	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:

No. de Práctica: 10

Integrantes: Ever Ivan Rosales Gómez

No de Lista o brigade:

Semestre: 2023-2

Fecha de entrega: 24/Mayo/2023
Observaciones:

Calificación

Arreglos multidimensionales

OBJETIVO:

Elaborar programas que requieran agrupar datos del mismo tipo con ayuda de arreglos de dos dimensiones.

INTRODUCCIÓN:

Para comenzar es importante mencionar que es un arreglo. Un arreglo o también llamado array son conjunto de datos del mismo tipo, es decir que todos los elementos deben ser enteros, flotantes, carácter, cadena, dependiendo lo que queramos presentar.
Se usan los arreglos para facilitar el programa ya que se utilizan variables del mismo tipo y con el mismo propósito.

En esta práctica ahora se usarán arreglos bidimensionales, es decir, de dos dimensiones también llamadas matrices de dos datos. Para acceder se requiere el uso de dos índices, comúnmente se encontrarán como i,j.
La letra i para el renglón del arreglo y la letra j para la columna del arreglo.

En esta práctica es importante la forma en que se crea la variable que es de la siguiente forma:

TIPO DE DATO	NOMBRE	DIMENSIÓN	;
int	*matriz*	*[2][2]*	*;*

Con la tabla anterior se conoce que es un arreglo bidimensional por los dos paréntesis que en ese caso tienen dimensión de 2.

En caso de conocer los valores que van a tomar la dimensión se pueden declarar de la siguientes manera

TIPO DE DATO	NOMBRE	DIMENSIÓN	VALORES	;
int	*matriz*	*[ ][ ]*	*{(2,2).(2,2)}*	*;*

PROGRAMA 1

```
#include<stdio.h>
int main() {
int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
int i, j;
printf("Imprimir Matriz\n");
for (i=0 ; i<3 ; i++) //Representa al renglón del arreglo
{
        for (j=0 ; j<3 ; j++)//Representa a la columna del arreglo
{
printf("%d, ",matriz[i][j]); }
printf("\n"); }
return 0;
}
~
~
~
```

```
Irlanda58:~ fp24alu35$ gcc prog1.c -o prog1.out
Irlanda58:~ fp24alu35$ ./prog1.out
Imprimir Matriz
1, 2, 3,
4, 5, 6,
7, 8, 9,
Irlanda58:~ fp24alu35$
```

Este programa tiene valores ya definidos para la matriz que en este caso es una matriz de 3x3. Como se puede observar el propósito es imprimir los valores ya declarados en una matriz y se lleva a cabo con dos ciclos for, uno que lleve fila y columna i, j, respectivamente.

PROGRAMA 2

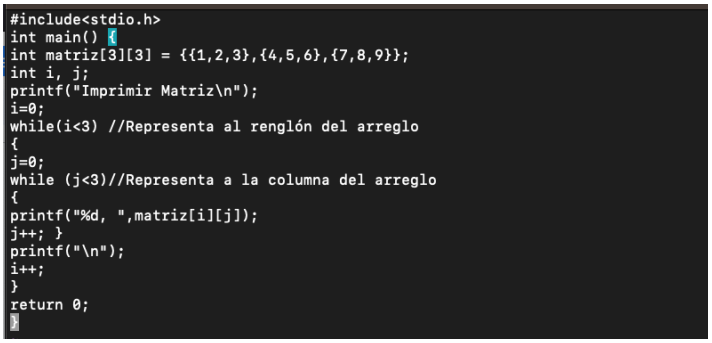
```
#include<stdio.h>
int main() {
int i,j,a[5][5];
for (i=0 ; i<5 ; i++)//Representa al renglón del arreglo
{
        for (j=0 ; j<5 ; j++)//Representa a la columna del arreglo
        {
a[i][j]=i+j;
printf("\t%d, ",a[i][j]);
}
printf("\n");
}
return 0;
}
```

```
Irlanda58:~ fp24alu35$ vi prog2.c
Irlanda58:~ fp24alu35$ gcc prog2.c -o prog2.out
Irlanda58:~ fp24alu35$ ./prog2.out
        0,      1,      2,      3,      4,
        1,      2,      3,      4,      5,
        2,      3,      4,      5,      6,
        3,      4,      5,      6,      7,
        4,      5,      6,      7,      8,
Irlanda58:~ fp24alu35$
```

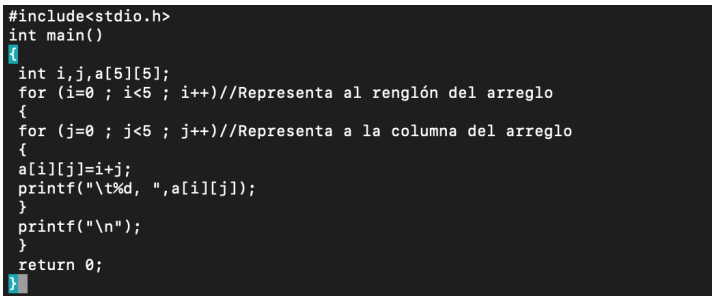
El propósito del programa es numerar tanto filas como columnas, la primera columna y fila van del 0 al 4.
El segundo renglón y segunda columna comienzan en 1 hasta 5, esto mismo pasa con el renglón y columna 3, comienzan en 2 y terminan en 6. Así sucesivamente teniendo en la última columna y renglón comienzo en 4 y concluye en 8.

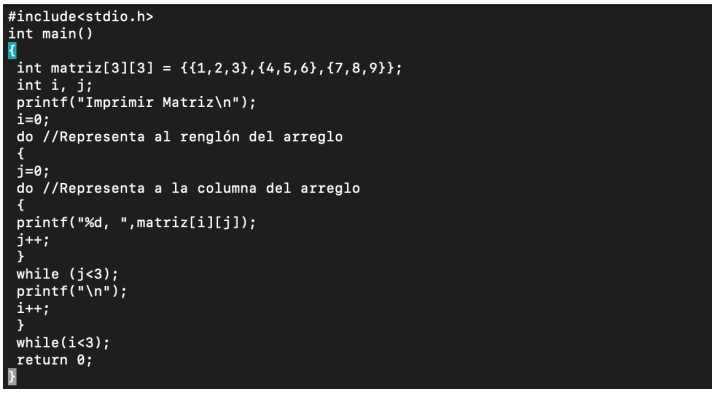
A continuación se presenta el mismo ejercicio (programa 2.c) pero con modificaciones utilizando instrucciones de repetición

WHILE	


`#include<stdio.h>`
`int main() {`
`int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};`
`int i, j;`
`printf("Imprimir Matriz\n");`
`i=0;`
`while(i<3) //Representa al renglón del arreglo`
`{`
`j=0;`
`while (j<3)//Representa a la columna del arreglo`
`{`
`printf("%d, ",matriz[i][j]);`
`j++; }`
`printf("\n");`
`i++;`
`}`
`return 0;`
`}`
	Se tienen el mismo arreglo bidimensional, pero aplicando ciclo while tanto para el arreglo i y j.
	Las condiciones es que **i**, **j** inicien desde 0 y vayan hasta 3 con incrementos.

FOR	


`#include<stdio.h>`
`int main()`
`{`
`int i,j,a[5][5];`
`for (i=0 ; i<5 ; i++)//Representa al renglón del arreglo`
`{`
`for (j=0 ; j<5 ; j++)//Representa a la columna del arreglo`
`{`
`a[i][j]=i+j;`
`printf("\t%d, ",a[i][j]);`
`}`
`printf("\n");`
`}`
`return 0;`
`}`
	Aquí existen dos ciclos for los cuales servirán para las variables del arreglo (i,j)
	Que inicien en 0, hasta 5 con incremento
	Y se imprimen en pantalla como a [i] [j].

DO-WHILE	
	Como ya se conoce, el ciclo do-while primero ejecuta el bloque de instrucciones y después revisa la condición en este caso la condición es que tanto i como j sean menores a 3

```c
#include<stdio.h>
int main()
{
 int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
 int i, j;
 printf("Imprimir Matriz\n");
 i=0;
 do //Representa al renglón del arreglo
 {
 j=0;
 do //Representa a la columna del arreglo
 {
 printf("%d, ",matriz[i][j]);
 j++;
 }
 while (j<3);
 printf("\n");
 i++;
 }
 while(i<3);
 return 0;
}
```

PROGRAMA 3

```c
#include <stdio.h>
int main () {
    int lista[10][10]; // Se declara el arreglo multidimensional
    int i,j;
    int renglon,columna;
    printf("\nDa el número de renglones y columnas separados con coma\n");
    scanf("%d,%d",&renglon,&columna);
    if(((renglon>=1) && (renglon<=10))&&((columna>=1) && (columna<=10)))
         {
          // Acceso a cada elemento del arreglo multidimensional usando for
for (i= 0 ; i <= renglon-1 ; i++)
{
        for(j= 0 ; j <= columna-1 ; j++)
{
                printf("\nNúmero para el elemento %d,%d del arreglo", i,j );
                scanf("%d",&lista[i][j]);
        }
}
         printf("\nLos valores dados son: \n");
           // Acceso a cada elemento del arreglo multidimensional usando for
for (i= 0 ; i <= renglon-1 ; i++) {
        for(j= 0 ; j <= columna-1 ; j++) {
                 printf("%d  ", lista[i][j]);
               }
             printf("\n");
           }
}
else printf("Los valores dados no es válido"); printf("\n");
return 0;
}
```

En este programa se cuenta con un arreglo bidimensional con parámetros de hasta 10 elementos, por lo que puede llegar a ser una matriz de 10x10.
El usuario ingresa tanto el número de renglones y de columnas que desea su arreglo, el programa sabe que arreglo será porque le pide al usuario separarlos por comas.
También el programa permite que el usuario ingrese los valores en su arreglo para finalmente imprimir el orden y el número de elementos.

```
Da el número de renglones y columnas separados con coma
2,2

Número para el elemento 0,0 del arreglo9

Número para el elemento 0,1 del arreglo8

Número para el elemento 1,0 del arreglo7

Número para el elemento 1,1 del arreglo6

Los valores dados son:
9  8
7  6

Irlanda58:~ fp24alu35$
```

Siguiendo la descripción del código, se elige un arreglo de 2x2 e ingreso los valores de 9, 8, 7, 6 como se muestra en pantalla.

```
#include<stdio.h>
int main()
{
 int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
 int i, cont=0, *ap;
 ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0];
 printf("Imprimir Matriz\n");
 for (i=0 ; i<9 ; i++)
 {
 if (cont == 3) //Se imprimió un renglón y se hace un salto de línea
 {
 printf("\n");
 cont = 0; //Inicia conteo de elementos del siguiente renglón
 }
 printf("%d\t",*(ap+i));//Se imprime el siguiente elemento de la matriz
 cont++;
 }
 printf("\n");
 return 0;
}
```

```
[Aldair:~ aldair$ gcc apuntador.c -o apuntador.out
[Aldair:~ aldair$ ./apuntador.out
Imprimir Matriz
1       2       3
4       5       6
7       8       9
```

En una matriz de dimensión 10 con ayuda de apuntadores la matriz irá de 0 a 100 con saltos de 10.

EJERCICIOS DE TAREA

1.-Realiza un programa que muestre tu nombre y número de cuenta haciendo uso de 2 arreglos, emplear while y for

```c
#include <stdio.h>
char nombre []={'e','v','e','r'};
int cuenta []={320216576};
int i;
int main (){
    printf("\nNOMBRE: %s",nombre);
    printf("\nCUENTA: %d",cuenta[i]);
    printf("\n");
return 0;
}
```

```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN

Aldair:menis aldair$ gcc tarea1.c -o tarea1.out
Aldair:menis aldair$ ./tarea1.out

NOMBRE: ever
CUENTA: 320216576
```

3.-Corrige e indica que realiza el siguiente programa:

```c
#include<stdio.h>
int main()
{
        int i,j, cont=0,n;
        float  M[2][2], s=0, *ap;
         ap = *M; //el apuntador tiene que llevar el * que es lo que va a buscar
         for(i=0; i<=3; i++)
         {
                for(j=0; j<=1;j++) //estaba en decremento.
                {
                        printf("\t\n Teclear el elemento %d %d \t", i+1,j+1);
                        scanf("%f",&M[i][j]);
                        s+=M[i][j];
                }
         }
        printf("\tLa Matriz es:\n");
        for (n=0 ; n<4 ; n++)
        {
                if (cont == 2)
                {
                        printf("\n");
                        cont = 1;
                }
                printf("%.2f\t\t",*(ap+n));
                cont++;
         }
         printf("\t\n\n La  suma de los elementos es:%0.2f", s);
return 0;
}
```

```
  Teclear el elemento 1 1        2

  Teclear el elemento 1 2        3

  Teclear el elemento 2 1        2

  Teclear el elemento 2 2        3

  Teclear el elemento 3 1        4

  Teclear el elemento 3 2        3

  Teclear el elemento 4 1        2

  Teclear el elemento 4 2        3
         La Matriz es:
 2.00            3.00
 2.00
 3.00
```

4.- Programa que solicite al usuario los valores de dos matrices de 3 x 3 y haga su multiplicación haciendo uso de arreglos. Conforme se muestrea a continuación:

C tarea1.c > main()

```c
#include <stdio.h>
//para la primera matriz
int a,b;
int M[3][3];
int s=0;
//para la segunda matriz
int f,g;
int r=0;
int N[3][3];

int main (){
//Pidiendo los datos de la primera matriz
for (a=0;a<3;a++){
    for (b=0;b<3;b++){
        printf("\nTeclear los elementos de la primera matriz: [%d] [%d]: \t",a+1,+b+1);
        scanf("%d",&M[a][b]);
        s+=M[a][b];
    }
}
//Pidiendo los datos para la segunda matriz
for (f=0;f<3;f++){
    for (g=0;g<3;g++){
        printf("\nTeclear los elementos de la primera matriz [%d] [%d]: \t",f+1,g+1);
        scanf("%d",&N[f][g]);
        r+=N[f][g];
    }
}
//Imprimiendo los datos de la primera matriz
printf("\n\tLa primera matriz es: \n");
for (a=0;a<3;a++){
    for (b=0;b<3;b++){
        printf("\t%d",M[a][b]);
    }
    printf("\t\n");
}
    printf ("\n");

//Imprimiendo los datos de la segunda matriz
printf("\n\tLa matriz dos es: \n");
for (f=0;f<3;f++){
```

```c
//Imprimiendo los datos de la segunda matriz
printf("\n\tLa matriz dos es: \n");
for (f=0;f<3;f++){
    for (g=0;g<3;g++){
        printf("\t%d",N[f][g]);
    }
    printf("\t\n");
}
    printf ("\n");
    return 0;
}
```

```
Aldair:menis aldair$ gcc tarea1.c -o tarea1.out
Aldair:menis aldair$ ./tarea1.out

 Teclear los elementos de la primera matriz: [1] [1]:    2

 Teclear los elementos de la primera matriz: [1] [2]:    2

 Teclear los elementos de la primera matriz: [1] [3]:    2

 Teclear los elementos de la primera matriz: [2] [1]:    2

 Teclear los elementos de la primera matriz: [2] [2]:    2

 Teclear los elementos de la primera matriz: [2] [3]:    2

 Teclear los elementos de la primera matriz: [3] [1]:    2

 Teclear los elementos de la primera matriz: [3] [2]:    2

 Teclear los elementos de la primera matriz: [3] [3]:    2

 Teclear los elementos de la primera matriz [1] [1]:     4

 Teclear los elementos de la primera matriz [1] [2]:     4

 Teclear los elementos de la primera matriz [1] [3]:     4

 Teclear los elementos de la primera matriz [2] [1]:     4

 Teclear los elementos de la primera matriz [2] [2]:     4

 Teclear los elementos de la primera matriz [2] [3]:     4

 Teclear los elementos de la primera matriz [3] [1]:     4

 Teclear los elementos de la primera matriz [3] [2]:     4

 Teclear los elementos de la primera matriz [3] [3]:     4

         La primera matriz es:
         2       2       2
         2       2       2
         2       2       2


         La matriz dos es:
         4       4       4
         4       4       4
         4       4       4
```

CONCLUSIÓN

Es muy importante conocer el uso de las instrucciones repetitivas, pues de un código complicado que requiere este tipo de instrucciones se puede simplificar usando la adecuada. Sinceramente yo prefiero utilizar la instrucción for ya que en primer lugar va todo una línea de código y considero que es más compacta y más fácil de utilizar.

REFERENCIAS

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