Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Alejando Pimentel
Asignatura:	Fundamentos de programación
Grupo:	3
No de Práctica(s):	11
Integrante(s):	Uno Karin Natalia
No. de Equipo de cómputo empleado:	26
No. de Lista o Brigada:	8723 #50
Semestre:	1
Fecha de entrega:	28 de octubre del 2019
Observaciones:	

CALIFICACIÓN: _____

PRÁCTICA 11:
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.

INTRODUCCIÓN

Las matrices o como algunos las llaman "arreglos multidimensionales" son una estructura de datos bastante similar a los vectores o arreglos. Una matriz no es más que una serie de vectores contenidos uno en el otro (u otros), es decir, una matriz es un vector cuyas posiciones son otros vectores. Hablemos con más detalle de esto para quedar más claros.

En términos generales, una matriz es una estructura conformada por filas y columnas, idealmente más de dos filas y columnas, de hecho, podemos decir que si una "matriz" tiene una única fila o una única columna, entonces estamos hablando de un vector y no una matriz como tal.

La intersección de una fila y una columna de la matriz son las casillas y cada una de ellas podrá poseer información, simple o compleja (ya dependerá de nuestras necesidades).

Ahora, tal como dije antes, un vector posee una única fila (o columna, como lo quieras ver) y de este modo un grupo de vectores unidos conforman una matriz, es por esto que al comienzo dije que una matriz es un vector conformado por otra serie de vectores.

Declarar una matriz en C++ es muy similar a la de un vector, se deben seguir las mismas normas para declarar una variable pero una vez más con un pequeño cambio en la sintaxis. Primero necesitaremos saber el tipo de los datos que irán al interior de este (números, decimales o cadenas de texto, etc.) necesitamos también, como siempre, un nombre para la matriz y un tamaño máximo tanto para las filas como para las columnas. La sintaxis para declarar una matriz en C++ es la siguiente:

```
tipoDato nombreMatriz[filas][columnas];
```

o con un valor especifico:

```
int myMatriz1[2][2] = {{1,2},{3,4}};
```

Ejemplo:

```c
#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;
}
```

DESARROLLO/RESULTADOS

ACTIVIDAD1.

```c
#include <stdio.h>

int main()
{
    int n,i;
    printf("Ingrese la cantidad en la lista:\n");
    scanf("%i",&n);

    int lista[n];
    for(int i=0; i<n; i++){
        printf("Ingrese n numero:");
        scanf("%i",&lista[i]);
    }
    int valor1=lista[0];
    int valor2=lista[0];
    for(int i=0; i<n; i++){
        if(lista[i]<valor1){
            valor1=lista[i];
        }
        if(lista[i]<valor2){
            valor2=lista[i];
        }
    }
    printf("El mayor es: %i\n", valor1);
    printf("El menor es: %i\n", valor2);

    return 0;
}
```

```
[Karins-MacBook-Pro:~ karinnataliaunc
Ingresa N: 1
Ingresa M: 2
Matriz 1:
1
```

ACTIVIDAD 2.

```c
#include <stdio.h>

int main()
{
    int N,M;
    printf("Ingresa N: "); scanf("%i", &N);
    printf("Ingresa M: "); scanf("%i", &M);

    int matriz[N][M];
    printf("Matriz 1:\n");
    for(int i=0; i<N; i++){
        for (int j=0; j<M; j++){
            scanf("%i", &matriz[i][j]);

        }
    }
int matriz2[N][M];
    printf("Matriz 2:\n");
    for(int i=0; i<N; i++){
        for (int j=0; j<M; j++){
            scanf("%i", &matriz[i][j]);
        }
}
int matrizR[N][M];
    printf("Matriz R:\n");
    for(int i=0; i<N; i++){
        for (int j=0; j<M; j++){
            matrizR[i][j]=matriz[i][j]+matriz2[i][j];
        }
}
    for(int i=0; i<N; i++){
        for (int j=0; j<M; j++){
            printf("%i\n", matrizR[i][j]);
    }
            printf("\n");
        }
    return 0;
}
```

```
[Karins-MacBook-Pro:~ karinnataliaunc
Ingresa N: 1
Ingresa M: 2
Matriz 1:
1
```

CONCLUSIÓN

Podemos concluir que las matrices son útiles para ejecutar listas utilizando argumentos que van incrementando, como una función utilizando una variable principal así conforme el programa avanza se le van asignando a cada lugar un valor.