



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: César Fabián Domínguez Velasco

Asignatura: Fundamentos de Programación

No. de práctica(s): 09

Integrante(s): 13_García_Sánchez_Alejandro
18_Lopez_Castro_Anastacia
34_Ramirez_Rivas_Gael
39_Ruiz_Hernandez_Ruben_Antonio

No. de lista o brigada: 1A

Semestre: 2024-2

Fecha de entrega: 18 de Abril del 2024

Observaciones:

CALIFICACIÓN: _____

PRÁCTICA 09: ARREGLOS UNIDIMENSIONALES

1) 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.

2) Introducción:

Un arreglo es un conjunto de datos contiguos del mismo tipo con un tamaño fijo definido al momento de crearse, ya que a cada elemento (dato) del arreglo se le asocia una posición particular, el cual se requiere indicar para acceder a un elemento en específico, de igual manera esto se logra a través del uso de índices, por ende, los arreglos pueden ser unidimensionales o multidimensionales. En cambio, la dimensión del arreglo va de acuerdo con el número de índices que se requiere emplear para acceder a un elemento del arreglo, así que sí, 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. A continuación, en esta práctica nos enfocaremos a trabajar con los arreglos unidimensionales.

3) Desarrollo (Capturas de pantalla de los programas en C):

Programa 1a.c

<pre>1 #include <stdio.h> 2 int main () 3 { 4 int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional 5 int indice = 0; 6 printf("\tLista\n"); 7 while (indice < 5) // Acceso a cada elemento del arreglo unidimensional usando while 8 { 9 printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]); 10 indice += 1; // Sentencia análoga a indice = indice + 1; 11 } 12 13 printf("\n"); 14 return 0; 15 16 } 17</pre>	<pre>/tmp/HkplCn1fEg.o 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 === Code Execution Successful ===</pre>
---	--



Programa 1b.c

1 #include <stdio.h>	/tmp/mNjEMi4mh2.o
2 int main ()	Lista
3 {	
4 int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional	Calificación del alumno 1 es 10
5 int indice = 0;	Calificación del alumno 2 es 8
6 printf("\tLista\n");	Calificación del alumno 3 es 5
7 do // Acceso a cada elemento del arreglo unidimensional usando do-while	Calificación del alumno 4 es 8
8 {	Calificación del alumno 5 es 7
9 printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);	
10 indice += 1;	
11 }	
12 while (indice < 5); printf("\n");	=== Code Execution Successful ===
13 return 0;	
14 }	



Programa 1c.c

1 #include <stdio.h>	/tmp/WPn5btQUUD.o
2 int main () {	Lista
3 int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional	
4 int indice=0;	Calificación del alumno 1 es 10
5 printf("\tLista\n");	Calificación del alumno 2 es 8
6 // Acceso a cada elemento del arreglo unidimensional usando for	Calificación del alumno 3 es 5
7 for (indice = 0 ; indice < 5 ; indice++) {	Calificación del alumno 4 es 8
8 printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]); }	Calificación del alumno 5 es 7
9 printf("\n");	
10 return 0; }	
11 }	=== Code Execution Successful ===



Programa 2.c

<div> <div> Welcome </div> <div> C programa2a.c </div> <div> C programa2.c </div> <div> C programa1c.c </div> </div> <div> <div>C programa2.c</div> <pre> 1 //Programa 2// 2 3 #include <stdio.h> 4 5 int main () 6 { 7 8 9 printf("\nEste programa permite realizar un arreglo unidimensional de un número entre 1 y 10.\n"); 10 11 int lista[10]; // Se declara el arreglo unidimensional 12 int indice = 0; 13 int numeroElementos = 0; 14 15 printf("\nDa un número entre 1 y 10 para indicar la cantidad de elementos que tiene el arreglo\n"); 16 scanf("%d",&numeroElementos); 17 18 if((numeroElementos >= 1) && (numeroElementos <= 10)) 19 { 20 // Se almacena un número en cada elemento del arreglo unidimensional usando for 21 22 for (indice = 0 ; indice <= numeroElementos-1 ; indice++) 23 { 24 printf("\nDar un número entero para el elemento %d del arreglo", indice); 25 scanf("%d",&lista[indice]); 26 } 27 28 printf("\nLos valores dados son: \n"); 29 30 // Se muestra el número almacenado en cada elemento del arreglo unidimensional usando for 31 32 for (indice = 0; indice <= numeroElementos - 1; indice++) 33 { 34 printf("%d ", lista[indice]); 35 } 36 37 else printf("el valor dado no es válido"); 38 39 printf("\n"); 40 41 return 0; 42 } 43 44 }</pre> </div>

Last login: Fri Apr 12 16:04:19 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit <https://support.apple.com/kb/HT208050>.

Andorra04:~ fp15alu39\$ cd Documents/

Andorra04:Documents fp15alu39\$ ls

programa1c programa2 programa2.c
programa1c.c programa2. programa2a.c

Andorra04:Documents fp15alu39\$ gcc programa2.c -o programa2

Andorra04:Documents fp15alu39\$./programa2

Este programa permite realizar un arreglo unidimensional de un número entre 1 y 10.,

Da un número entre 1 y 10 para indicar la cantidad de elementos que tiene el arreglo
5

Dar un número entero para el elemento 0 del arreglo: 15

Dar un número entero para el elemento 1 del arreglo

Dar un número entero para el elemento 2 del arreglo

Dar un número entero para el elemento 3 del arreglo

Dar un número entero para el elemento 4 del arreglo

Los valores dados son:

-1282766448 32759 -1282766832 32759 -1282766976

Andorra04:Documents fp15alu39\$ █

Programa 3.c

```
1 #include <stdio.h>
2
3 int main ()
4
5 {
6     char *ap, c = 'a'; // Se declara el apuntador ap de tipo alfanumérico
7     ap = &c; //Se le asigna al apuntador la dirección de memoria de la variable c
8     printf("Carácter: %c\n",*ap); /* Se imprime el contenido de la variable a la
9     que apunta el apuntador ap */
10    printf("Código ASCII: %d\n",*ap); /*Se imprime el código ASCII del carácter
11    'a' */
12    printf("Dirección de memoria: %d\n",ap);/*Se imprime la dirección de
13    memoria que almacena el apuntador*/
14    return 0;
15 }
```

main.c: In function 'main':

main.c:12:33: warning: format '%d' expects argument of type 'int', but argument 2 has type 'char *' [-Wformat=]

```
12 | printf("Dirección de memoria: %d\n",ap);/*Se imprime la dirección de
```

	~^	~~
	int	char *
	%s	

Carácter: a

Código ASCII: 97

Dirección de memoria: 1963040799

...Program finished with exit code 0

Press ENTER to exit console.█



Programa 4.c

```

1  #include <stdio.h>
2  int main () {
3  int a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0};
4  int *apEnt;
5  apEnt = &a;
6  printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n"); printf("apEnt = &a\n");
7  /*A la variable b se le asigna el contenido de la variable a la que
8  apunta apEnt*/
9  b = *apEnt;
10 printf("b = *apEnt \t-> b = %i\n", b);
11 /*A la variable b se le asigna el contenido de la variable a la que
12 apunta apEnt y se le suma uno*/
13 b = *apEnt + 1;
14 printf("b = *apEnt + 1 \t-> b = %i\n", b);
15 //La variable a la que apunta apEnt se le asigna el valor cero
16 *apEnt = 0;
17 printf("*apEnt = 0 \t-> a = %i\n", a);
18 /*A apEnt se le asigna la dirección de memoria que tiene el elemento 0
19 del arreglo c*/
20 apEnt = &c[0];
21 printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt); return 0;
22 }

```

```

/tmp/FhbodfX1tn.o
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

```

=== Code Execution Successful ===



Programa 5.c

```

1  #include <stdio.h>
2  int main () {
3  int arr[] = {5, 4, 3, 2, 1};
4  int *apArr; //Se declara el apuntador apArr
5  int x;
6  apArr = arr;
7  printf("int arr[] = {5, 4, 3, 2, 1};\n");
8  printf("apArr = &arr[0]\n");
9  x = *apArr; /*A la variable x se le asigna el contenido del arreglo arr en
10 su elemento 0*/
11 printf("x = *apArr \t -> x = %d\n", x);
12 x = *(apArr+1); /*A la variable x se le asigna el contenido del arreglo arr
13 en su elemento 1*/
14 printf("x = *(apArr+1) \t -> x = %d\n", x);
15 x = *(apArr+2); /*A la variable x se le asigna el contenido del arreglo arr
16 en su elemento 2*/
17 printf("x = *(apArr+2) \t -> x = %d\n", x);
18 x = *(apArr+3); /*A la variable x se le asigna el contenido del arreglo arr
19 en su elemento 3*/
20 printf("x = *(apArr+3) \t -> x = %d\n", x);
21 x = *(apArr+4); /*A la variable x se le asigna el contenido del arreglo arr
22 en su elemento 4*/
23 printf("x = *(apArr+2) \t -> x = %d\n", x); return 0;
24 }

```

```

/tmp/yms5o1f88o.o
int arr[] = {5, 4, 3, 2, 1};
apArr = &arr[0]
x = *apArr -> x = 5
x = *(apArr+1) -> x = 4
x = *(apArr+2) -> x = 3
x = *(apArr+3) -> x = 2
x = *(apArr+2) -> x = 1

```

=== Code Execution Successful ===



Programa 6a.c

```

C: programad.c
1  //Programa 6//
2
3  #include <stdio.h>
4
5  int main ()
6  {
7
8  printf("\nEste programa permite declarar un apuntador ap con 5 números enteros.\n");
9
10 int lista[5] = {10, 8, 5, 8, 7};
11 int *ap = lista; //Se declara el apuntador ap
12 int indice;
13
14 printf("\nLista:\n");
15
16 //Se accede a cada elemento del arreglo haciendo uso del ciclo for
17 for (indice = 0; indice < 5; indice++)
18 {
19
20 printf("\nCalificación del alumno: %d es %d", indice+1, *(ap+indice));
21 }
22
23 printf("\n");
24
25 return 0;
26
27 }
28
29

```

Last login: Fri Apr 12 16:05:38 on ttys000

The default interactive shell is now zsh.

To update your account to use zsh, please run `chsh -s /bin/zsh`.

For more details, please visit <https://support.apple.com/kb/HT208050>.

Andorra04:~ fp15alu39\$ cd Documents/

Andorra04:Documents fp15alu39\$ ls

```
programa1c      programa2      programa2.c     programa6
programa1c.c    programa2.    programa2a.c    programa6.c
```

Andorra04:Documents fp15alu39\$ gcc programa6.c -o programa6

Andorra04:Documents fp15alu39\$./programa6

Este programa permite declarar un apuntador ap con 5 números enteros.

Lista:

Calificación del alumno: 1 es 10

Calificación del alumno: 2 es 8

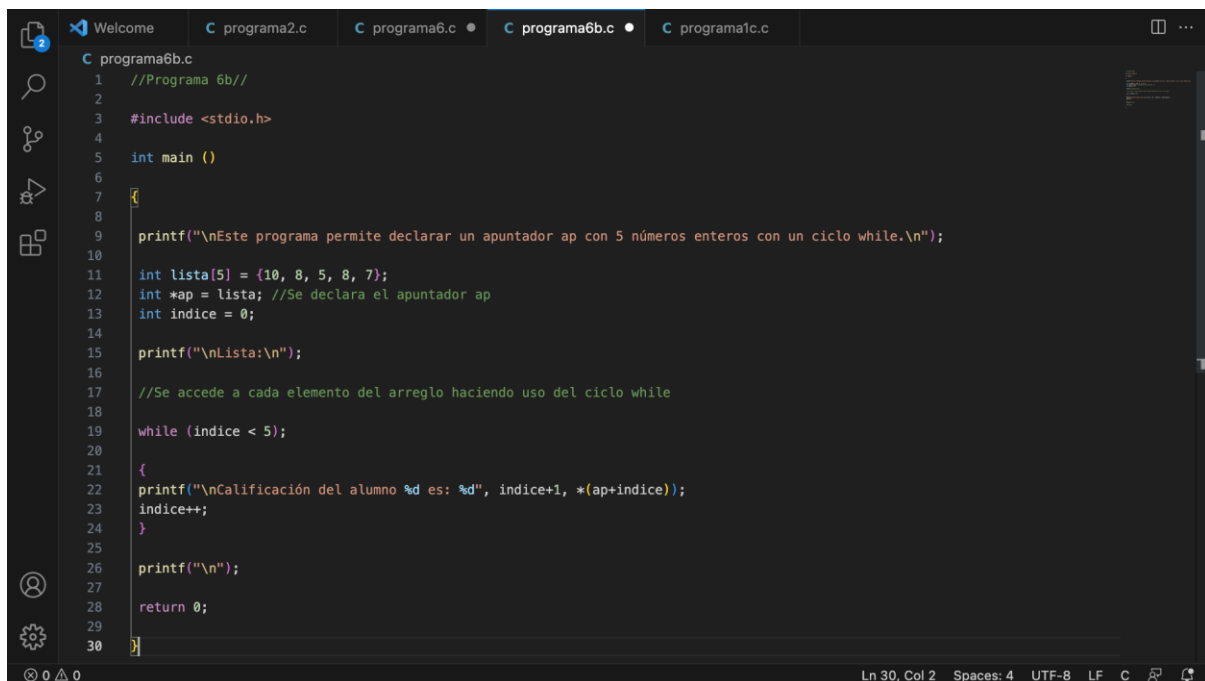
Calificación del alumno: 3 es 5

Calificación del alumno: 4 es 8

Calificación del alumno: 5 es 7

Andorra04:Documents fp15alu39\$

Programa 6b.c



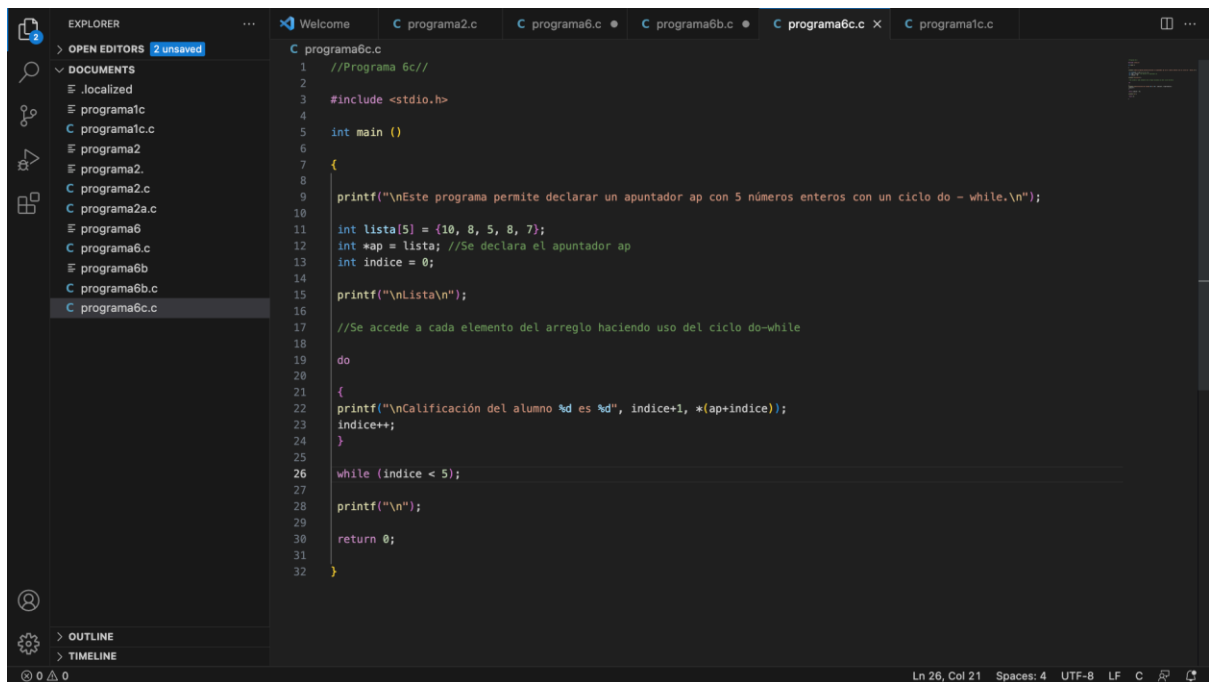
```
1 //Programa 6b//
2
3 #include <stdio.h>
4
5 int main ()
6 {
7
8
9     printf("\nEste programa permite declarar un apuntador ap con 5 números enteros con un ciclo while.\n");
10
11     int lista[5] = {10, 8, 5, 8, 7};
12     int *ap = lista; //Se declara el apuntador ap
13     int indice = 0;
14
15     printf("\nLista:\n");
16
17     //Se accede a cada elemento del arreglo haciendo uso del ciclo while
18
19     while (indice < 5);
20
21     {
22         printf("\nCalificación del alumno %d es: %d", indice+1, *(ap+indice));
23         indice++;
24     }
25
26     printf("\n");
27
28     return 0;
29
30 }
```

Console

Este programa permite declarar un apuntador ap con 5 números enteros con un ciclo while.

Lista:

Programa 6c.c



```
1 //Programa 6c//
2
3 #include <stdio.h>
4
5 int main ()
6 {
7
8     printf("\nEste programa permite declarar un apuntador ap con 5 números enteros con un ciclo do - while.\n");
9
10    int lista[5] = {10, 8, 5, 8, 7};
11    int *ap = lista; //Se declara el apuntador ap
12    int indice = 0;
13
14    printf("\nLista\n");
15
16    //Se accede a cada elemento del arreglo haciendo uso del ciclo do-while
17    do
18    {
19        printf("\nCalificación del alumno %d es %d", indice+1, *(ap+indice));
20        indice++;
21    }
22    while (indice < 5);
23
24    printf("\n");
25
26    return 0;
27 }
```

Last login: Fri Apr 12 16:27:44 on ttys000

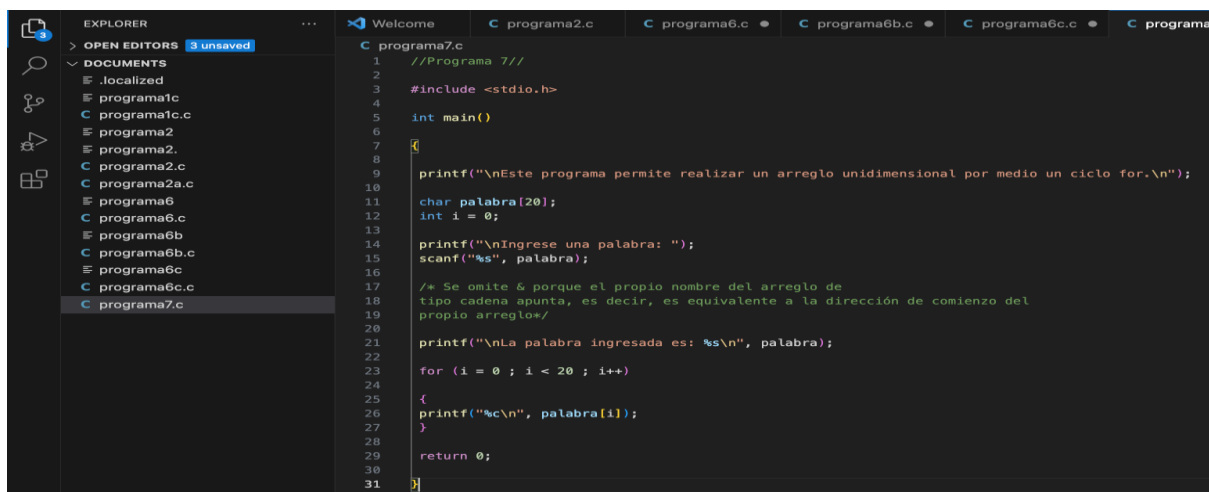
```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Andorra04:~ fp15alu39$ cd Documents/
Andorra04:Documents fp15alu39$ ls
programa1c      programa2.      programa6      programa6b.c
programa1c.c    programa2.c    programa6.c    programa6c.c
programa2       programa2a.c   programa6b
Andorra04:Documents fp15alu39$ gcc programa6c.c -o programa6c
Andorra04:Documents fp15alu39$ ./programa6c
```

Este programa permite declarar un apuntador ap con 5 números enteros con un ciclo do - while.

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
Andorra04:Documents fp15alu39$
```

Programa7.c



```
1 //Programa 7//
2
3 #include <stdio.h>
4
5 int main()
6 {
7
8     printf("\nEste programa permite realizar un arreglo unidimensional por medio un ciclo for.\n");
9
10    char palabra[20];
11    int i = 0;
12
13    printf("\nIngresa una palabra: ");
14    scanf("%s", palabra);
15
16    /* Se omite & porque el propio nombre del arreglo de
17    tipo cadena apunta, es decir, es equivalente a la dirección de comienzo del
18    propio arreglo*/
19    printf("\nLa palabra ingresada es: %s\n", palabra);
20
21    for (i = 0 ; i < 20 ; i++)
22    {
23        printf("%c\n", palabra[i]);
24    }
25
26    return 0;
27 }
```

```

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Andorra04:~ fp15alu39$ cd Documents/
Andorra04:Documents fp15alu39$ ls
programa1c  programa2.  programa6  programa6b.c  programa7.c
programa1c.c  programa2.c  programa6.c  programa6c  programa6c
programa2  programa2a.c  programa6b  programa6c.c
Andorra04:Documents fp15alu39$ gcc programa7.c -o programa7
Andorra04:Documents fp15alu39$ ./programa7

```

Este programa permite realizar un arreglo unidimensional por medio un ciclo for.

Ingrese una palabra: guapito

La palabra ingresada es: guapito

g
u
a
p
i
t
o

?
1
Z
?

?
R
?

Andorra04:Documents fp15alu39\$ █

Problema Adicional 1:

```

main.c
1- /*
2-
3- Programa Adicional 1 de La Práctica 09:
4-
5- Desarrolla el siguiente programa en C. Define un arreglo tamaño 8,
6- llena el arreglo (con scanf) iterativamente (usando ciclo for, while o do-while)
7- con valores flotantes. Suma e imprime los índices por pares, es decir,
8- el índice 0 y 1, el 2 y 3, ..., 6 y 7.
9-
10- Ejemplo de arreglo: 2.3, 1.1, 3.1, 0.1, 0.1, 0.3, 7.0, 2.6
11- Salida esperada: 3.4 3.2 0.4 9.6
12-
13- */
14-
15- #include <stdio.h>
16-
17- float arregloflotantes[8];
18- int i;
19-
20- int main(int argc, char * argv[])
21- {
22-     //Llenado del arreglo de tamaño 8.
23-     for(i = 0; i <= 7; i++)
24-     {
25-         printf("\nIngresa un flotante para el índice %d: ", i);
26-         scanf("%f", &arregloflotantes[i]);
27-     }
28-
29-     //Prueba de impresión del arreglo
30-     for(i = 1; i <= 7; i+=2)
31-     {
32-         printf("%.2f,", arregloflotantes[i] + arregloflotantes[i-1]);
33-     }
34-
35-     return 0;
36- }

```

```

Run Debug Stop Share Save
input
Ingresa un flotante para el índice 0: 2.3
Ingresa un flotante para el índice 1: 1.1
Ingresa un flotante para el índice 2: 3.1
Ingresa un flotante para el índice 3: 0.1
Ingresa un flotante para el índice 4: 0.1
Ingresa un flotante para el índice 5: 0.3
Ingresa un flotante para el índice 6: 7.0
Ingresa un flotante para el índice 7: 2.6
3.40,3.20,0.40,9.60,
...Program finished with exit code 0
Press ENTER to exit console.


```


Problema Adicional 2:


```
1 /*
2 Programa Adicional 2 de la Práctica 09:
3
4 Desarrollo de un programa en C, donde crees un arreglo tamaño 5.
5 llenando el arreglo iterativamente usando un ciclo for,
6 asimismo el arreglo está llenado con caracteres.
7 Con otro ciclo for, imprimimos sus valores ASCII como enteros.
8
9
10 Ejemplo de arreglo: 'a', 'b', 'c', 'd', 'e'
11 Salida esperada: 97 98 99 100 101
12 */
13
14 #include <stdio.h>
15
16 char arr [5];
17 char * ap;
18 int pos;
19
20 int main ()
21 {
22     printf("Este programa permite realizar un arreglo de tamaño 5 de caracteres con el fin de encontrar su equivalencia en Código ASCII.\n");
23     //ap = &arr[0]
24     ap = arr;
25     //llenando el arreglo de caracteres
26     for (pos = 0; pos < 5; pos++)
27     {
28         printf("Ingrese un carácter para arr [%d]: ", pos);
29         scanf("%c", ap + pos);
30         getchar();
31     }
32     //Impresión para comprobar si el arreglo está lleno
33     for (pos = 0; pos < 5; pos++)
34     {
35         printf("%d", *(ap + pos));
36     }
37     return 0;
38 }
```


```
1 /*
2 Programa Adicional 2 de la Práctica 09:
3
4 Desarrollo de un programa en C, donde crees un arreglo tamaño 5.
5
6 Input
7 Este programa permite realizar un arreglo de tamaño 5 de caracteres con el fin de encontrar su equivalencia en Código ASCII.
8 Ingrese un carácter para arr [0]: a
9 Ingrese un carácter para arr [1]: b
10 Ingrese un carácter para arr [2]: c
11 Ingrese un carácter para arr [3]: d
12 Ingrese un carácter para arr [4]: e
13 979899100101
14 ...Program finished with exit code 0
15 Press ENTER to exit console.
```


4) Conclusiones (Individuales):

 **García Sánchez Alejandro:** Esta práctica me ayudó a entender no sólo la definición acerca de un apuntador y un arreglo, sino también a comprender en qué consisten y en qué determinado momento los podemos emplear, ya sea, uno en un programa o ambos. Mientras que el arreglo es un conjunto de índices o espacios que almacenan N elementos, dependiendo el tamaño definido y el apuntador es una dirección de memoria, es decir, podemos determinar en qué espacio de la memoria puede ser guardado determinado dato.

 **López Castro Anastacia:** En esta práctica, pudimos visualizar las diferencias entre usar un ciclo a otro, cabe mencionar que estas mismas

ya se habían visto casi desde el comienzo del curso, a lo cual no hubo problemas con eso. Sin embargo, casi al final de la práctica viene la parte de lo que son los apuntadores y como trabajarlos, de igual manera, la definición que se da no es tan fácil de asimilar

 **Ramírez Rivas Gael:** En esta práctica se realizaron programas en lenguaje C orientados a probar y usar arreglos, los cuales sirven para organizar o agrupar datos según sea requerido. Además, se hizo énfasis en los apuntadores, que son variables que almacenan la dirección de memoria de otra variable, los cuales son útiles en el desarrollo de ciertos programas según el problema que se busque resolver. Los ejercicios de la práctica se realizaron y ejecutaron sin problemas permitiéndonos adquirir conocimientos y cumplir los objetivos planteados previamente.


 **Ruíz Hernández Rubén Antonio:** Se cumplieron al pie de la letra los objetivos de la práctica, ya que aprendí la diferencia entre un arreglo y apuntador, ya que el arreglo son estructuras de datos almacenadas en una sola variable, de igual manera como introducción aprendí a manejar arreglos unidimensionales los cuales son tipos de datos estructurados que están formados por una colección finita y ordenada de datos del mismo tipo, por otro lado, vimos un poco de enfoque respecto al concepto de apuntadores donde las variables contienen direcciones de memoria como sus valores, del mismo modo, realice el programa 5.c en lenguaje C, donde se aprecia un arreglo unidimensional de 5 elementos, asimismo accede a cada uno de los elementos del arreglo haciendo uso de un apuntador, para ello se utilizando un ciclo for.

5) Retroalimentación (Equipo):

Un arreglo es una forma concreta de organizar o agrupar elementos de datos de un mismo tipo, permitiendo almacenarlos en ubicaciones de memoria contiguas, facilitando su manejo. Un apuntador es una variable que almacena la dirección de memoria de otra variable, siendo útiles en la gestión de memoria. El manejo y uso de estas características programables es útil en la creación de programas usados en la resolución de problemas, así como

también en la optimización de estos. El conocimiento y correcto manejo de los arreglos y apuntadores nos permite dar mejores respuestas a problemas, siendo estos una herramienta útil y con aplicaciones notables a lo largo de nuestras carreras permitiéndonos lograr un mejor desempeño.

6) Fuentes en APA:

 Laboratorio Salas A y B. (s.f.). *Manual de Prácticas de la Asignatura Fundamentos de Programación (Guía práctica de estudio 09: Arreglos Unidimensionales, pág. 137 - 150)*. Recuperado el 12 de Abril del 2024, de Laboratorio de Computación Salas A y B: <http://lcp02.fi-b.unam.mx/>

 RuyAntonio. (s.f.). *GitHub - RuyAntonio/practica9_fdp: Práctica 09: Arreglos Unidimensionales*, de GitHub. Recuperado el 12 de Abril del 2024, de GitHub: https://github.com/RuyAntonio/practica09_fdp.git