	Arreglos unidimensionales y multidimensionales	
Facultad de Ingeniería	Laboratorio de docencia	

Laboratorios de computación
salas A y B

Profesor: Alejandro Pimentel

Asignatura: Fundamentos de Programación

Grupo: 135

No de Práctica(s): 11

Integrante(s): Lorena Basurto Amezcu

*No. de Equipo de
cómputo empleado:* Mónaco 21

No. de Lista o Brigada: 2858

Semestre: 2020-1

Fecha de entrega: Octubre 28, 2019.

Observaciones: Muy bien

CALIFICACIÓN: **10**

Objetivo:

Reconocer la importancia y utilidad de los arreglos, en la elaboración de programas que resuelvan problemas que requieran agrupar datos del mismo tipo, así como trabajar con arreglos tanto unidimensionales como multidimensionales.

Desarrollo:

- *Arreglos unidimensionales*

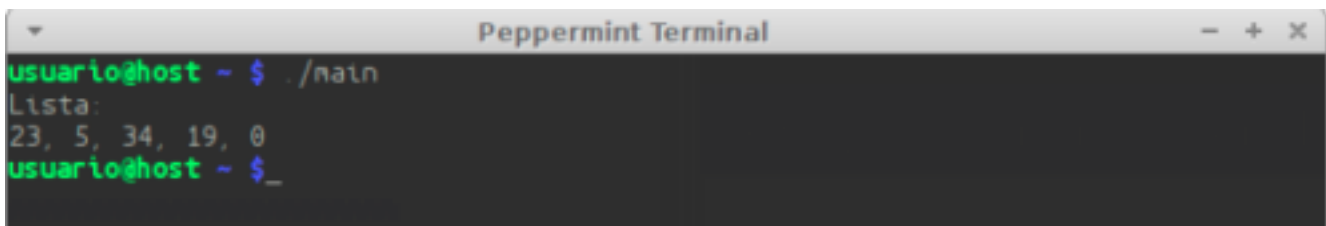
```
#include <stdio.h>
#define TAMANO 5

int main(int argc, char *argv[])
{
    int lista[TAMANO] = {23, 5, 34, 19, 0};

    printf("Lista:\n");
    for(int i=0; i< TAMANO-1; i++){
        printf("%i, ",lista[i]);
    }

    printf("%i\n",lista[TAMANO-1]);

    return 0;
}
```



```
Peppermint Terminal
usuario@host ~ $ ./main
Lista:
23, 5, 34, 19, 0
usuario@host ~ $ _
```

- *Arreglos multidimensionales*

```
#include <stdio.h>
#define DIM 3

int main(int argc, char *argv[])
{
    int matriz[DIM][DIM] = {{23, 5, 34},
                             { 8, 46, 22},
                             { 3, 9, 12}};

    printf("Matriz:\n");
    for(int i=0; i < DIM; i++){
        for(int j=0; j < DIM ; j++){
            printf("%i\t",matriz[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

```
Peppermint Terminal
usuario@host ~ $ ./main
Lista:
23, 5, 34, 19, 8
usuario@host ~ $ ./main
Matriz:
23    5    34
8     46   22
3     9    12
usuario@host ~ $ _
```

- *Actividad 1*

Hacer un programa que:

- Pida al usuario un número.
- Genere un arreglo de esa longitud.
- Pida al usuario números suficientes para llenar el arreglo.
- Muestre al usuario el número menor y el mayor de dicho arreglo.

```
actividad2.c actividad1.c
1  #include <stdio.h>
2
3  int main ( int argc, char *argv[]) {
4
5      int n, x;
6
7      printf("Ingresa un numero entero: ");
8      scanf("%d", &n);
9
10     int arreglo[n];
11
12     printf("Ahora ingresa %d numeros enteros: \n", n);
13
14     for(int i=0; i<n; i++) {
15         scanf("%d", &x);
16         arreglo[i] = x;
17     }
18
19     printf("\nArreglo:\n[");
20
21     for(int i=0; i<n; i++) {
22         printf(" %d ", arreglo[i]);
23     }
24     printf("]\n");
25
26     int mayor = arreglo[0];
27
28     for(int i=1; i<n; i++) {
29         if(arreglo[i]>mayor) {
30             mayor = arreglo[i];
31         }
32     }
33
34     int menor = arreglo[0];
35
36     for(int i=1; i<n; i++) {
37         if(arreglo[i]<menor) {
38             menor = arreglo[i];
39         }
40     }
41
42     printf("El numero mayor del arreglo es: %d\n", mayor);
43     printf("El numero menor del arreglo es: %d\n", menor);
44
45
46     return 0;
47 }
48 }
```

```
Ingresa un numero entero: 5
Ahora ingresa 5 numeros enteros:
1
2
3
4
5

Arreglo:
[ 1 2 3 4 5 ]
El numero mayor del arreglo es: 5
El numero menor del arreglo es: 1
```

```
Ingresa un numero entero: 9
Ahora ingresa 9 numeros enteros:
-5
-33
92
16547
-298
0
67
100
6

Arreglo:
[ -5 -33 92 16547 -298 0 67 100 6 ]
El numero mayor del arreglo es: 16547
El numero menor del arreglo es: -298
```

- *Actividad 2*
 - Pida al usuario dos números N y M .
 - Genere dos matrices de $N \times M$.
 - Pida al usuario números suficientes para llenar ambas matrices.
 - Muestre al usuario la matriz resultado de sumar las dos de entrada.

```
actividad2.c  actividad1.c  X
1  #include <stdio.h>
2
3  int main ( int argc, char *argv[]) {
4
5      int n, m, x;
6
7      printf("Ingresa dos numeros enteros positivos: ");
8      scanf("%d %d", &n, &m);
9
10     int matrix1[n][m];
11     int matrix2[n][m];
12     int matrixSuma[n][m];
13
14     printf("Ahora ingresa %d numeros enteros positivos para generar la primera matriz: \n", n*m);
15
16     for(int j=0; j<m; j++) {
17         for(int i=0; i<n; i++) {
18             scanf("%d", &x);
19             matrix1[i][j] = x;
20         }
21     }
22
23     printf("Ahora ingresa otros %d numeros enteros positivos para generar la segunda matriz: \n", n*m);
24
25     for(int j=0; j<m; j++) {
26         for(int i=0; i<n; i++) {
27             scanf("%d", &x);
28             matrix2[i][j] = x;
29         }
30     }
31
32     printf("\nMatriz %dx%d que resulta de la suma de ambas matrices:\n", n, m);
33
34     for(int j=0; j<m; j++) {
35         for(int i=0; i<n; i++) {
36             matrixSuma[i][j] = matrix1[i][j] + matrix2[i][j];
37         }
38     }
39
40     for(int j=0; j<m; j++) {
41         for(int i=0; i<n; i++) {
42             printf("%d\t", matrixSuma[i][j]);
43         }
44         printf("\n");
45     }
46
47     return 0;
48 }
```

```
Ingresa dos numeros enteros positivos: 2 4
Ahora ingresa 8 numeros enteros positivos para generar la primera matriz:
1
1
2
2
3
3
4
4
Ahora ingresa 8 numeros enteros positivos para generar la segunda matriz:
1
1
2
2
3
3
4
4
Matriz 2x4 que resulta de la suma de ambas matrices:
2      2
4      4
6      6
8      8
```

```
Ingresa dos numeros enteros positivos: 5 2
Ahora ingresa 10 numeros enteros positivos para generar la primera matriz:
1
1
1
1
1
2
2
2
2
2
Ahora ingresa 10 numeros enteros positivos para generar la segunda matriz:
5
5
5
5
5
7
7
7
7
7
Matriz 5x2 que resulta de la suma de ambas matrices:
6      6      6      6      6
9      9      9      9      9
```

Conclusiones:

Los arreglos unidimensionales y multidimensionales son de gran utilidad en la elaboración de programas en C, y en cualquier otro lenguaje de programación, que resuelvan problemas que requieran agrupar datos del mismo tipo.