



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* Ing. Karina García Morales

*Asignatura:* Fundamentos de la Programación

*Grupo:* 20

*No. de práctica(s):* 09

*Integrante(s):* Martínez Ordoñez Diego Tonatíuh

*No. de lista o brigada:* 30

*Semestre:* 2023-1

*Fecha de entrega:* 03 / diciembre/ 2022

*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.

Cada elemento de este arreglo se le va asociar con una posición particular, el cual necesitara indicaciones para poder acceder a un elemento en específico, lo cual se podrá llevar a cabo a través de los índices.

Los arreglos pueden ser unidimensionales o multidimensionales, la dimensión de cada arreglo va depender del número de índices que va requerir para acceder a un elemento del arreglo.

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 es un tipo de datos estructurado que está formado de una colección finita y ordenada de datos del mismo tipo.

Es la estructura natural para modelar listas de elementos iguales, Un arreglo unidimensional de n elementos se almacena en la memoria de la siguiente manera:



La sintaxis para definir un arreglo unidimensional en lenguaje C es la siguiente:  
`tipoDeDato nombre[tamaño]`

## - Códigos en clase:

1. 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.

```
1 //
2 // main.c
3 // arreglo2
4 //
5 // Created by Martinez Ordonez Diego Tonatiuh on 11/23/22.
6 // Copyright © 2022 Martinez Ordonez Diego Tonatiuh. All rights reserved.
7 //
8 #include <stdio.h> //libreria
9 int indice; //declaro variable
10 int main ()
11 {
12     int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional int
13     indice = 0; //***** VALOR INICIAL
14     printf("\tlista\n");
15     while (indice < 7) // Acceso a cada elemento del arreglo unidimensional usando while
16     //*****CONDICIÓN
17     {
18         printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]); indice += 1; //
19         // Sentencia análoga a indice = indice + 1; //*****INCREMENTO
20     }
21     printf("\n");
22     return 0;
23 }
```

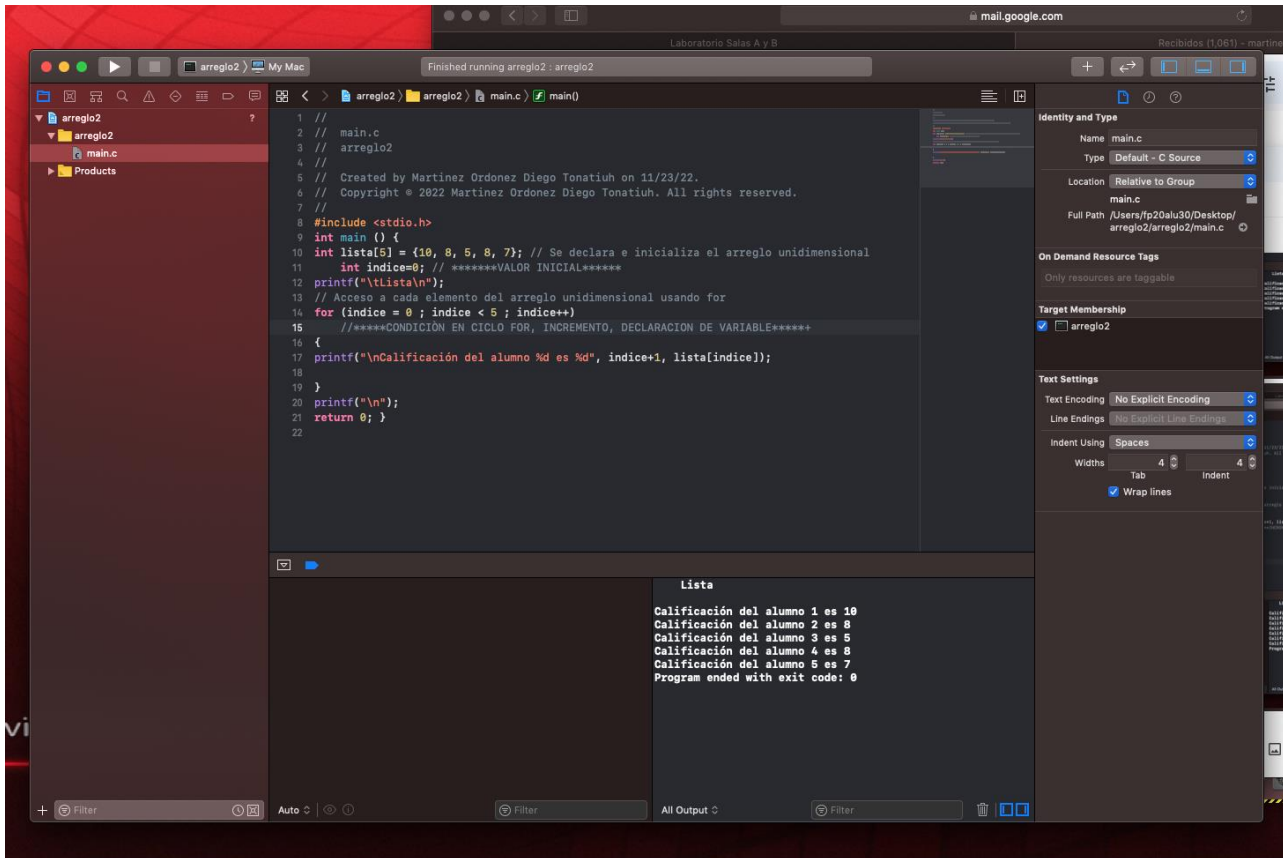
Calificación del alumno 3 es 5  
Calificación del alumno 4 es 8  
Calificación del alumno 5 es 7  
Calificación del alumno 6 es 32766  
Calificación del alumno 7 es -1489385479  
Program ended with exit code: 0

2. 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.

```
1 //
2 // main.c
3 // arreglo2
4 //
5 // Created by Martinez Ordonez Diego Tonatiuh on 11/23/22.
6 // Copyright © 2022 Martinez Ordonez Diego Tonatiuh. All rights reserved.
7 //
8 #include <stdio.h>
9 int main () {
10     int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional int
11     int indice = 0; //*****VALOR INICIAL*****
12     printf("\tlista\n");
13     do // *****CONDICIÓN DO-WHILE*****
14     {
15         printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
16         indice += 1; // ***INCREMENTO*****
17     }
18     while (indice < 5 ); printf("\n");
19     return 0;
20 }
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  
Program ended with exit code: 0

3. 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 índice desde 0 hasta 4 haciendo uso de un ciclo for.



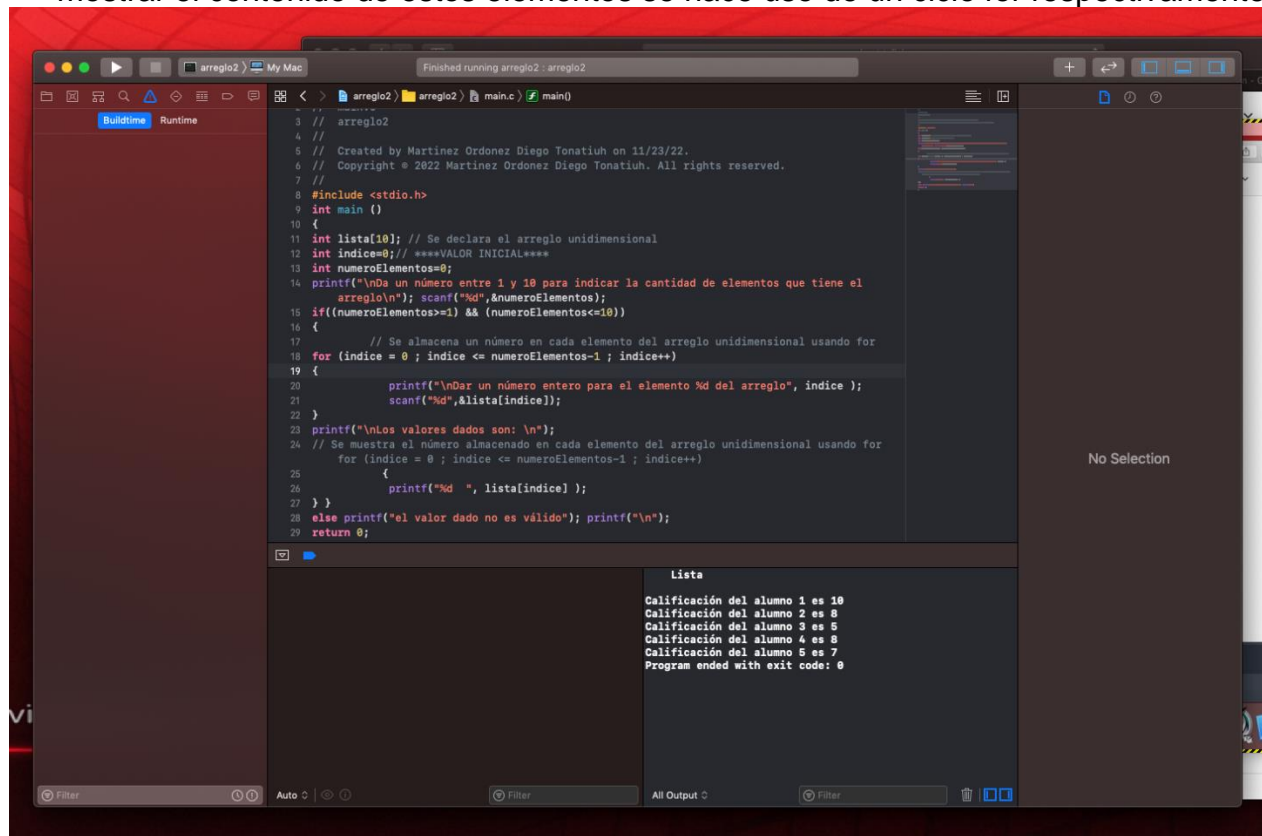
The screenshot shows a code editor with a C program. The code declares an array of 5 integers, initializes it with values 10, 8, 5, 8, and 7, and then uses a for loop to print each element. The output window shows the results of the program execution.

```
1 //  
2 // main.c  
3 // arreglo2  
4 //  
5 // Created by Martinez Ordóñez Diego Tonatiuh on 11/23/22.  
6 // Copyright © 2022 Martinez Ordóñez Diego Tonatiuh. All rights reserved.  
7 //  
8 #include <stdio.h>  
9 int main () {  
10     int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional  
11     int indice=0; // *****VALOR INICIAL*****  
12     printf("\nLista\n");  
13     // Acceso a cada elemento del arreglo unidimensional usando for  
14     for (indice = 0 ; indice < 5 ; indice++)  
15     {  
16         //*****CONDICIÓN EN CICLO FOR, INCREMENTO, DECLARACIÓN DE VARIABLE*****  
17         printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);  
18     }  
19     printf("\n");  
20     return 0; }  
21  
22
```

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  
Program ended with exit code: 0
```

4. 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.



The screenshot shows a code editor with a C program. The code declares an array of 10 integers, reads the number of elements to be stored from the user, and then uses a for loop to read and print each element. The output window shows the results of the program execution.

```
3 // arreglo2  
4 //  
5 // Created by Martinez Ordóñez Diego Tonatiuh on 11/23/22.  
6 // Copyright © 2022 Martinez Ordóñez Diego Tonatiuh. All rights reserved.  
7 //  
8 #include <stdio.h>  
9 int main ()  
10 {  
11     int lista[10]; // Se declara el arreglo unidimensional  
12     int indice=0; // *****VALOR INICIAL*****  
13     int numeroElementos=0;  
14     printf("\nDa un número entre 1 y 10 para indicar la cantidad de elementos que tiene el  
15     arreglo\n"); scanf("%d",&numeroElementos);  
16     if((numeroElementos>1) && (numeroElementos<=10))  
17     {  
18         // Se almacena un número en cada elemento del arreglo unidimensional usando for  
19         for (indice = 0 ; indice <= numeroElementos-1 ; indice++)  
20         {  
21             printf("\nDa un número entero para el elemento %d del arreglo", indice );  
22             scanf("%d",&lista[indice]);  
23         }  
24         printf("\nLos valores dados son: \n");  
25         // Se muestra el número almacenado en cada elemento del arreglo unidimensional usando for  
26         for (indice = 0 ; indice <= numeroElementos-1 ; indice++)  
27         {  
28             printf("%d ", lista[indice]);  
29         }  
30         printf("\n");  
31         return 0;  
32     }  
33 }
```

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  
Program ended with exit code: 0
```

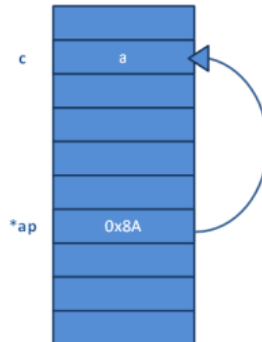
## Apuntadores.

Un apuntador es una variable que contiene la dirección de una variable, es decir, hace referencia a la localidad de memoria de otra variable, estos trabajan directamente con la memoria, obteniendo rápidamente un dato solicitado.

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

```
TipoDeDato *apuntador, variable;  
apuntador = &variable;
```

La declaración de una variable apuntador inicia con el carácter \*. 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 \*.



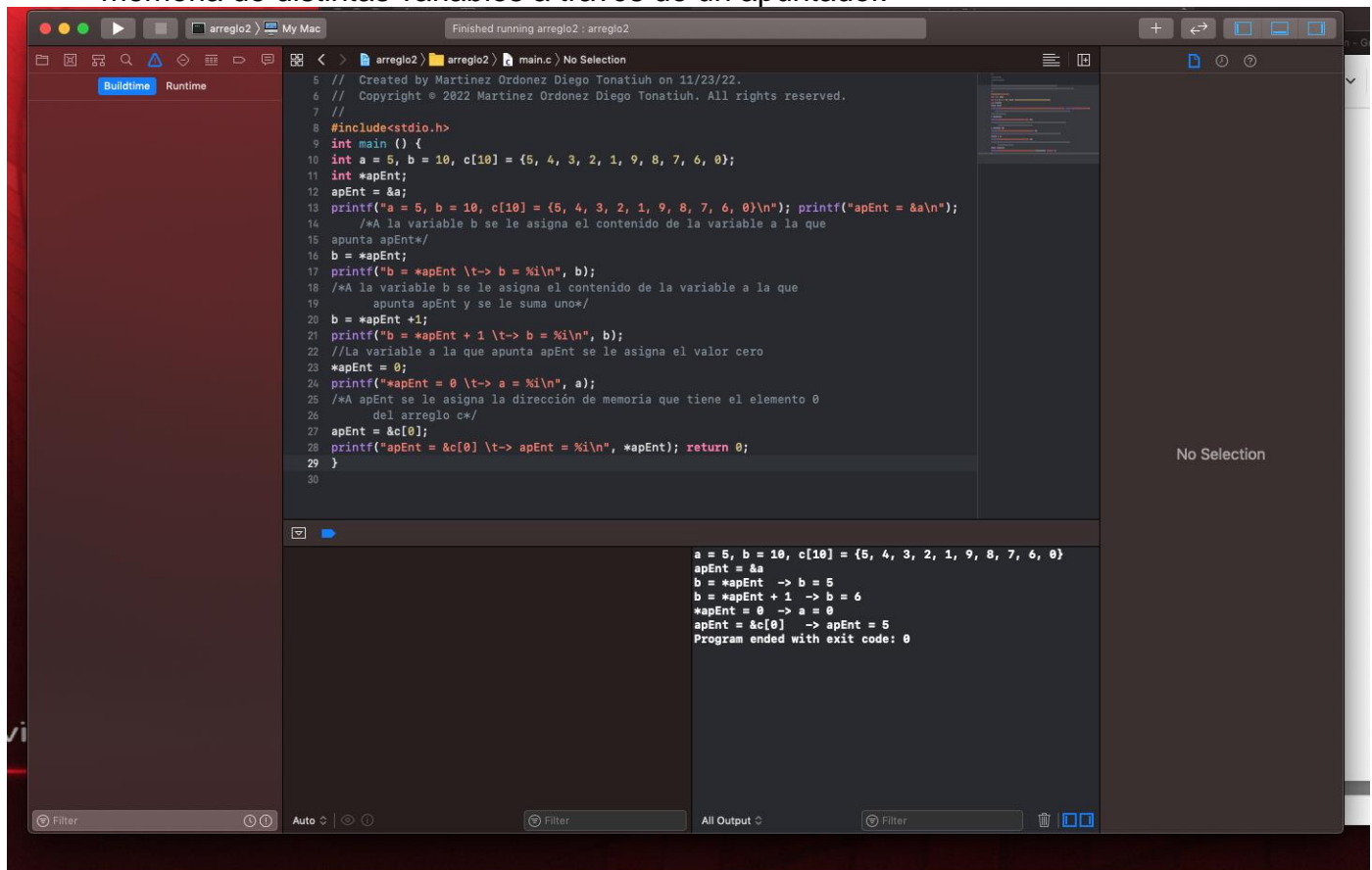
## - Códigos en clase.

5. 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.

```
1 //  
2 // main.c  
3 // arreglo2  
4 //  
5 // Created by Martinez Ordonez Diego Tonatiuh on 11/23/22.  
6 // Copyright © 2022 Martinez Ordonez Diego Tonatiuh. All rights reserved.  
7 //  
8 #include <stdio.h>  
9 int main ()  
10 {  
11     char *ap, c = 'a'; // Se declara el apuntador ap de tipo alfanumérico  
12     ap = &c; //Se le asigna el apuntador la dirección de memoria de la variable  
13     printf("Carácter: %c\n", *ap); /* Se imprime el contenido de la variable a laque apunta  
14     el apuntador ap */  
15     printf("Código ASCII: %d\n", *ap); /*Se imprime el código ASCII del carácter'a' */  
16     printf("Dirección de memoria: %d\n", ap); /*Si se agrega el * solo damos el contenido  
17     de la variable que asignamos  
18     return 0;  
19 }  
20 //memoria que almacena el apuntador*/
```

Carácter: a  
Código ASCII: 97  
Dirección de memoria: -272632577  
Program ended with exit code: 0

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



```
5 // Created by Martinez Ordonez Diego Tonatiuh on 11/23/22.
6 // Copyright © 2022 Martinez Ordonez Diego Tonatiuh. All rights reserved.
7 //
8 #include<stdio.h>
9 int main () {
10 int a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0};
11 int *apEnt;
12 apEnt = &a;
13 printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n"); printf("apEnt = &a\n");
14 /*A la variable b se le asigna el contenido de la variable a la que
15 apunta apEnt*/
16 b = *apEnt;
17 printf("b = *apEnt \t-> b = %i\n", b);
18 /*A la variable b se le asigna el contenido de la variable a la que
19 apunta apEnt y se le suma uno*/
20 b = *apEnt + 1;
21 printf("b = *apEnt + 1 \t-> b = %i\n", b);
22 //La variable a la que apunta apEnt se le asigna el valor cero
23 *apEnt = 0;
24 printf("**apEnt = 0 \t-> a = %i\n", a);
25 /*A apEnt se le asigna la dirección de memoria que tiene el elemento 0
26 del arreglo c*/
27 apEnt = &c[0];
28 printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt); return 0;
29 }
30
```

Output:

```
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
Program ended with exit code: 0
```

### - 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 en clase:

7. 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.



```
7 //
8 #include <stdio.h>
9 int main () {
10 int arr[] = {5, 4, 3, 2, 1};
11 int *apArr; //Se declara el apuntador apArr
12 int x;
13 apArr = arr;
14 printf("int arr[] = {5, 4, 3, 2, 1};\n");
15 printf("apArr = &arr[0]\n");
16 x = *apArr; /*A la variable x se le asigna el contenido del arreglo arr en
17 su elemento 0*/
18 printf("x = *apArr \t -> x = %d\n", x);
19 x = *(apArr+1); /*A la variable x se le asigna el contenido del arreglo arr
20 en su elemento 1*/
21 printf("x = *(apArr+1) \t -> x = %d\n", x);
22 x = *(apArr+2); /*A la variable x se le asigna el contenido del arreglo arr
23 en su elemento 2*/
24 printf("x = *(apArr+2) \t -> x = %d\n", x);
25 x = *(apArr+3); /*A la variable x se le asigna el contenido del arreglo arr
26 en su elemento 3*/
27 printf("x = *(apArr+3) \t -> x = %d\n", x);
28 x = *(apArr+4); /*A la variable x se le asigna el contenido del arreglo arr
29 en su elemento 4*/
30 printf("x = *(apArr+4) \t -> x = %d\n", x); return 0;
31 }
32
```

```
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+4) -> x = 1
Program ended with exit code: 0
```

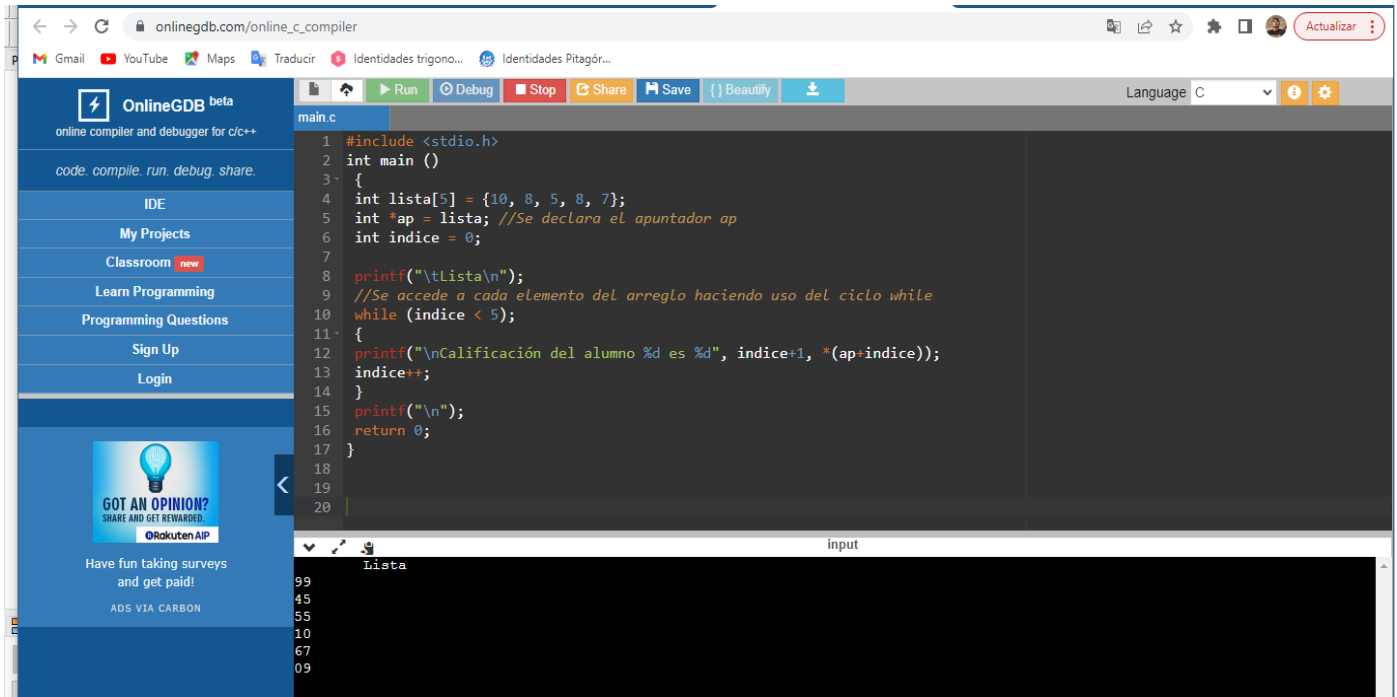
8. El siguiente programa genera un arreglo unidimensional de 5 elementos y accede a cada uno de los elementos del arreglo haciendo uso de un apuntador, para ello se utiliza un ciclo for.

```
1 //
2 // main.c
3 // arreglo2
4 //
5 // Created by Martinez Ordonez Diego Tonatiuh on 11/23/22.
6 // Copyright © 2022 Martinez Ordonez Diego Tonatiuh. All rights reserved.
7 //
8 #include <stdio.h>
9 int main () {
10 int lista[5] = {10, 8, 5, 8, 7};
11 int *ap = lista; //Se declara el apuntador ap
12 int indice;
13 printf("\tlista\n");
14 //Se accede a cada elemento del arreglo haciendo uso del ciclo for
15 for (indice = 0 ; indice < 5 ; indice++)
16 {
17 printf("\nCalificación del alumno %d es %d", indice+1, *(ap+indice)); }
18 printf("\n");
19 return 0; }
20
```

```
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
Program ended with exit code: 0
```

## Ejercicio de tarea:

9. 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.



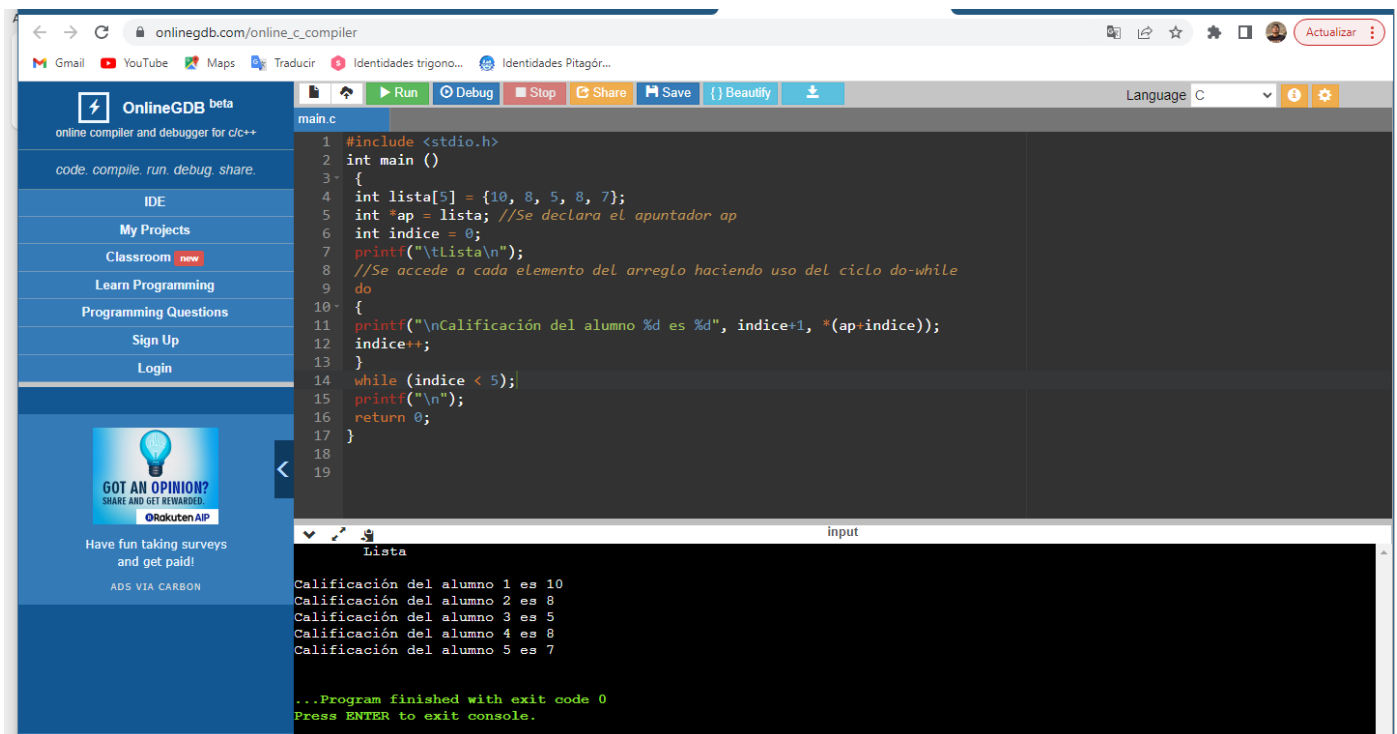
The screenshot shows the OnlineGDB web interface. The code in the editor is as follows:

```
1 #include <stdio.h>
2 int main ()
3 {
4     int lista[5] = {10, 8, 5, 8, 7};
5     int *ap = lista; //Se declara el apuntador ap
6     int indice = 0;
7
8     printf("\tlista\n");
9     //Se accede a cada elemento del arreglo haciendo uso del ciclo while
10    while (indice < 5);
11    {
12        printf("\nCalificación del alumno %d es %d", indice+1, *(ap+indice));
13        indice++;
14    }
15    printf("\n");
16    return 0;
17 }
```

The output window shows the following text:

```
Lista
99
45
55
10
67
09
```

10. 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.



The screenshot shows the OnlineGDB web interface. The code in the editor is as follows:

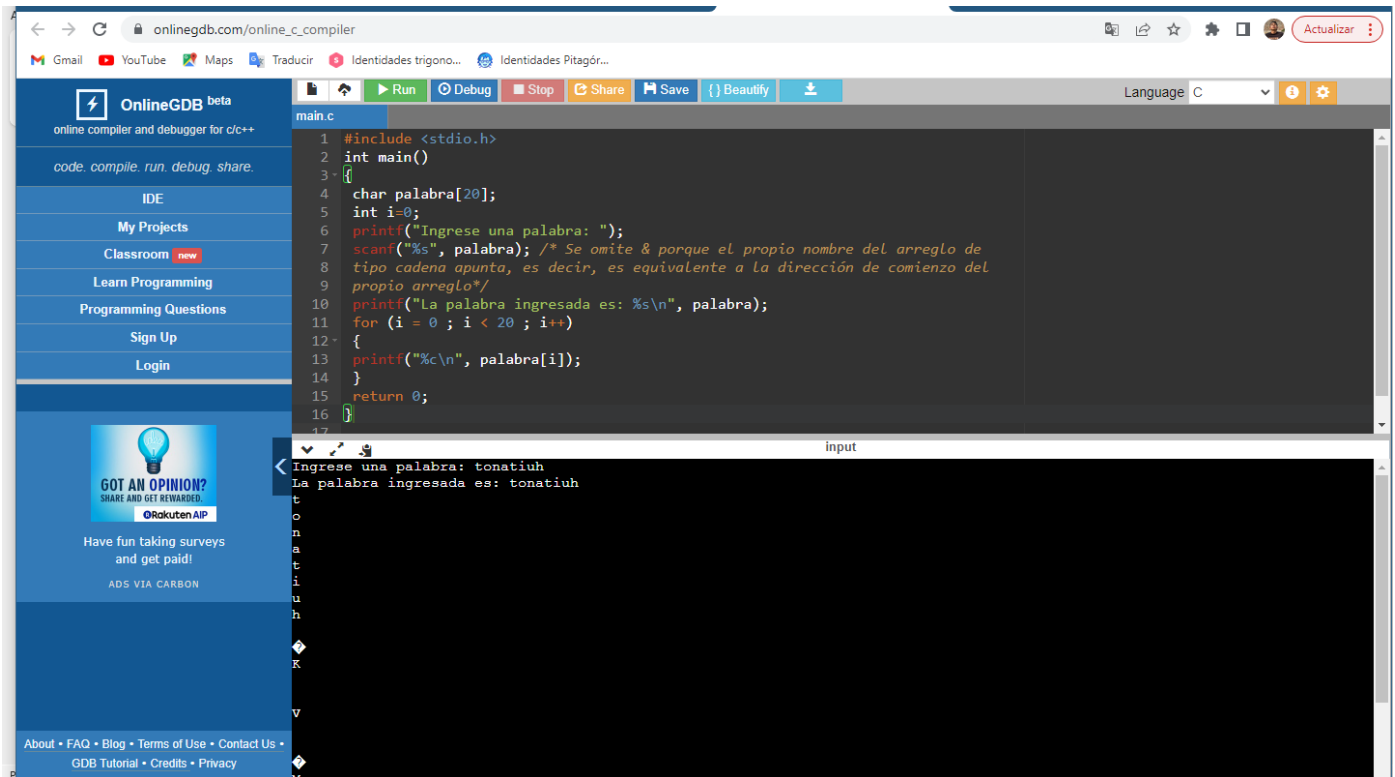
```
1 #include <stdio.h>
2 int main ()
3 {
4     int lista[5] = {10, 8, 5, 8, 7};
5     int *ap = lista; //Se declara el apuntador ap
6     int indice = 0;
7     printf("\tlista\n");
8     //Se accede a cada elemento del arreglo haciendo uso del ciclo do-while
9     do
10    {
11        printf("\nCalificación del alumno %d es %d", indice+1, *(ap+indice));
12        indice++;
13    }
14    while (indice < 5);
15    printf("\n");
16    return 0;
17 }
```

The output window shows the following text:

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



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



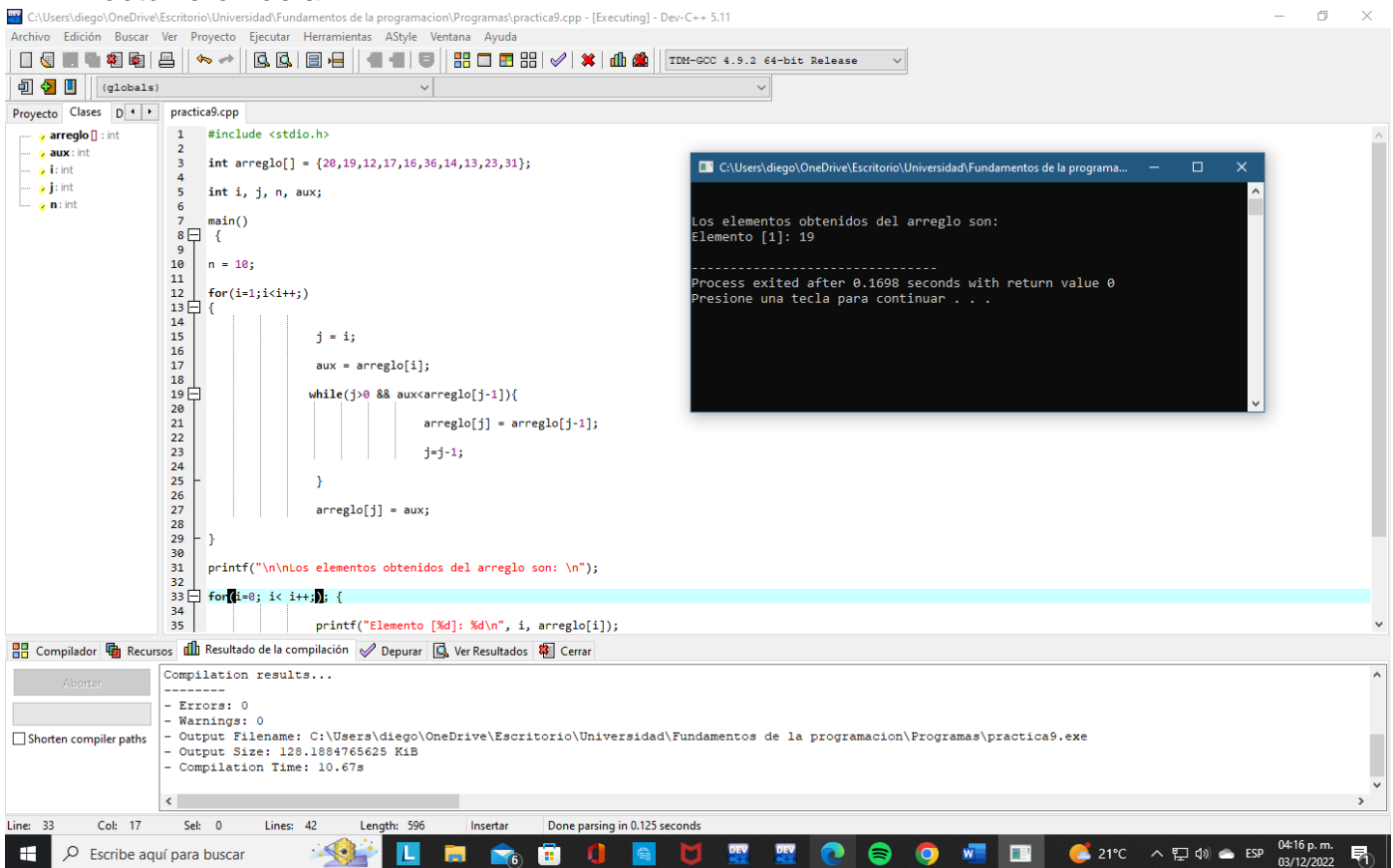
The screenshot shows the OnlineGDB website interface. On the left is a sidebar with navigation links like 'My Projects', 'Classroom', and 'Sign Up'. The main area displays a C program in a text editor. The program includes `<stdio.h>` and defines a `main` function. It declares a character array `palabra[20]` and a loop counter `i`. The program prompts the user to enter a word, reads it using `scanf`, and then prints the word using `printf` in a loop that iterates over each character. Below the code editor, there is an 'input' section showing the user's input 'tonatiuh' and the program's output 'La palabra ingresada es: tonatiuh'.

```
1 #include <stdio.h>
2 int main()
3 {
4     char palabra[20];
5     int i=0;
6     printf("Ingresa una palabra: ");
7     scanf("%s", palabra); /* Se omite & porque el propio nombre del arreglo de
8                             tipo cadena apunta, es decir, es equivalente a la dirección de comienzo del
9                             propio arreglo*/
10    printf("La palabra ingresada es: %s\n", palabra);
11    for (i = 0 ; i < 20 ; i++)
12    {
13        printf("%c\n", palabra[i]);
14    }
15    return 0;
16 }
```

input

Ingresa una palabra: tonatiuh  
La palabra ingresada es: tonatiuh

- **Ejercicio a ejecutar:** nos muestra los datos obtenidos de los arreglos, los valores que se obtuvieron de ahí.



The screenshot shows a C++ IDE with a project named 'practica9.cpp'. The code defines an array `arreglo` with 10 elements: {20, 19, 12, 17, 16, 36, 14, 13, 23, 31}. The `main` function sets `n = 10` and uses a `for` loop to iterate over the array. Inside the loop, it uses a `while` loop to print the elements in reverse order. The output window shows the result: 'Los elementos obtenidos del arreglo son: Elemento [1]: 19'. The compilation results at the bottom show 0 errors and 0 warnings.

```
1 #include <stdio.h>
2
3 int arreglo[] = {20,19,12,17,16,36,14,13,23,31};
4
5 int i, j, n, aux;
6
7 main()
8 {
9     n = 10;
10    for(i=1;i<i++;){
11        j = i;
12        aux = arreglo[i];
13        while(j>0 && aux<arreglo[j-1]){
14            arreglo[j] = arreglo[j-1];
15            j=j-1;
16        }
17        arreglo[j] = aux;
18    }
19    printf("\n\nLos elementos obtenidos del arreglo son: \n");
20    for(i=0; i< i++;){
21        printf("Elemento [%d]: %d\n", i, arreglo[i]);
22    }
```

Los elementos obtenidos del arreglo son:  
Elemento [1]: 19

Process exited after 0.1698 seconds with return value 0  
Presione una tecla para continuar . . .

Compilation results...  
- Errors: 0  
- Warnings: 0  
- Output Filename: C:\Users\diego\OneDrive\Escritorio\Universidad\Fundamentos de la programacion\Programas\practica9.exe  
- Output Size: 128.1884765625 KiB  
- Compilation Time: 10.67s

### - Conclusiones:

Pienso que esta práctica nos ayudo mucho en la cuestión de que nos brindo nuevas herramientas para poder trabajar de una manera más compleja, y de esta forma tener los conocimientos para poder facilitarnos o apoyarnos a cierto punto de poder realizar cualquier cosa que deseemos hacer o se nos pueda solicitar.

Los arreglos y apuntadores van de la mano, se complementan muy bien, lo que nos abre puertas para poder en un futuro trabajar de manera en que nos guardan datos y de igual forma más directa a tal punto de que es mucho más sencillo.

### - Bibliografía:

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