	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Alejandro Esteban Pimentel Alarcón

Asignatura: Fundamentos de Programación

Grupo: 03

No de Práctica(s): 11

Integrante(s): Cureño Arvizu Ameyalli

No. de Equipo de cómputo empleado:

No. de Lista o Brigada: 0779

Semestre: 2020 - 1

Fecha de entrega: 04 de noviembre de 2019

Observaciones:

CALIFICACIÓN: _____

ARREGLOS UNIDIMENSIONALES Y MULTIDIMENSIONALES

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.

¿Qué es un arreglo unidimensional?
Un arreglo unidimensional es un tipo de datos estructurado que está formado por una colección finita y ordenada de datos del mismo tipo. Es la estructura natural para modelar listas de elementos iguales. Los datos que se guarden en los arreglos todos deben ser del mismo tipo.

Lenguaje C permite crear arreglos de varias dimensiones con la siguiente sintaxis:
tipoDato nombre[tamaño][tamaño]…[tamaño];
Donde nombre se refiere al identificador del arreglo, tamaño es un número entero y define el número máximo de elementos que puede contener el arreglo por dimensión (el número de dimensiones está determinado por el número de corchetes).
Los tipos de dato que puede tolerar un arreglo multidimensional son: entero, real, carácter o estructura. De manera práctica se puede considerar que la primera dimensión corresponde a los renglones, la segunda a las columnas, la tercera al plano, y así sucesivamente. Sin embargo, en la memoria cada elemento del arreglo se guarda de forma contigua, por lo tanto, se puede recorrer un arreglo multidimensional con apuntadores

<u>Actividad 1</u>
Hacer un programa que pida:

- Pida al usuario un número, genere un arreglo de esa longitud, pida al usuario los números suficientes para llenar el arreglo, muestre al usuario el número menor y el mayor de dicho arreglo.

En este programa con el primer *for* llenamos la lista. Luego declaramos nuestra variable *menor*. Por consiguiente, le asignamos un valor a *menor*, suponiendo que el primer elemento de la lista es el numero menor.
El segundo *for* es para obtener el numero menor.
Y ahora con el if vamos a comparar el siguiente elemento de la lista (porque i=1) con el numero menor relativo que fue el primero (que la primera vez es lista[0]).
Una vez que ya realizamos eso si el siguiente elemento es menor éste pasará a ser el nuevo número menor relativo.
Y seguirá así hasta que se comparen todos los elementos de la lista.
Por ultimo mostramos en pantalla el resultado del numero mayor y menor obtenidos.

Ya que realizamos todo lo anterior y comprobamos que funcionara, éstos fueron nuestros resultados:

Actividad 2

Hacer un programa que:

- Pida al usuario dos números N y M, genere dos matrices de N x M, pida al usuario números suficientes para llenar ambas matrices, muestre al usuario la matriz resultado de sumar las dos de entrada.

tablas.c promedio.c a2.c

```c
#include<stdio.h>

int main(){
    int x,y;
    printf("Ingrese columnas de la matriz\n");
    scanf("%i",&x);
    printf("Ingrese filas de la matriz\n");
    scanf("%i",&y);

    int m1[x][y];
    int m2[x][y];
    int m3[x][y];
    printf("Llenar la Matriz 1\n");
    printf("\n");
    for(int a=0;a<x;a++){

        for (int b=0;b<y;b++){
            printf("lugar[%i][%i]\n",a+1,b+1);
            scanf("%i",&m1[a][b]);
        }

    }
 printf("\n");

    printf("Llenar la Matriz 2\n");

    for (int a=0;a<x;a++){

        for (int b=0;b<y;b++){
            printf("lugar[%i][%i]\n",a+1,b+1);
            scanf("%i",&m2[a][b]);
        }
```

```c
34
35
36      }
37       printf("\n");
38
39
40       printf("La suma de las dos matrices es\n");
41
42
43
44       for (int a=0;a<x;a++){
45
46          for (int b=0;b<y;b++){
47              m3[a][b]= m1[a][b]+ m2[a][b];
48              printf("%i\t",m3[a][b]);
49          }
50
51        printf("\n");
52
53       }
54
55       return 0;
56  }
```

En este programa lo que hacemos de inicio, es declarar las variables de matrices: 1, 2 y resultado, luego le indicamos al usuario que ingrese los números para rellenar el arreglo numero 1 y eso es posible gracias al primer *for e*l cual tiene dentro otro *for* que de igual manera nos ayudará a rellenar el segundo arreglo y el tercer *for* tiene como función sumar e imprimir el resultado de la suma.

Conclusión:

Para concluir, hemos observado que los for tienen una gran utilidad en este tipo de arreglos, o al menos son una forma de hacerlos más eficientes o más prácticos ya que precisamente, como lo habíamos visto en actividades pasadas los *for* nos ayudan a crear arreglos o listas.

En este caso los arreglos son más complicados, ya que estamos trabando con matrices, pero solo fue necesario aclarar bien las ideas, pensar que teníamos que hacer y generar una buena estructura de cada programa sin complicarnos tanto.