/\*\*Este Proyecto Explora los Vectores de 2 Dimensiones - Las Matrices\*\*/

#include <stdio.h>

#include <stdlib.h>

int main()

{

    initializeMatrix();

    multiplicationTable();

    /\*inputMatrix();\*/

    memoryAddress();

    return 0;

}

/\*\*Este Procedimiento Inicializa y Declara una Matriz de Distintas Maneras\*\*/

void initializeMatrix()

{

    int f, k;

    //Standard Way

    int iMatrix[3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};

    printf("Impresion de Matriz (3 x 4) que fue Inicializada de Manera Estandar.\n");

    for(f = 0; f < 3; f++)

    {

        for (k = 0; k < 4; k++)

            printf("\t%i", iMatrix[f][k]);

        printf("\n");

    }

    printf("\n \n");

    //Incomplete Values in the Internal Curly Brackets (2 Examples)

    int iMatrix2[4][3] = {{1,2}, {4,5,8}, {10}};

    printf("Impresion de Matriz (4 x 3) que Fue Inicializada con Valores en Blanco.\n");

    for(f = 0; f < 4; f++)

    {

        for(k = 0; k < 3; k++)

            printf("\t%i", iMatrix2[f][k]);

        printf("\n");

    }

    printf("\n \n");

    int iMatrix3[2][3] = { {5,6}, {7} };

    printf("Impresion de Matriz (2 x 3) que Fue Inicializada con Valores en Blanco.\n");

    for (f = 0; f < 2; f++)

    {

        for (k = 0; k < 3; k++)

            printf("\t%i", iMatrix3[f][k]);

        printf("\n");

    }

    printf("\n \n");

    //Excessive Values in the Internal Curly Brackets

    int iMatrix4[5][6] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30};

    printf("Impresion de Matriz (5 x 6) que Fue Inicializada con Una Secuncia Consecutiva de Valores.\n");

    for (f = 0; f < 5; f++)

    {

        for (k = 0; k < 6; k++)

            printf("\t%i", iMatrix4[f][k]);

        printf("\n");

    }

    printf("\n \n");

}

/\*\*Este Procedimiento Crea una Tabla de Multiplicar\*\*/

#include <stdio.h>

#define iSIZE 10

void multiplicationTable()

{

    int iTable[iSIZE][iSIZE], f, k;

    for (f = 0; f < iSIZE; f++)

    {

        for (k = 0; k < iSIZE; k++)

            iTable[f][k] = (f + 1) \* (k + 1);

    }

    printf("Impresion de las Tablas de Multiplicar.\n");

    for (f = 0; f < iSIZE; f++)

    {

        for (k = 0; k < iSIZE; k++)

            printf("\t%i", iTable[f][k]);

        printf("\n");

    }

    printf("\n \n");

}

/\*\*Este Procedimiento Recibe del Usuario la Informaci�n Necesaria\*\*/

#define iROW 3

#define iCOL 4

void inputMatrix()

{

    int iMatrix[iROW][iCOL], f, k;

    for (f = 0; f < iROW; f++)

    {

        for (k = 0; k < iCOL; k++)

        {

            printf("Ingresa el numero entero de la celda %c:%i --> ", k + 65, f + 1);

            scanf("%i", &iMatrix[f][k]);

        }

    }

    printf("Impresion de la Matriz que Has Ingresado.\n");

    for (f = -1; f < iROW; f++)

    {

        if (f != -1)

            printf("%i", f + 1);

        for (k = -1; k < iCOL; k++)

        {

            if (f == -1 && k != -1)

                printf("\t %c", k + 65);

            else if (k != -1)

                printf("\t %i", iMatrix[f][k]);

        }

        printf("\n");

    }

    printf("\n \n");

}

/\*\*Este Procedimiento se Encarga de Desplegar la Direccion en Memoria de los Elementos de Vectores (1D y 2D)\*\*/

void memoryAddress()

{

    int f, k;

    //Impresion de un Vector y sus Direcciones de Memoria

    int iArray[5] = {1,2,3};

    printf("Impresion de un Vector y sus Direcciones en Memoria. \n");

    for (f = 0; f < 5; f++)

    {

        printf("Elemento: %i\nDireccion:%lu\n\n", iArray[f], &iArray[f]);

    }

    printf("\n \n");

    //Impresion de una Matriz y sus Direcciones de Memorio

    int iMatrix[2][5] = { {1,2,3,4,5}, {25,26} };

    printf("Impresion de una Matriz y sus Direcciones en Memoria.\n");

    for (f = 0; f < 2; f++)

    {

        for (k = 0; k < 5; k++)

        {

            printf("Elemento: %i\nDireccion: %lu\n\n", iMatrix[f][k], &iMatrix[f][k]);

        }

    }

    printf("\n \n");

}