sin números, excluir(0,1,2,3,4,5,6,7,8,9).

DS: Obtener la impresión de las letras.

Ejemplo:

Entrada de usuario= Ej3mpl0c4d3n4

Salida en pantalla= Ejmplcdn

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>
#include <ctype.h>
int main() {
    char entradaUsuario[100];
    int i;
    printf("Ingresa una cadena de letras y numeros: ");
    fgets(entradaUsuario, 100, stdin);
    printf("Salida en pantalla= ");
    for (i = 0; entradaUsuario[i] != '\0'; i++)
    {
        if (isalpha(entradaUsuario[i]))
        {
            printf("%c", entradaUsuario[i]);
        }
    }
    printf("\n");
    return 0;
}

```

input

```
Ingresa una cadena de letras y numeros: vic3ent4e932
Salida en pantalla= vicente


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

*Programa en funcionamiento.

3.- Genera un programa que solicite al usuario un vector de 15 enteros haciendo uso de un arreglo y la estructura iterativa para recorrerlo por medio de sus índices e imprimir en pantalla.

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>
#define TAMANO 15

int main() {

    int vector[TAMANO];
    int i;
    printf("--- Ingresa %d numeros enteros ---\n", TAMANO);
    for (i = 0; i < TAMANO; i++)
    {
        printf("Ingresa el valor para el indice [%d]: ", i);
        scanf("%d", &vector[i]);
    }
    printf("\n--- Mostrando los numeros ingresados ---\n");
    for (i = 0; i < TAMANO; i++) {
        printf("Elemento en indice [%d]: %d\n", i, vector[i]);
    }
    return 0;
}
```

input

```
--- Mostrando los numeros ingresados ---
Elemento en indice [0]: 4
Elemento en indice [1]: 7
Elemento en indice [2]: 4
Elemento en indice [3]: 2
Elemento en indice [4]: 9
Elemento en indice [5]: 10
```

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>
#define TAMANO 15

int main() {

    int vector[TAMANO];
    int i;
    printf("--- Ingresa %d numeros enteros ---\n", TAMANO);
    for (i = 0; i < TAMANO; i++)
    {
        printf("Ingresa el valor para el indice [%d]: ", i);
        scanf("%d", &vector[i]);
    }
    printf("\n--- Mostrando los numeros ingresados ---\n");
    for (i = 0; i < TAMANO; i++) {
        printf("Elemento en indice [%d]: %d\n", i, vector[i]);
    }
    return 0;
}
```

input

```
Elemento en indice [8]: 43
Elemento en indice [9]: 15
Elemento en indice [10]: 38
Elemento en indice [11]: 20
Elemento en indice [12]: 62
Elemento en indice [13]: 30
Elemento en indice [14]: 24
```

*Vista de los dos resultados del programa, muestra cada elemento del arreglo y su posición en el índice.

4.- Modifica el programa (Programa7.c) y aplicar el uso de apuntadores.

main.c

```c
//Edilberto Vicente Martínez
#include <stdio.h>
#include <string.h>
int main()
{
 char palabra[20]; //arreglo tamaño 20
 char *ptr;
    printf("Ingrese una palabra: ");
    scanf("%s", palabra);
    printf("La palabra ingresada es: %s\n", palabra);
    printf("Imprimiendo con apuntadores: ");
 for (ptr = palabra; *ptr != '\0'; ptr++)
  {
    printf("%c", *ptr);
  }
  printf("\n");
 return 0;
}
```

input

```
Ingrese una palabra: Programación
La palabra ingresada es: Programación
Imprimiendo con apuntadores: Programación
```

*Resultado del programa con la palabra programación.

Conclusión

En esta práctica trabajamos con los arreglos unidimensionales primeramente conociendo sus conceptos y después empleándolos en las tres estructuras iterativas (while, do while y for). En lo personal esta práctica me ayudo a recordar como pasar pasar de while a do while y de este a for, al principio si era un poco confuso, pero gracias a la explicación de la profesora pude recordar dichos pasos. También vi como se emplea el arreglo unidimensional explícito e implícito.

Bibliografía

*Facultad de Ingeniería. (2025). Manual de prácticas del laboratorio de Fundamentos de Programación. Laboratorio de computación. Salas A y B. Universidad Nacional Autónoma de México. pp. 135-148. Recuperado el 04 de noviembre 2025 de http://lcp02.fi-b.unam.mx/

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