	Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia	

Laboratorios de computación salas A y B

Profesor: Karina García Morales

Asignatura: Fundamentos de programación

Grupo: 20

No. de práctica(s): 09

Integrante(s): Suzán Herrera Álvaro

No. de lista o brigada: 49

Semestre: 1

Fecha de entrega: 4 de diciembre 2022

Observaciones:

CALIFICACIÓN: _____

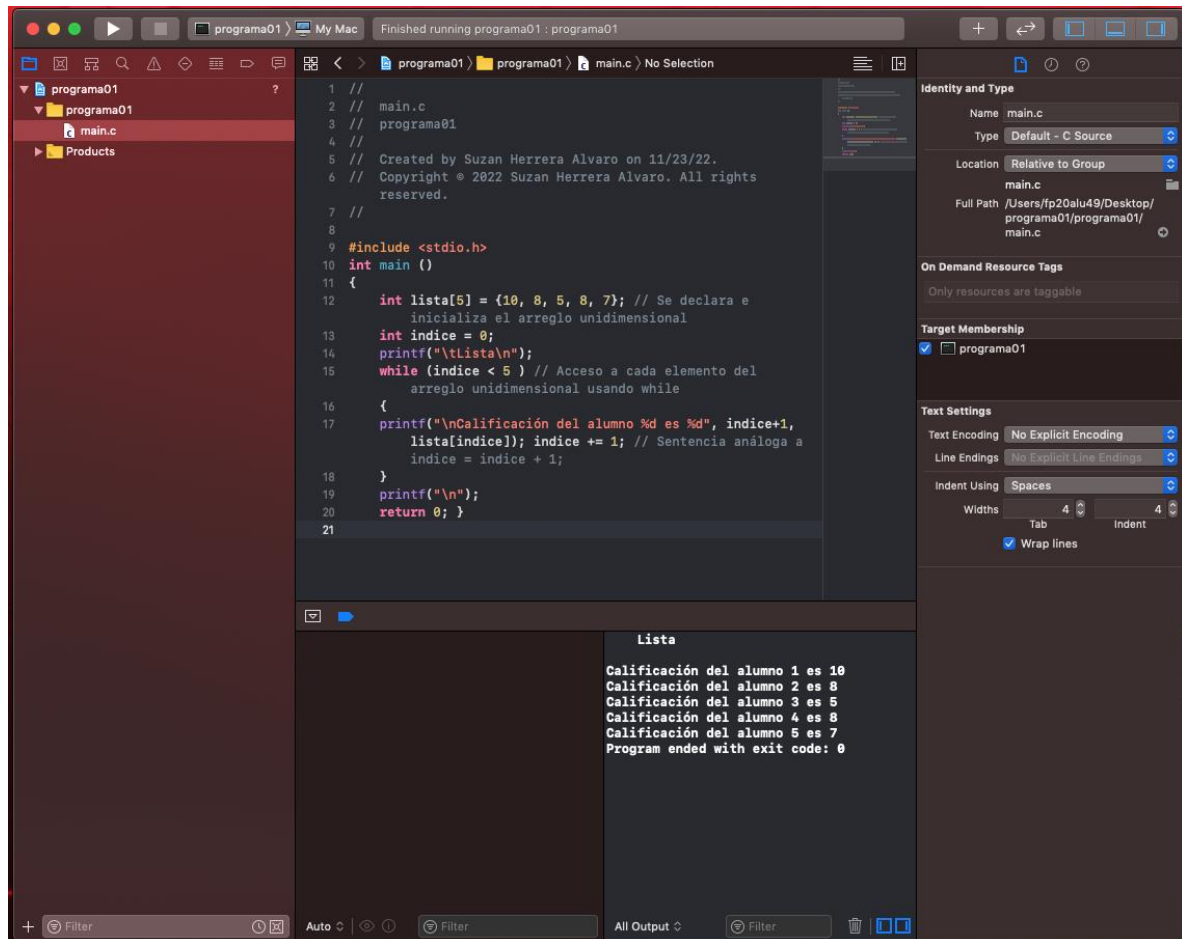
Objetivo:

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.

Conceptos:

Arreglo: Un arreglo es un conjunto de datos contiguos del mismo tipo con un tamaño fijo definido al momento de crearse.

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.



The screenshot shows a code editor with a C program. The program declares an array 'lista' of size 5, initializes it with values {10, 8, 5, 8, 7}, and uses a while loop to iterate through the array, printing each element and its index. The output window shows the results of the program execution.

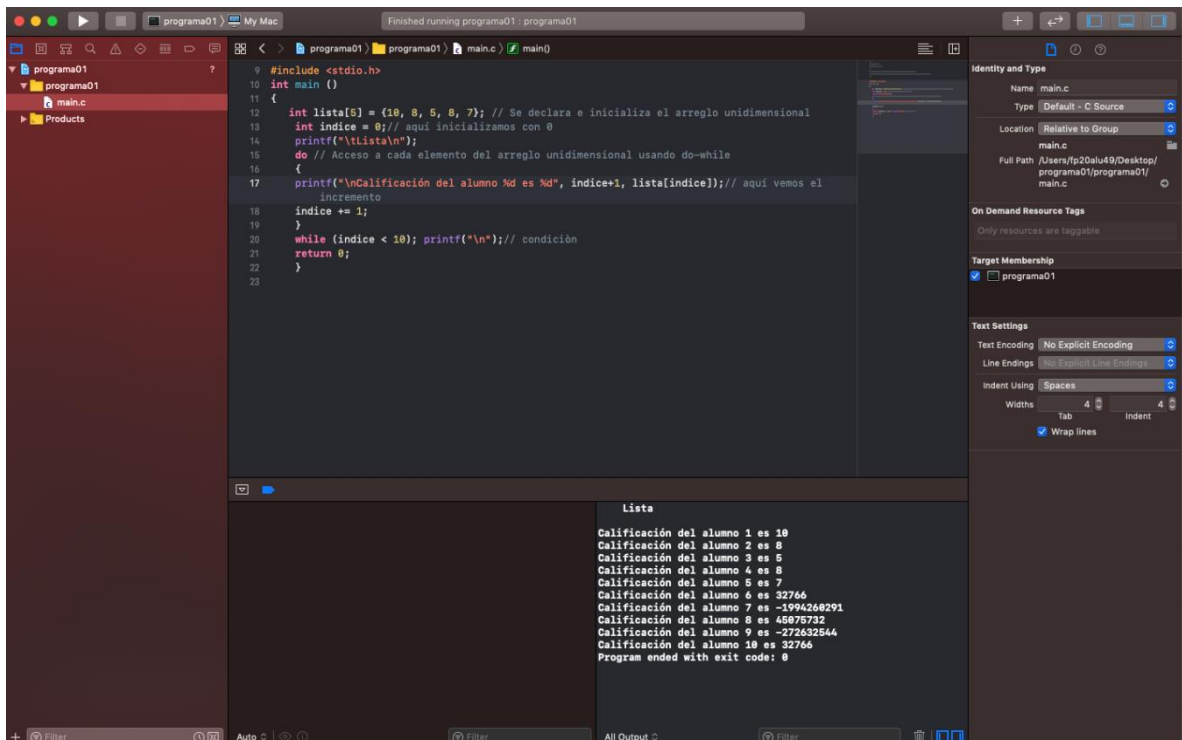
```
1 //
2 // main.c
3 // programa01
4 //
5 // Created by Suzan Herrera Alvaro on 11/23/22.
6 // Copyright © 2022 Suzan Herrera Alvaro. All rights reserved.
7 //
8
9 #include <stdio.h>
10 int main ()
11 {
12     int lista[5] = {10, 8, 5, 8, 7}; // Se declara e
        inicializa el arreglo unidimensional
13     int indice = 0;
14     printf("\tlista\n");
15     while (indice < 5 ) // Acceso a cada elemento del
        arreglo unidimensional usando while
16     {
17         printf("\nCalificación del alumno %d es %d", indice+1,
            lista[indice]); indice += 1; // Sentencia análoga a
            indice = indice + 1;
18     }
19     printf("\n");
20     return 0; }
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
```

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.



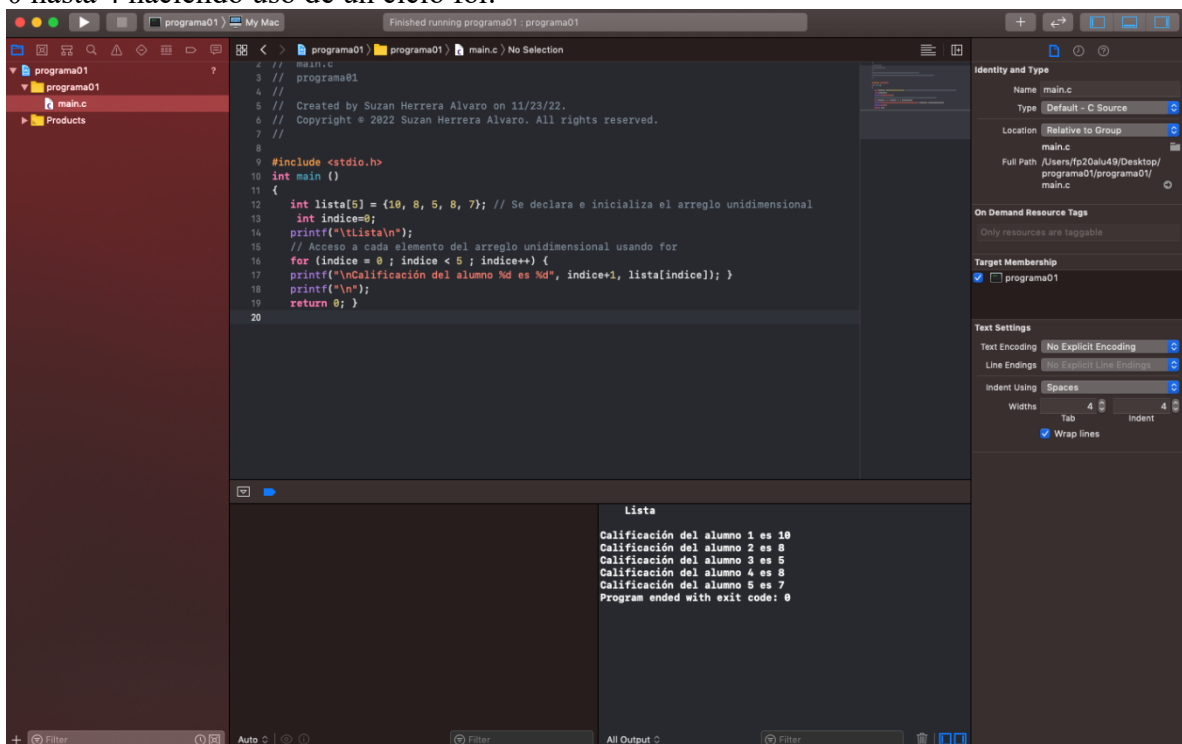
```
9 #include <stdio.h>
10 int main ()
11 {
12     int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional
13     int indice = 0; // aquí inicializamos con 0
14     printf("\nLista\n");
15     do // Acceso a cada elemento del arreglo unidimensional usando do-while
16     {
17         printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]); // aquí vemos el
            incremento
18         indice += 1;
19     }
20     while (indice < 10); printf("\n"); // condición
21     return 0;
22 }
23
```

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
Calificación del alumno 6 es 32766
Calificación del alumno 7 es -1096268291
Calificación del alumno 8 es 45878732
Calificación del alumno 9 es -272632544
Calificación del alumno 10 es 32766
Program ended with exit code: 0
```

Código (arreglo unidimensional empleando la estructura for)

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.



```
1 // main.c
2 // programa01
3 //
4 // Created by Suzan Herrera Alvaro on 11/23/22.
5 // Copyright © 2022 Suzan Herrera Alvaro. All rights reserved.
6 //
7 //
8 //
9 #include <stdio.h>
10 int main ()
11 {
12     int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional
13     int indice=0;
14     printf("\nLista\n");
15     // Acceso a cada elemento del arreglo unidimensional usando for
16     for (indice = 0 ; indice < 5 ; indice++) {
17         printf("\nCalificación del alumno %d es %d", indice+1, lista[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
```

Código (arreglo unidimensional empleando la estructura for)

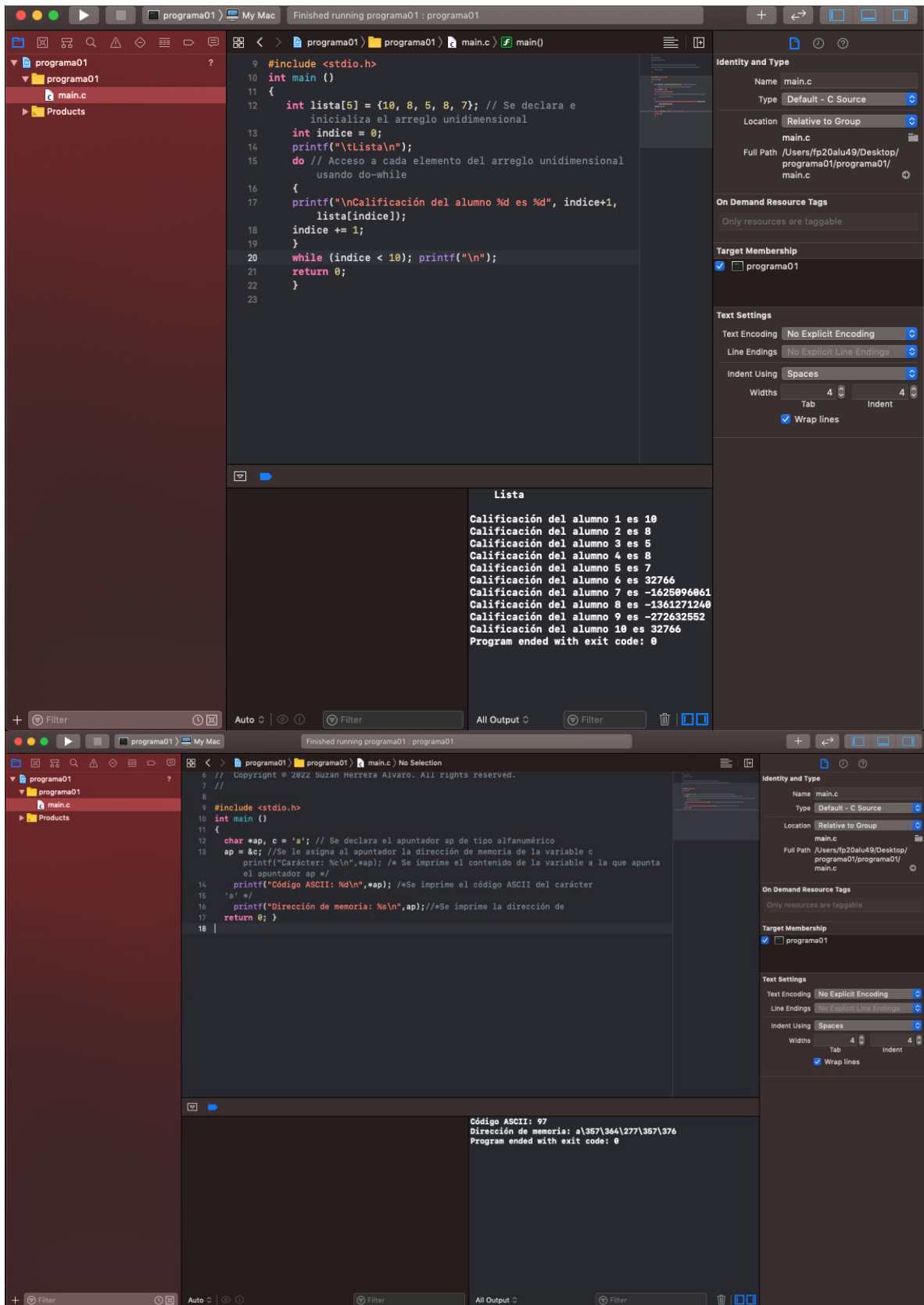
Se muestra un programa que genera un arreglo unidimensional de máximo 10 elementos.

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

```
10 int main ()
11 {
12     int lista[] = {10, 8, 5, 8, 7, 9}; // Se declara e
        inicializa el arreglo unidimensional
13     int indice = 0;
14     printf("\tlista\n");
15     while (indice < 7) // Acceso a cada elemento del
        arreglo unidimensional usando while
16     {
17         printf("\nCalificación del alumno %d es %d", indice+1,
            lista[indice]); indice += 1; // Sentencia análoga a
            indice = indice + 1;
18     }
19     printf("\n");
20     return 0; }
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
Calificación del alumno 6 es 9
Calificación del alumno 7 es 890044423
Program ended with exit code: 0
```



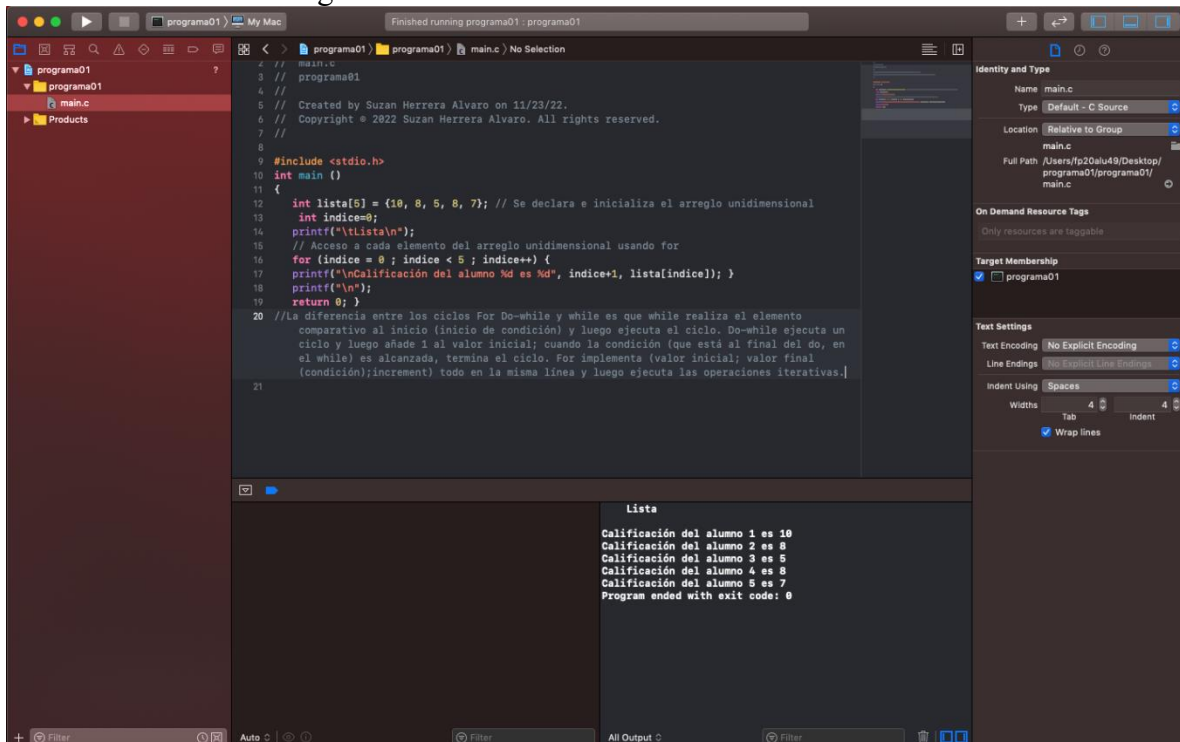
Código (apuntadores) Se observa el código de un programa que permite acceder a las

localidades de memoria de distintas variables a través de un apuntador.

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.

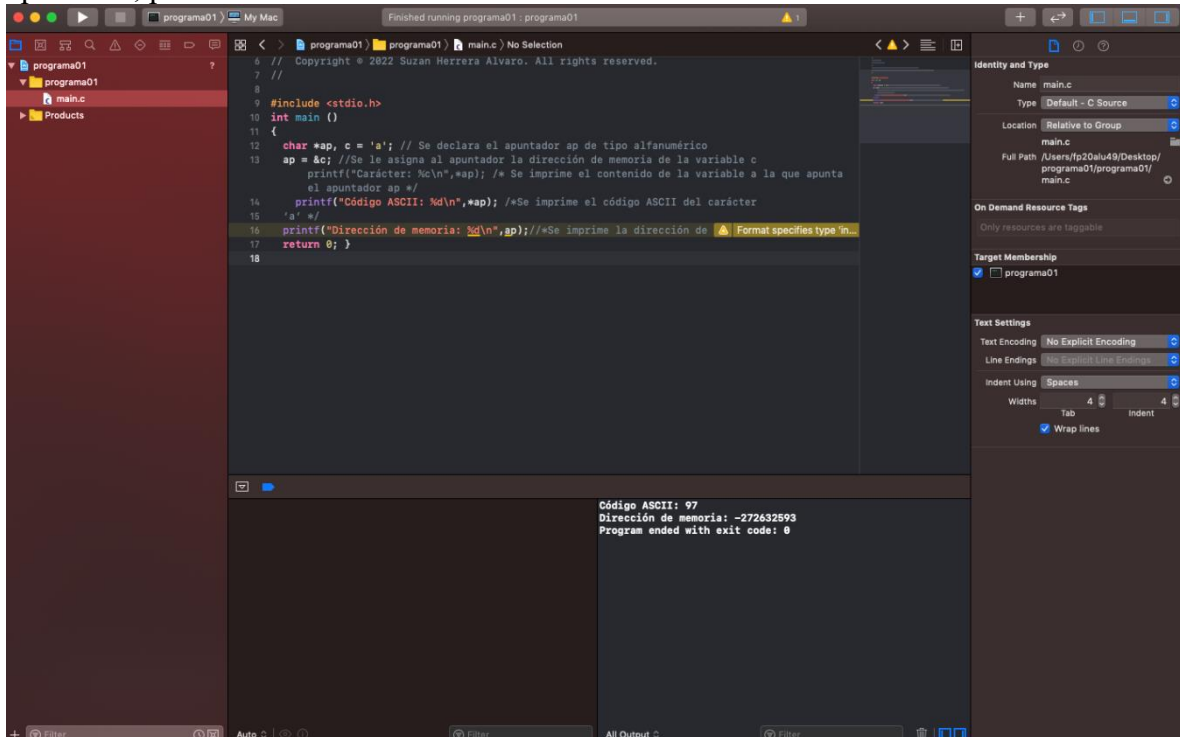


```
1 // main.c  
2 // programa01  
3  
4 //  
5 // Created by Suzan Herrera Alvaro on 11/23/22.  
6 // Copyright © 2022 Suzan Herrera Alvaro. All rights reserved.  
7 //  
8  
9 #include <stdio.h>  
10 int main ()  
11 {  
12     int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional  
13     int indice=0;  
14     printf("\tLista\n");  
15     // Acceso a cada elemento del arreglo unidimensional usando for  
16     for (indice = 0 ; indice < 5 ; indice++) {  
17         printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]); }  
18     printf("\n");  
19     return 0; }  
20 //La diferencia entre los ciclos For Do-while y while es que while realiza el elemento  
21 //comparativo al inicio (inicio de condición) y luego ejecuta el ciclo. Do-while ejecuta un  
22 //ciclo y luego añade 1 al valor inicial; cuando la condición (que está al final del do, en  
23 //el while) es alcanzada, termina el ciclo. For implementa (valor inicial; valor final  
24 //(condición);increment) todo en la misma línea y luego ejecuta las operaciones iterativas.]  
25
```

Lista

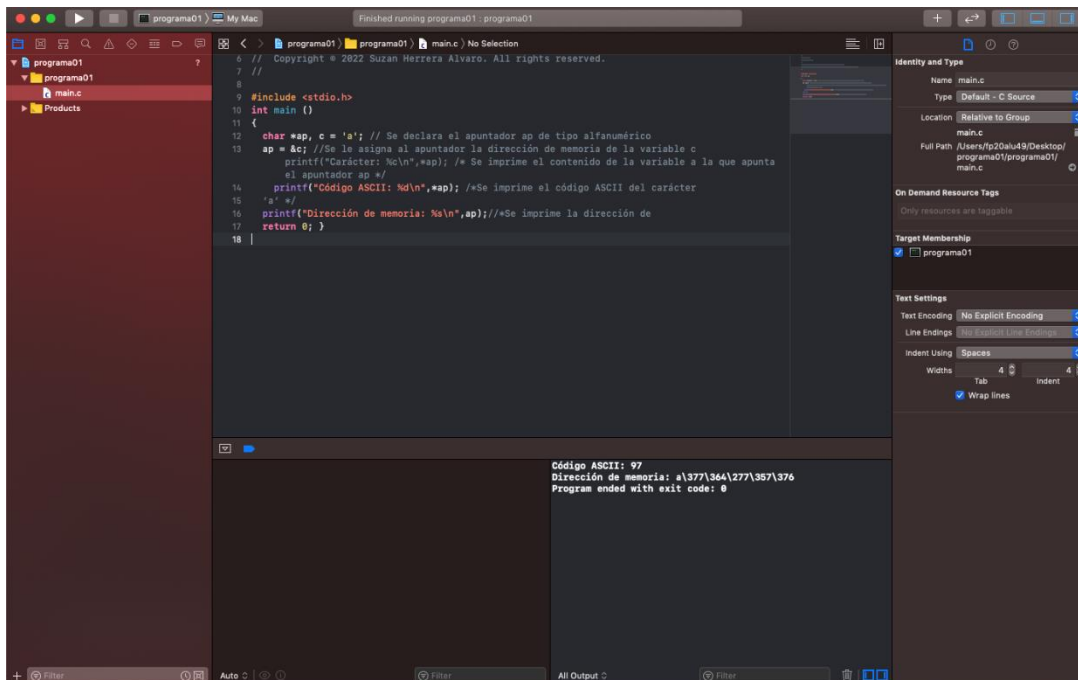
Calificación del alumno 1 es 10
Calificación del alumno 2 es 8
Calificación del alumno 3 es 5
Calificación del alumno 4 es 8
Calificación del alumno 5 es 7
Program ended with exit code: 0

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.



```
6 // Copyright © 2022 Suzan Herrera Alvaro. All rights reserved.
7 //
8
9 #include <stdio.h>
10 int main ()
11 {
12     char *ap, c = 'a'; // Se declara el apuntador ap de tipo alfanumérico
13     ap = &c; //Se le asigna al apuntador la dirección de memoria de la variable c
14     printf("Carácter: %c\n",*ap); /* Se imprime el contenido de la variable a la que apunta
15     el apuntador ap */
16     printf("Código ASCII: %d\n",*ap); /*Se imprime el código ASCII del carácter
17     'a' */
18     printf("Dirección de memoria: %d\n",ap);/*Se imprime la dirección de
19     return 0; }
```

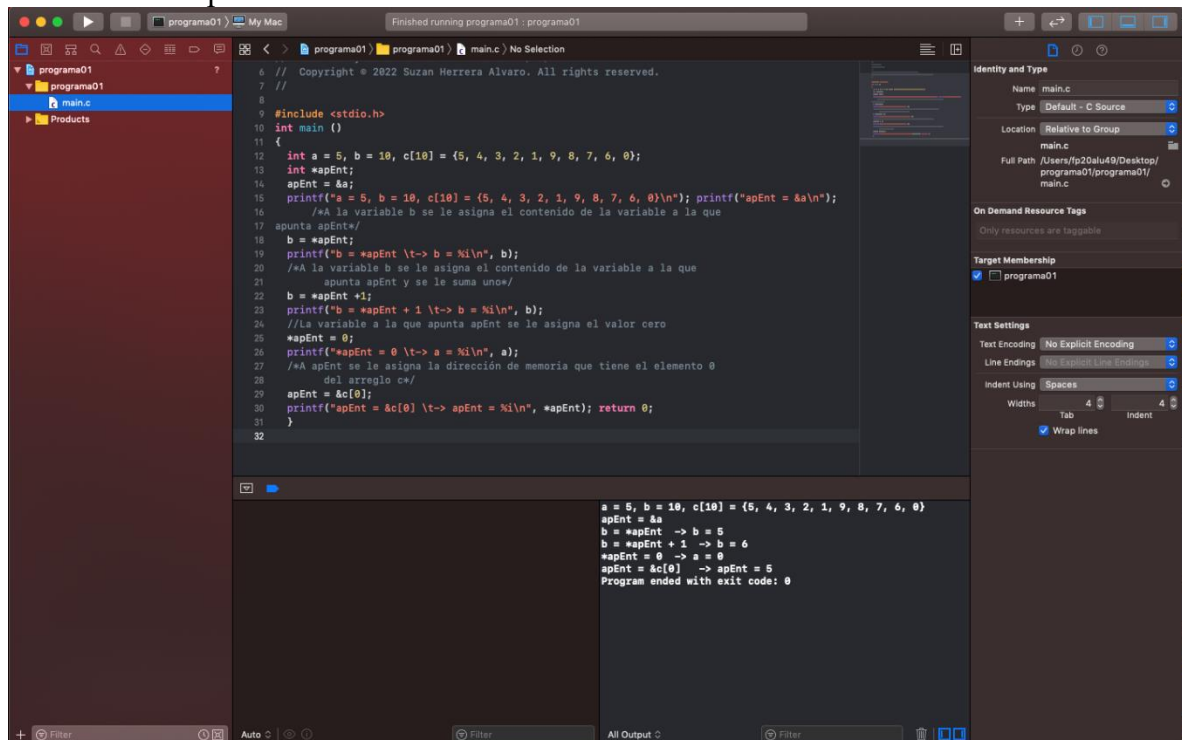
Código ASCII: 97
Dirección de memoria: -272632593
Program ended with exit code: 0



```
6 // Copyright © 2022 Suzan Herrera Alvaro. All rights reserved.
7 //
8
9 #include <stdio.h>
10 int main ()
11 {
12     char *ap, c = 'a'; // Se declara el apuntador ap de tipo alfanumérico
13     ap = &c; //Se le asigna al apuntador la dirección de memoria de la variable c
14     printf("Carácter: %c\n",*ap); /* Se imprime el contenido de la variable a la que apunta
15     el apuntador ap */
16     printf("Código ASCII: %d\n",*ap); /*Se imprime el código ASCII del carácter
17     'a' */
18     printf("Dirección de memoria: %d\n",ap);/*Se imprime la dirección de
19     return 0; }
```

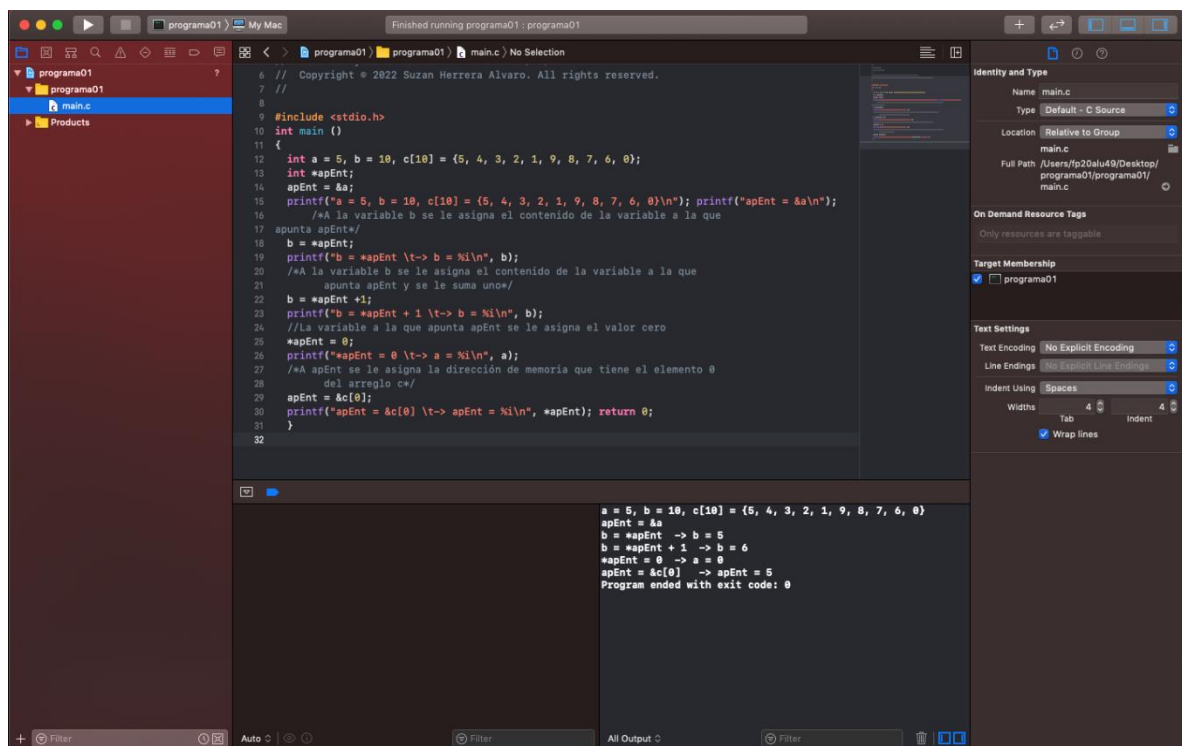
Código ASCII: 97
Dirección de memoria: a\377\364\277\357\376
Program ended with exit code: 0

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.



```
6 // Copyright © 2022 Suzan Herrera Alvaro. All rights reserved.
7 //
8
9 #include <stdio.h>
10 int main ()
11 {
12     int a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0};
13     int *apEnt;
14     apEnt = &a;
15     printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n"); printf("apEnt = %a\n");
16     /*A la variable b se le asigna el contenido de la variable a la que
17     apunta apEnt*/
18     b = *apEnt;
19     printf("b = *apEnt \t-> b = %i\n", b);
20     /*A la variable b se le asigna el contenido de la variable a la que
21     apunta apEnt y se le suma uno*/
22     b = *apEnt +1;
23     printf("b = *apEnt + 1 \t-> b = %i\n", b);
24     /*La variable a la que apunta apEnt se le asigna el valor cero
25     *apEnt = 0;
26     printf("apEnt = 0 \t-> a = %i\n", a);
27     /*A apEnt se le asigna la dirección de memoria que tiene el elemento 0
28     del arreglo c*/
29     apEnt = &c[0];
30     printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt); return 0;
31 }
32
```

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



```
6 // Copyright © 2022 Suzan Herrera Alvaro. All rights reserved.
7 //
8
9 #include <stdio.h>
10 int main ()
11 {
12     int a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0};
13     int *apEnt;
14     apEnt = &a;
15     printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n"); printf("apEnt = %a\n");
16     /*A la variable b se le asigna el contenido de la variable a la que
17     apunta apEnt*/
18     b = *apEnt;
19     printf("b = *apEnt \t-> b = %i\n", b);
20     /*A la variable b se le asigna el contenido de la variable a la que
21     apunta apEnt y se le suma uno*/
22     b = *apEnt +1;
23     printf("b = *apEnt + 1 \t-> b = %i\n", b);
24     /*La variable a la que apunta apEnt se le asigna el valor cero
25     *apEnt = 0;
26     printf("apEnt = 0 \t-> a = %i\n", a);
27     /*A apEnt se le asigna la dirección de memoria que tiene el elemento 0
28     del arreglo c*/
29     apEnt = &c[0];
30     printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt); return 0;
31 }
32
```

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

Código (apuntadores en ciclo do-while) El programa que 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 image displays two screenshots of an IDE, likely Visual Studio Code, showing a C program and its execution output. The code defines an array of 5 integers and uses a pointer to traverse it with a do-while loop. The output shows the memory addresses and values of the array elements.

Top Screenshot:

```
//
8
9 #include <stdio.h>
10 int main ()
11 {
12     int arr[] = {5, 4, 3, 2, 1};
13     int *apArr; //Se declara el apuntador apArr
14     int x;
15     apArr = arr;
16     printf("int arr[] = {5, 4, 3, 2, 1};\n");
17     printf("apArr = &arr[0]\n");
18     x = *apArr; //A la variable x se le asigna el contenido del arreglo arr en
19     su elemento 0*/
20     printf("x = *apArr \t -> x = %d\n", x);
21     x = *(apArr+1); //A la variable x se le asigna el contenido del arreglo arr
22     en su elemento 1*/
23     printf("x = *(apArr+1) \t -> x = %d\n", x);
24     x = *(apArr+2); //A la variable x se le asigna el contenido del arreglo arr
25     en su elemento 2*/
26     printf("x = *(apArr+2) \t -> x = %d\n", x);
27     x = *(apArr+3); //A la variable x se le asigna el contenido del arreglo arr
28     en su elemento 3*/
29     printf("x = *(apArr+3) \t -> x = %d\n", x);
30     x = *(apArr+4); //A la variable x se le asigna el contenido del arreglo arr
31     en su elemento 4*/
32     printf("x = *(apArr+4) \t -> x = %d\n", x); return 0;
33 }
34
```

Output:

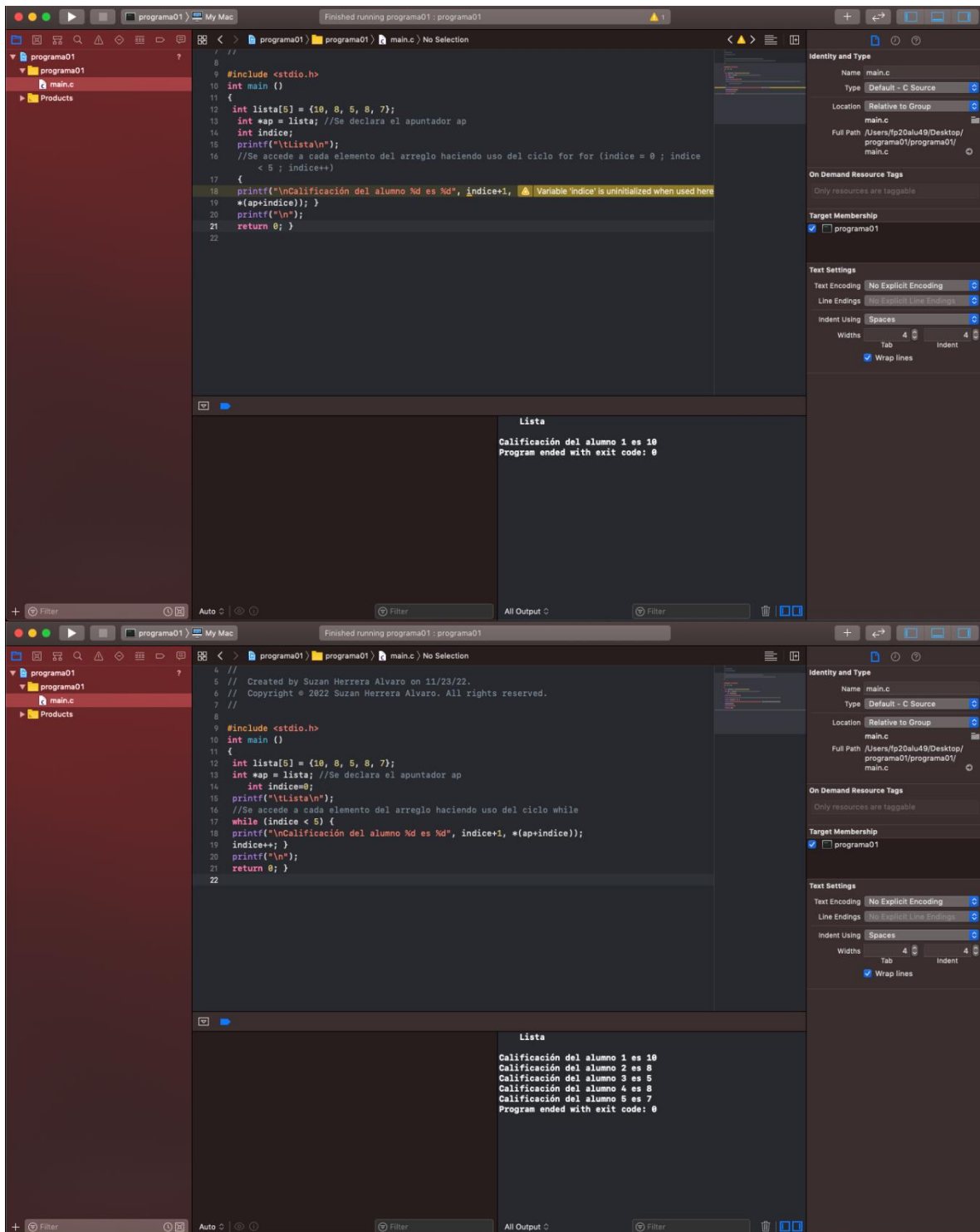
```
int arr[] = {5, 4, 3, 2, 1};
apArr = &arr[0]
x = *apArr -> x = 5
x = *(apArr+1) -> x = 4
x = *(apArr+2) -> x = 3
x = *(apArr+3) -> x = 2
x = *(apArr+4) -> x = 1
Program ended with exit code: 0
```

Bottom Screenshot:

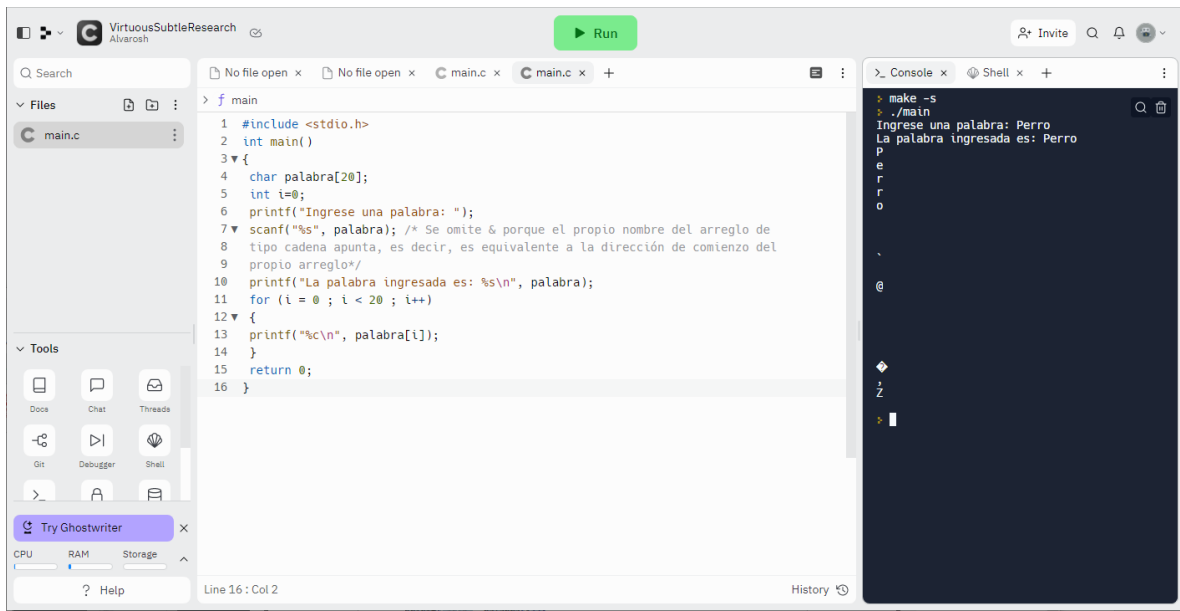
```
//
8
9 #include <stdio.h>
10 int main ()
11 {
12     int arr[] = {5, 4, 3, 2, 1};
13     int *apArr; //Se declara el apuntador apArr
14     int x;
15     apArr = arr;
16     printf("int arr[] = {5, 4, 3, 2, 1};\n");
17     printf("apArr = &arr[0]\n");
18     x = *apArr; //A la variable x se le asigna el contenido del arreglo arr en
19     su elemento 0*/
20     printf("x = *apArr \t -> x = %d\n", x);
21     x = *(apArr+1); //A la variable x se le asigna el contenido del arreglo arr
22     en su elemento 1*/
23     printf("x = *(apArr+1) \t -> x = %d\n", x);
24     x = *(apArr+2); //A la variable x se le asigna el contenido del arreglo arr
25     en su elemento 2*/
26     printf("x = *(apArr+2) \t -> x = %d\n", x);
27     x = *(apArr+3); //A la variable x se le asigna el contenido del arreglo arr
28     en su elemento 3*/
29     printf("x = *(apArr+3) \t -> x = %d\n", x);
30     x = *(apArr+4); //A la variable x se le asigna el contenido del arreglo arr
31     en su elemento 4*/
32     printf("x = *(apArr+2) \t -> x = %d\n", x); return 0;
33 }
34
```

Output:

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



Código (apuntadores en cadenas) El siguiente programa muestra el manejo básico de cadenas en lenguaje C.



The screenshot shows a C programming environment with a file named `main.c`. The code in the editor is as follows:

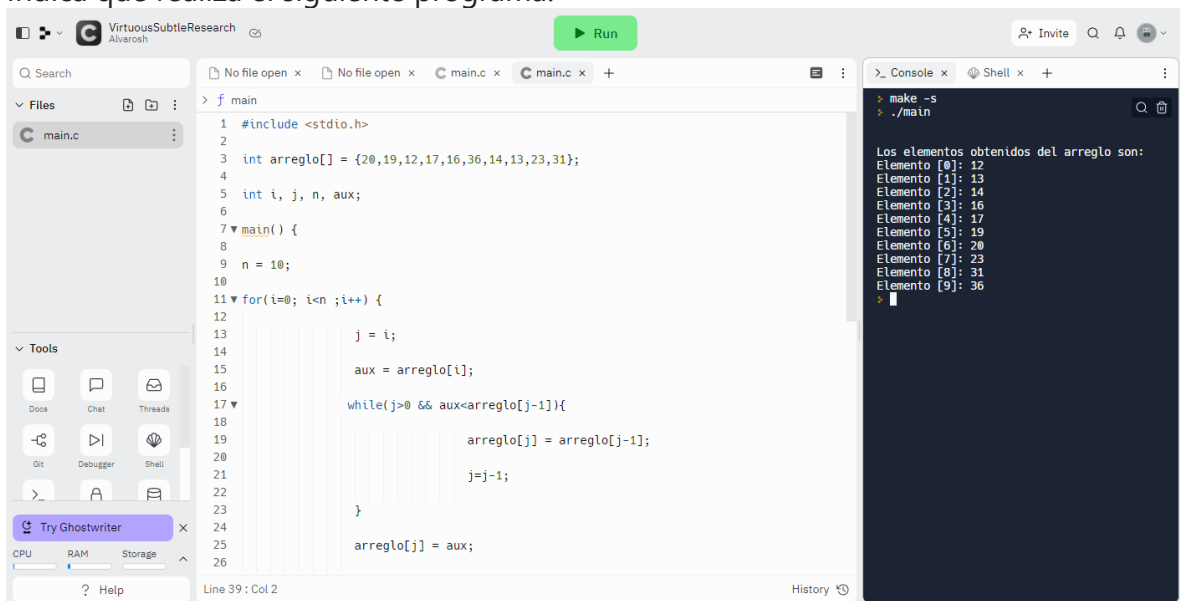
```
1 #include <stdio.h>
2 int main()
3 {
4     char palabra[20];
5     int i=0;
6     printf("Ingrese una palabra: ");
7     scanf("%s", palabra); /* Se omite & porque el propio nombre del arreglo de
8 tipo cadena apunta, es decir, es equivalente a la dirección de comienzo del
9 propio arreglo*/
10    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 }
```

The console output shows the program execution:

```
> make -s
> ./main
Ingrese una palabra: Perro
La palabra ingresada es: Perro
P
e
r
r
o
```

Ejercicios en casa

Indica que realiza el siguiente programa:



The screenshot shows a C programming environment with a file named `main.c`. The code in the editor is as follows:

```
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
11    for(i=0; i<n ;i++) {
12
13        j = i;
14
15        aux = arreglo[i];
16
17        while(j>0 && aux<arreglo[j-1]){
18
19            arreglo[j] = arreglo[j-1];
20
21            j=j-1;
22        }
23
24        arreglo[j] = aux;
25
26    }
```

The console output shows the program execution:

```
> make -s
> ./main

Los elementos obtenidos del arreglo son:
Elemento [0]: 12
Elemento [1]: 13
Elemento [2]: 14
Elemento [3]: 16
Elemento [4]: 17
Elemento [5]: 19
Elemento [6]: 20
Elemento [7]: 23
Elemento [8]: 31
Elemento [9]: 36
```

Da unos arreglos y al ejecutar el programa los ordena los datos de manera individual y por renglones.

Conclusión

En esta práctica conocí los arreglos y su funcionamiento, los puse en práctica para reforzar lo aprendido.

Fue una buena práctica no muy pesada y sobre todo útil. Usamos arreglos unidimensionales que es un tipo de dato que permite almacenar un conjunto de datos homogéneos, es decir, del mismo tipo de dato. Al declararse el arreglo debe indicarse el tamaño, en ese momento el compilador reserva la memoria que se necesite para almacenar los datos solicitados por el programador, en resumen.

Nos servirá y facilitará el trabajo en futuros programas esta práctica.

<https://github.com/3201050964/Pr-ctica-9>