



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorios de
docencia

Laboratorio de Computación

Salas A y B

Profesor(a): César Fabián Domínguez Velasco

Asignatura: Fundamentos de Programación

Grupo: 15

No de Práctica(s): 9

Integrante(s): Camacho Duarte Héctor Enrique

Gutiérrez Esquivel Giovani Emiliano

Flores Jiménez Diego

Moreno Chapan Amilet

Reyes García Raúl de Jesús

No. de lista 07,11,15,16,25,35.

o brigada:

Semestre: 2024-02

Fecha de entrega: 10/04/24

Observaciones:

CALIFICACIÓN:

Introducción:

Los arreglos son estructuras fundamentales en la programación que permiten almacenar conjuntos de datos contiguos del mismo tipo, con un tamaño fijo definido en el momento de su creación. Cada elemento en un arreglo se asocia con una posición específica, la cual se accede mediante el uso de índices. La eficiencia y la manipulación de datos de manera coherente son ventajas clave que ofrecen los arreglos en la construcción de programas.

Existen arreglos unidimensionales y multidimensionales, cuya dimensión está determinada por el número de índices requeridos para acceder a un elemento específico. Desde un solo índice para arreglos unidimensionales hasta múltiples índices para arreglos de mayor complejidad, el uso de estas estructuras es esencial para optimizar el código y gestionar datos de manera eficiente.

En esta práctica, nos centraremos en explorar y trabajar con los arreglos unidimensionales, comprendiendo su importancia y aplicaciones en el desarrollo de programas.

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.

Programa1a.c

```
C programa1a.c
C programa1a.c
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("\nLista\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
15    return 0;
16 }
```

```
Raúl@LAPTOP-53RCPAOK /cygdrive/c/Fundamentos
$ ls
programa1a.c  programa1a.exe

Rau1@LAPTOP-53RCPAOK /cygdrive/c/Fundamentos
$ gcc programa1a.c -o programa1a

Rau1@LAPTOP-53RCPAOK /cygdrive/c/Fundamentos
$ ./programa1a
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

Rau1@LAPTOP-53RCPAOK /cygdrive/c/Fundamentos
$
```

En este programa se muestra el código de un arreglo unidimensional de 5 elementos que para poder acceder, recorrer y mostrar cada elemento del arreglo se usa la variable índice haciendo uso de un ciclo while.

Programa1b.c

```
C programa1b.c
C programa1b.c
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     do // Acceso a cada elemento del arreglo unidimensional usando do-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    while (indice < 5 );
13    printf("\n");
14    return 0;
15 }
16 
```

```
/cygdrive/c/Fundamentos
Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ ls
programala.c programala.exe programa1b.c

Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ gcc programa1b.c -o programa1b

Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ ./programa1b
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

Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ 
```

Como en el anterior programa , se muestra el código de un arreglo unidimensional de 5 elementos que para poder acceder, recorrer y mostrar cada elemento del arreglo se usa la variable índice pero esta vez haciendo uso de un ciclo do-while.

Programa1c.c

```
C programa1c.c
C programa1c.c
1 #include <stdio.h>
2 int main ()
3 {
4
5     int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional
6     int indice=0;
7     printf("\tLista\n");
8     // Acceso a cada elemento del arreglo unidimensional usando for
9     for (indice = 0 ; indice < 5 ; indice++)
10    {
11        printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
12    }
13    printf("\n");
14    return 0;
15 } 
```

```
/cygdrive/c/Fundamentos
Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ ls
programala.c programala.exe programa1b.c programa1b.exe programa1c.c

Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ gcc programa1c.c -o programa1c

Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ ./programa1c
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

Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ | 
```

Finalmente, se muestra el código de un arreglo unidimensional de 5 elementos que para poder acceder, recorrer y mostrar cada elemento del arreglo se usa la variable índice haciendo uso de un ciclo for.

Programa2.c

```
C programa2.c X
C programa2.c
1 #include <stdio.h>
2 int main ()
3 {
4
5     int lista[10]; // Se declara el arreglo unidimensional
6     int indice=0;
7     int numeroElementos=0;
8     printf("\nDa un número entre 1 y 10 para indicar la cantidad de elementos que tiene el arreglo\n");
9     scanf("%d",&numeroElementos);
10    if((numeroElementos>=1) && (numeroElementos<=10))
11    {
12        // Se almacena un número en cada elemento del arreglo unidimensional usando for
13        for (indice = 0 ; indice <= numeroElementos-1 ; indice++)
14        {
15            printf("\nDar un número entero para el elemento %d del arreglo", indice );
16            scanf("%d",&lista[indice]);
17        }
18        printf("\nLos valores dados son: \n");
19        // Se muestra el número almacenado en cada elemento del arreglo unidimensional usando for
20        for (indice = 0 ; indice <= numeroElementos-1 ; indice++)
21        {
22            printf("%d ", lista[indice] );
23        }
24    }
25    else printf("el valor dado no es válido");
26    printf("\n");
27    return 0;
28 }
29 |
```

```
/cygdrive/c/Fundamentos
Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ ls
programa1.c  programa1b.c  programalc.c  programa2.c
programa1.exe programa1b.exe  programalc.exe  programa2.exe
Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ gcc programa2.c -o programa2
Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ ./programa2
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 10
Dar un número entero para el elemento 1 del arreglo 4
Dar un número entero para el elemento 2 del arreglo 7
Dar un número entero para el elemento 3 del arreglo 6
Dar un número entero para el elemento 4 del arreglo 1
Los valores dados son:
10 4 7 6 1
Raúl@LAPTOP-5JRCPAOK /cygdrive/c/Fundamentos
$ |
```

En este programa se 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 haciendo uso de un ciclo for.

Programa 3.c

The screenshot shows a Mac desktop with a dark theme. In the center is a terminal window titled "Documentos -- bash -- 94x33". The code in the terminal is:

```
[Australis11:DOCUMENTS fp15alu07$ gcc programa3.c -o programa3
[Australis11:DOCUMENTS fp15alu07$ ./programa3
Carácter: a
Código ASCII: 97
Dirección de memoria: -1205233801
Australis11:DOCUMENTS fp15alu07$ ]
```

Below the terminal are several icons for system applications like Finder, Calendar, Mail, and Safari.

```
C programa3.c
1 #include <stdio.h>
2 int main ()
3 {
4     char *ap, c = 'a';
5     ap = &c;
6     printf("Carácter: %c\n",*ap);
7     printf("Código ASCII: %d\n",*ap);
8     printf("Dirección de memoria: %d\n",ap);
9     return 0;
10 }
```

En este programa a través de apunadores permite acceder a las localidades de memoria de distintas variables por ejemplo: Se imprime el código ASCII del carácter 'a' que en ASCII es 97.

Programa4. C

The screenshot shows a Mac desktop with a dark theme. In the center is a terminal window titled "Documentos -- bash -- 94x33". The code in the terminal is:

```
[Australis11:DOCUMENTS fp15alu07$ gcc programa3.c -o programa3
[Australis11:DOCUMENTS fp15alu07$ ./programa3
Carácter: a
Código ASCII: 97
Dirección de memoria: -1205233801
[Australis11:DOCUMENTS fp15alu07$ 
[Australis11:DOCUMENTS fp15alu07$ gcc programa3.c -o programa3
[Australis11:DOCUMENTS fp15alu07$ ./programa3
Carácter: a
Código ASCII: 97
Dirección de memoria: 0x7ff7b8cccd777
[Australis11:DOCUMENTS fp15alu07$ gcc programa4.c -o programa4
[Australis11:DOCUMENTS fp15alu07$ ./programma4
a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}
apEnt = &a;
b = *apEnt
b = *apEnt + 1 -> b = 4
*apEnt = 0;
printf("*apEnt = 0 \t-> a = %i\n", a);
apEnt = &c[0];
printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt); return 0;
}
```

Below the terminal are several icons for system applications like Finder, Calendar, Mail, and Safari.

```
EXPLORER
... Welcome C programa1a.c C programa2a.c C programa3.c C programa4.c X ⓘ ...
C programa4.c
1 #include<stdio.h>
2 int main ()
3 {
4     int a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0};
5     int *apEnt;
6     apEnt = &a;
7     printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n");
8     printf("apEnt = &a\n");
9     b = *apEnt;
10    printf("b = *apEnt \t-> b = %i\n", b);
11    b = *apEnt +1;
12    printf("b = *apEnt + 1 \t-> b = %i\n", b);
13    *apEnt = 0;
14    printf("*apEnt = 0 \t-> a = %i\n", a);
15    apEnt = &c[0];
16    printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt); return 0;
}
```

Este programa permite acceder a las localidades de memoria de distintas variables a través de un apuntador y finalmente imprimimos las localidades de memoria utilizando los apuntadores.

Programa5.C

The screenshot shows a Mac OS X desktop environment. In the foreground, a terminal window titled "Documentos -- bash -- 81x12" displays the output of a C program named "programa5". The program prints the values of an integer array "arr" at different memory locations. The terminal window has a dark background with light-colored text. In the background, a code editor window titled "programa5.c" is open, showing the source code for the program. The code includes comments explaining the pointer arithmetic used to access each element of the array.

```
programa5.c — Documents
... programa1a.c C programa2a.c C programa3.c C programa4.c C programa5.c ... APRENDA

programa5.c
1 #include <stdio.h>
2 main ()
3 {
4     int arr[] = {5, 4, 3, 2, 1};
5     int *apArr; /*Se declara el apuntador apArr*/
6     int x;
7     apArr = arr;
8     printf("int arr[] = {5, 4, 3, 2, 1};\n");
9     printf("apArr = &arr[0]\n");
10    x = *apArr; /*A la variable x se le asigna el contenido del arreglo arr en
11    elemento 0*/
12    printf("x = *apArr \t -> x = %d\n", x);
13    x = *(apArr+1); /*A la variable x se le asigna el contenido del arreglo arr
14    en su elemento 1*/
15    printf("x = *(apArr+1) \t -> x = %d\n", x);
16    x = *(apArr+2); /*A la variable x se le asigna el contenido del arreglo arr
17    en su elemento 2*/
18    printf("x = *(apArr+2) \t -> x = %d\n", x);
19    x = *(apArr+3); /*A la variable x se le asigna el contenido del arreglo arr
20    en su elemento 3*/
21    printf("x = *(apArr+3) \t -> x = %d\n", x);
22    x = *(apArr+4); /*A la variable x se le asigna el contenido del arreglo arr
23    en su elemento 4*/
24    printf("x = *(apArr+4) \t -> x = %d\n", x); return 0;
25 }

[ Australia11:DOCUMENTS fp15alu07$ gcc programa5.c -o programa5
Australia11:DOCUMENTS fp15alu07$ ./programa5
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
Australia11:DOCUMENTS fp15alu07$ ]
```

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

Programa6a.c

```
C problema6a.c > main()
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;
7     printf("\tLista\n");
8     //se accede a cada elemento del arreglo haciendo uso del ciclo for
9     for (indice = 0 ; indice < 5 ; indice++)
10    {
11        printf("\nCalificación del alumno %d es %d", indice+1,
12              *(ap+indice));
13    }
14    printf("\n");
15    return 0;
16 }
```

```
/cygdrive/c/FP
Lista
5

bpsom@Hope /cygdrive/c/FP
$ gcc problema6a.c -o problema6a.exe
bpsom@Hope /cygdrive/c/FP
$ ./problema6a.exe
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
bpsom@Hope /cygdrive/c/FP
$
```

En este programa se genera un arreglo unidimensional de 5 elementos y se accede a cada elemento del arreglo a través de un apuntador utilizando un ciclo do while. Y imprime una lista de calificaciones de 5 alumnos distintos mostrando su respectiva calificación.

Programa 7.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 de propio arreglo*/
    printf("La palabra ingresada es: %s\n", palabra);
    for (i = 0 ; i < 20 ; i++)
    {
        printf("%c\n", palabra[i]);
    }
    return 0;
```

```
S /cygdrive/c/Fundamentos/Problemas adicionales
ineff@Inefable ~
$ cd /cygdrive/c/Fundamentos/Problemas\ adicionales/
ineff@Inefable /cygdrive/c/Fundamentos/Problemas adicionales
$ gcc Programa7.c -o Programa7.exe
ineff@Inefable /cygdrive/c/Fundamentos/Problemas adicionales
$ ./Programa7
Ingrese una palabra: parangaricutirimicuaro
La palabra ingresada es: parangaricutirimicuaro
p
a
r
a
n
g
a
r
i
c
u
t
i
```

Problemas propuestos

Ejercicio adicional 1

```
#include <stdio.h>

int main() {
    float arreglo[8];

    printf("Ingrese 8 valores:\n");
    for (int i = 0; i < 8; i++) {
        scanf("%f", &arreglo[i]);
    }

    printf("Suma de indices:\n");
    for (int i = 0; i < 7; i += 2) {
        float suma = arreglo[i] + arreglo[i + 1];
        printf("Indice %d y %d: %.2f\n", i, i + 1, suma);
    }

    return 0;
}
```

```
dfj_2@DESKTOP-PTC23KV ~
$ cd /cygdrive/c/fp

dfj_2@DESKTOP-PTC23KV /cygdrive/c/fp
$ gcc estRep.c -o estRep

dfj_2@DESKTOP-PTC23KV /cygdrive/c/fp
$ ./arrayPro.exe
./arrayPro.exe: No such file or directory

dfj_2@DESKTOP-PTC23KV /cygdrive/c/fp
$ ./estRep.exe
Ingrese 8 valores flotantes:
1
2
3
4
5
6
7
8
Suma de índices por pares:
Indice 0 y 1: 3.00
Indice 2 y 3: 7.00
Indice 4 y 5: 11.00
Indice 6 y 7: 15.00
```

En este programa haciendo el uso de dos ciclos for, se solicita al usuario que ingrese 8 valores flotantes (decimales) para agregarlos en un array que posteriormente se sumarán en parejas por índice en este mismo haciendo uso de nuestro segundo ciclo for

Ejercicio adicional 2

Con uso de apuntadores

```
#include <stdio.h>

int main() {
    char arreglo[5];
    char *ptr = arreglo; //apuntador y asignación a la dirección del primer elemento del array

    printf("Ingrese 5 caracteres:\n");
    for ( ; ptr < arreglo + 5; ptr++) {
        scanf(" %c", ptr);
    }

    printf("Valores ASCII de los caracteres:\n");
    for (ptr = arreglo; ptr < arreglo + 5; ptr++) {
        printf("Carácter '%c': %d\n", *ptr, *ptr);
    }

    return 0;
}
```



```
dfj_2@DESKTOP-PTC23KV /cygdrive/c/fp
$ gcc apuntadores.c -o apuntadores

dfj_2@DESKTOP-PTC23KV /cygdrive/c/fp
$ ./apuntadores.exe
Ingrese 5 caracteres:
a
b
c
d
e
Valores ASCII de los caracteres:
Carácter 'a': 97
Carácter 'b': 98
Carácter 'c': 99
Carácter 'd': 100
Carácter 'e': 101

dfj_2@DESKTOP-PTC23KV /cygdrive/c/fp
$
```

Sin uso de apuntadores

```
#include <stdio.h>

int main() {
    char arreglo[5];

    printf("Ingrese 5 caracteres:\n");
    for (int i = 0; i < 5; i++) {
        scanf(" %c", &arreglo[i]);
    }

    printf("Valores ASCII de los caracteres:\n");
    for (int i = 0; i < 5; i++) {
        printf("Carácter '%c': %d\n", arreglo[i], arreglo[i]);
    }

    return 0;
}
```



```
/cygdrive/c/fp  
  
dfj_2@DESKTOP-PTC23KV /cygdrive/c/fp  
$ gcc apuntadores.c -o apuntadores  
  
dfj_2@DESKTOP-PTC23KV /cygdrive/c/fp  
$ ./apuntadores.exe  
Ingrese 5 caracteres:  
a  
b  
c  
d  
e  
Valores ASCII de los caracteres:  
Carácter 'a': 97  
Carácter 'b': 98  
Carácter 'c': 99  
Carácter 'd': 100  
Carácter 'e': 101  
  
dfj_2@DESKTOP-PTC23KV /cygdrive/c/fp
```

Este programa haciendo el uso de estructuras de repetición se solicita al usuario introducir 5 caracteres que posteriormente se mostraran en código ASCII, igualmente haciendo uso de apuntadores en nuestro programa

Conclusión

La comprensión de los arreglos unidimensionales y su aplicación en el desarrollo de programas es fundamental para cualquier programador. A través de esta práctica, hemos explorado cómo los arreglos permiten almacenar datos de manera eficiente y coherente, facilitando el acceso a ellos mediante el uso de índices. Además, hemos destacado la importancia de los arreglos como herramientas para optimizar el código y manipular conjuntos de datos con significado común.

Al dominar los conceptos y técnicas relacionadas con los arreglos unidimensionales, podemos mejorar la eficiencia y la claridad de los programas, lo que resulta en un código más legible, y eficiente.

Es por ello que consideramos se cumplieron los objetivos propuestos los cuales indicaban que elaboraramos programas con arreglos de una dimensión.

Bibliografía

Laboratorio Salas A y B. (n.d.). <http://lcp02.fi-b.unam.mx/>

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