|  |  |  |
| --- | --- | --- |
| Imagen que contiene dibujo, taza  Descripción generada automáticamente | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

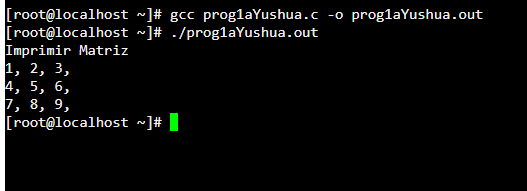
|  |  |
| --- | --- |
| *Profesor:* | Ingeniería Karina García Morales |
| *Asignatura:* | Fundamentos de Programación |
| *Grupo:* | 20 |
| *No de Práctica:* | *10 ARREGLOS MULTIDIMENSIONALESº* |
| *Integrante:* | Yushua Haza Jibaja |
| *No. de Equipo de cómputo empleado:* |  |
| *No. de Lista o Brigada:* | 22 |
| *Semestre:* | 2022 - 1 |
| *Fecha de entrega:* | 28 de septiembre de 2021 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

Prog 1

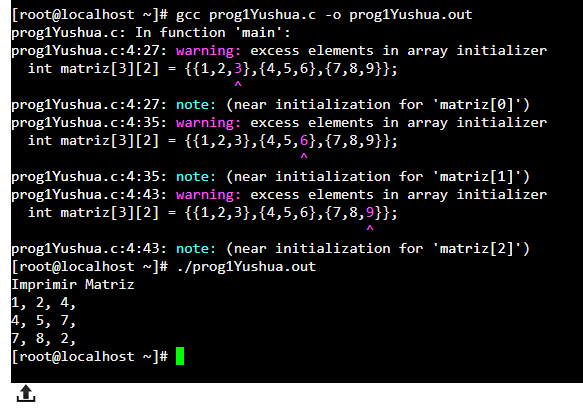
Ejecución del primer programa de la práctica 1a.c

Ejecución sin modificar (primera iteración sin modificación)



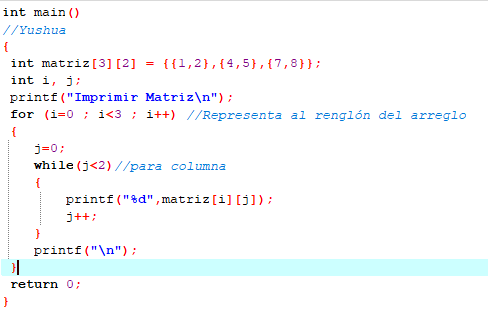
1.-Realiza una matriz inicializada de enteros de 3\*2

Segunda iteración con cambio de matriz 3,2

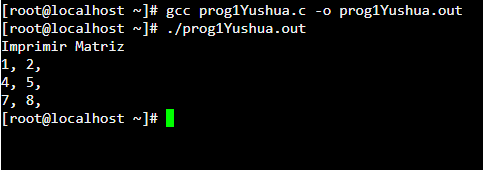


Tercera iteración

Para esta iteración se solicito que añadiese un ciclo while, el cual se ve reflejado en la siguiente prueba de escritura de código:

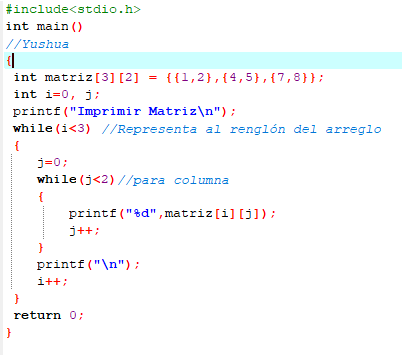


Prueba de consola de linux

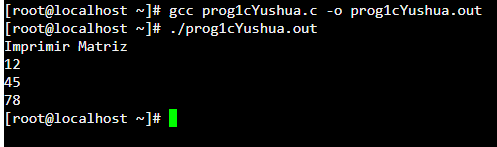


Cuarta iteración

Para esta iteración se solicitó modificar para agregar 2 ciclos while

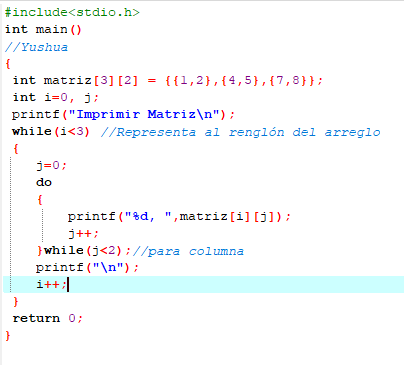


Prueba de consola de linux

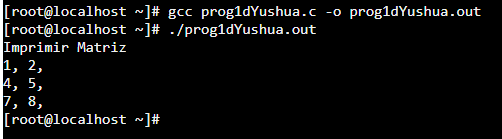


5ta iteración

Ahora se solicitó modificar el programa para que contemple un ciclo do while

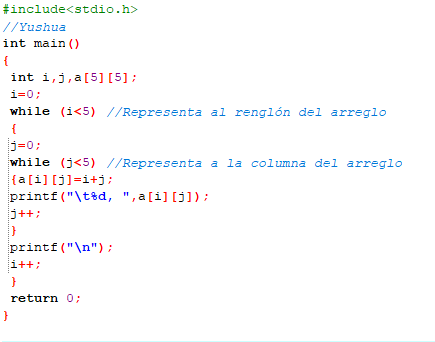


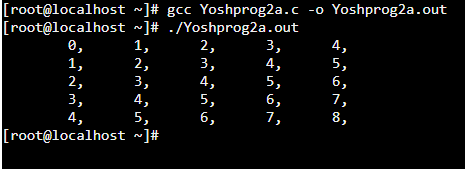
Prueba de consola de linux



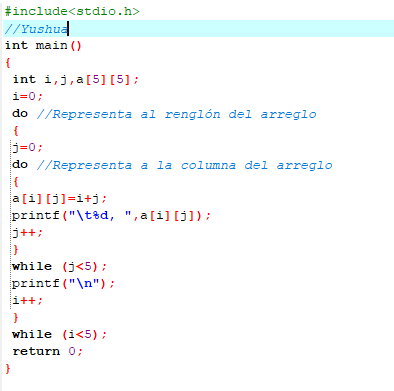
Programa 2

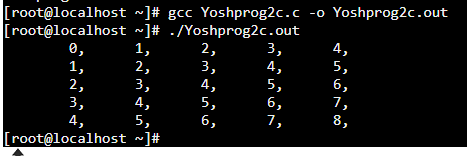
Ahora se solicita probar la ejecución del programa 2b.c para seguir visualizando el funcionamiento de las matrices, en concreto la mayor diferencia con este programa y los anteriores es que hace uso de 2 ciclos while y la matriz es de 5x5





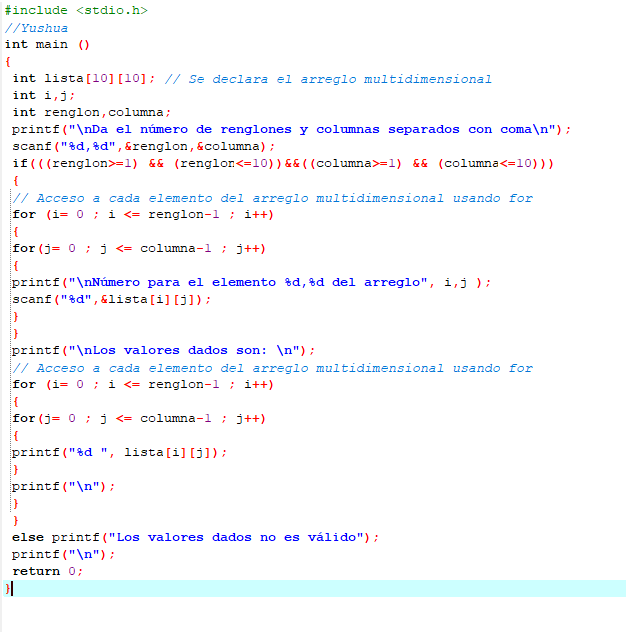
Similar a los programas previos, las diferencias entre uno y otro programa es el tamaño del arreglo y las estructuras empleadas, para los siguientes programas hasta el 3.c eran opcionales de ejecutar puesto que el objetivo era denotar estas diferencias ya mencionadas.

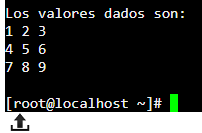
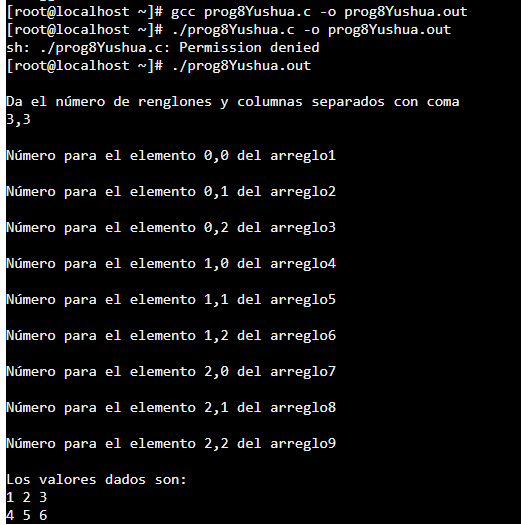


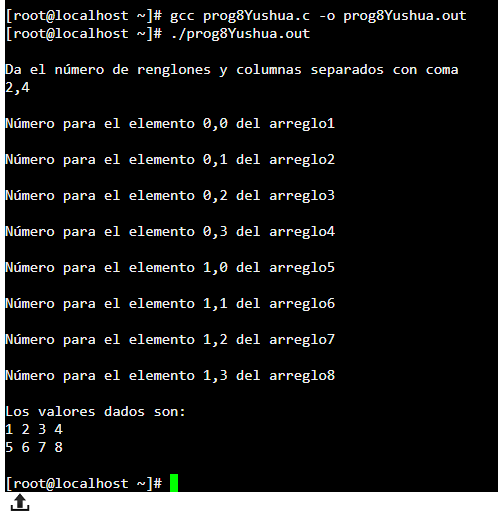


Programa 3

Para este ejercicio se requirió ejecutar sin cambios al código 3.c de la práctica, este código ahora permite definir la cantidad de renglones y columnas del arreglo y cuales van a ser sus valores.

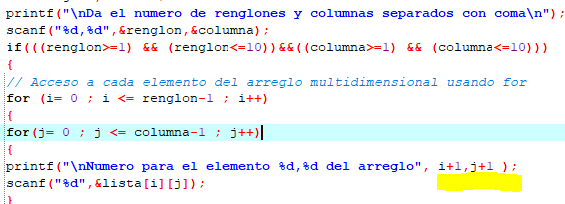


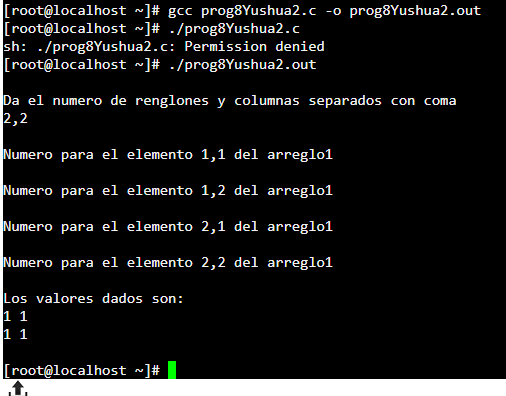




Segunda iteración

En esta segunda iteración se solicitó hacer un cambio para que la visualización del cósigo se viera correctamente en las impresiones de pantalla, concretamente respecto a la línea de texto “Numero para el elemento %d,%d del arreglo”, en la cual inicialmente aparecía 0,0 como elemento inicial pero ahora se cambia a elemento (1,1), además de que como modificación personal quité las “u” acentuadas del programa porque el símbolo carácter se interpreta erróneamente en C++ pero correctamente en Linux.





Tercera iteración

Se solicito cambiar el código para usar estructuras while y do while

Apuntadores y su relación con arreglos de dos dimensiones

Un apuntador es un tipo de variable que almacena la dirección o valor de una variable distinta.

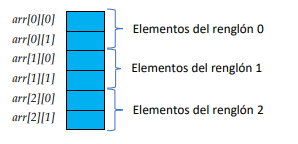
La sintaxis para declarar un apuntarod y para asiganrle la dirección de memoria de otra varia es, respectivamente:

TipoDeDato\*apuntador,variable;

Apuntador=&variable;

El símboloampersand(&) es referirse a la dirección de memoria del valor del la variable

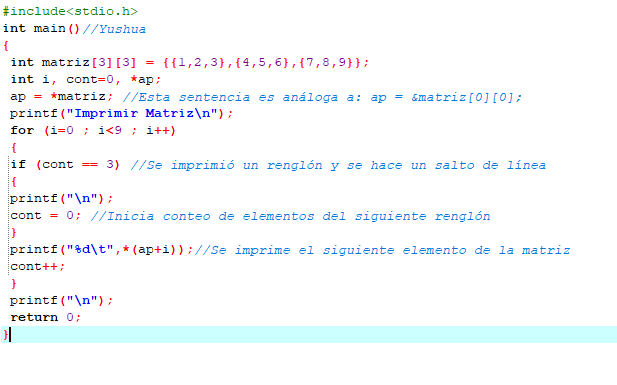
Los apuntadores solo pueden apuntar a direcciones de memoria del mismo tipo de dato con el que fueron declarados; para acceder al contenido de dicha dirección, a la variable apuntador se le agrega previamente un “\*”.



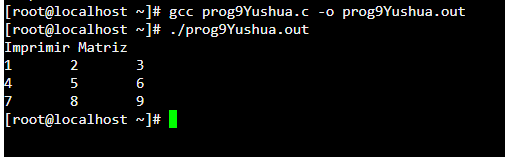
Programa 4

Se prueba la ejecución del programa 4ª.c

Este programa ahora incluye un apuntador

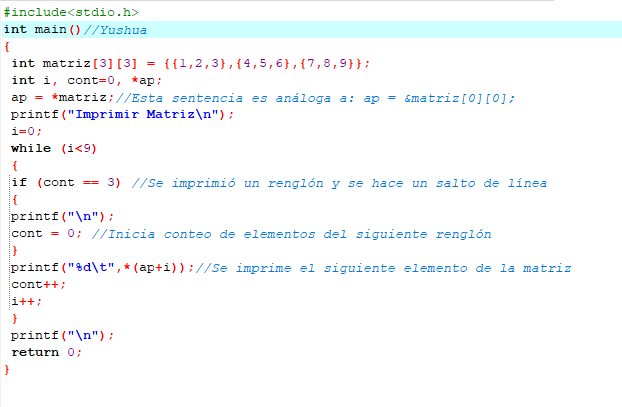


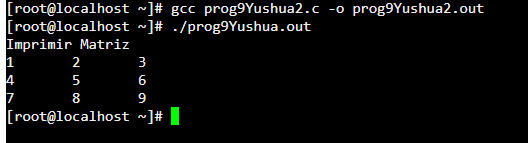
Prueba de ejecución



Segunda iteración

Continuando con la dinámica de cambio de estructuras, al programa anterior se le cambio un ciclo for por uno while





**Conclusión**

Se pudo visualizar y aprender mejor el funcionamiento de las estructuras for, while y do-while que finalmente funcionan de manera similar y se puede utilizar para llegar a programas que ejecutan los mismos resultados, pero a la vez de distinta manera, la práctica fue finalmente muy clara y concisa y además pudo introducir el tema de matrices

Tarea

1.- Indica que realiza el siguiente programa y completa los letreros del programa:

#include <stdio.h>

#define p printf

#define s scanf

int main()

{

char nombres[6][10] = {" jose ","teresa","mariana","mario","pedro","juan"};

char alum[10];

int cont=0;

p("Ingrese el nombre del alumno: \n");

s("%s", alum);

int i;

for(i=0; i<6; i++){

if (strcmp(nombres[i],alum)==0){ // compara el número mayor de nombres[i] con alum

p("El alumno %s aquí indicar que muestra el programa\n", nombres[i]);

}else{

cont++;

}

}

if (cont==6){

p("%s aquí indicar que muestra el programa ",alum);

}

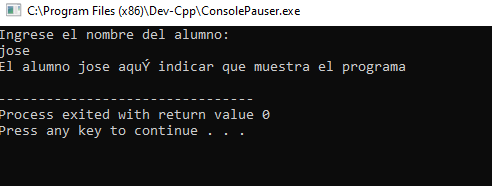
return 0;

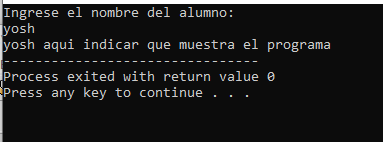
}

Inicialmente este programa no se ejecuta correctamente puesto que C++ no identifica el comando o texto “strcmp”

Sin embargo, con una corrección de añadir una librería se puede ejecutar, cuyo resultado es el siguiente:







Este programa debería poder leer e identificar el nombre de los alumnos que están dentro de un arreglo, y a partir de ello diferencia que alumnos o texto ingresado no coincide con los nombres del arreglo, a lo cual arroja 2 tipos de resultados siendo la diferencia que cuando el nombre coincide con la lista, se imprime”alumno…” y cuando no, se imprime “(nombre o texto)…”

2.-Leer un arreglo de 3x3 e imprimir la matriz y su matriz transpuesta.

Para elaborar este programa me fue necesario usar C++ y un código previo de una práctica previamente elaborada, el código previo permite leer un arreglo matriz de 3 elementos, lo que me restaba por elaborar era la impresión de pantalla de la matriz traspuesta, la cual cambia los valores de los renglones a columnas

Codificación

#include<stdio.h>

//Yushua

int main()

{

int i,j;

float M[3][3], s=0;

for (i=0; i<3; i++)

{

for(j=0; j<3; j++)

{

printf("\n Teclear el elemento %d %d\t", i+1,j+1);

scanf("%f",&M[i][j]);

s+=M[i][j];

}

}

printf("\n\tLa matriz es:\n");

for(i=0; i<3; i++)

{

for(j=0; j<3;j++)

{

printf("\t%.1f",M[i][j]);

}

printf("\t\n");

}

printf("La Matriz Transpuesta\n");

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

printf("\t%.1f",M[j][i]);

}

printf("\n");

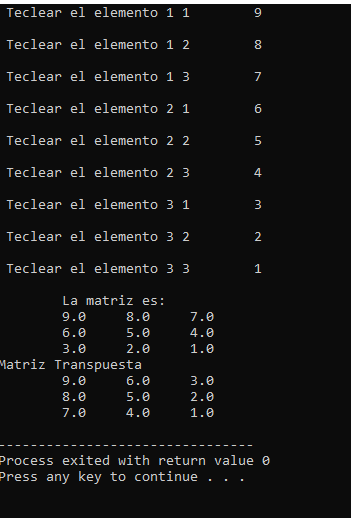
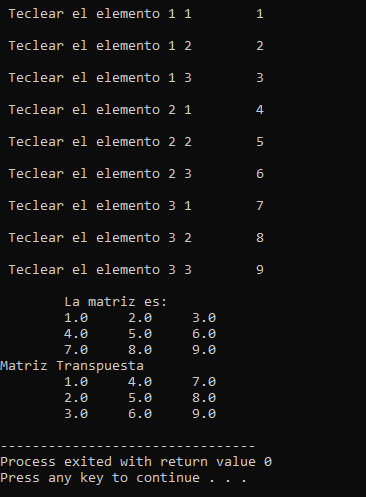
}

return 0;

}



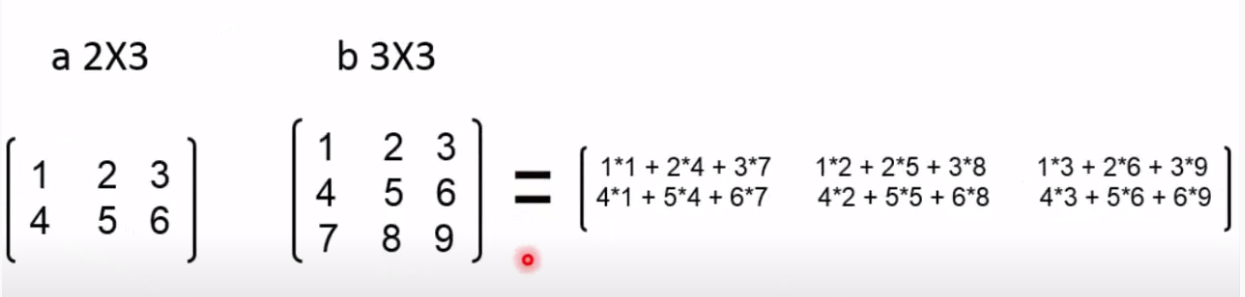
Pruebas de ejecución



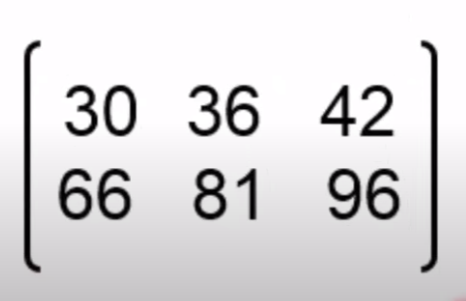
3.- Programa que solicite al usuario los valores de dos matrices de 3 x 3 y haga su multiplicación haciendo uso de arreglos. Conforme se muestrea a continuación:

Para este programa me fue necesario usar C++ y el código previamente usado para leer los valores del arreglo matriz que nos da el usuario, pues el código es útil y cómodo y funciona correctamente como un bloque aparte de la multiplicación de matrices

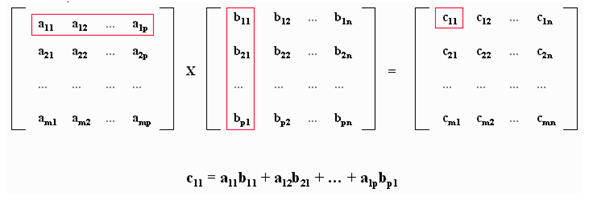
Para poder llevar a cabo esta tarea también se tuvo que entender como se lleva a cabo una multiplicación de matrices, la cual se explica con la siguiente imagen



La mutilplicación de matrices solo se puede llevar a cabo si es que existen es la misma cantidad de columnas en la matriz “a” y renglones en la matriz “b”, así entonces obtenemos:



Siendo esta la matriz resultante de la multiplicación



Código

#include<stdio.h>

#include<stdlib.h>

int main(){

int a[3][3];

int b[3][3];

int c[3][3];

int i,j;

printf("Primera matriz\n");

for(i=1;i<=3;i++)

for(j=1;j<=3;j++)

{

printf("\n Introduce los valores de [%d][%d] : ",i,j);

scanf("%d",&a[i][j]);

}

printf("\nSegunda matriz\n");

for(i=1;i<=3;i++)

for(j=1;j<=3;j++)

{

printf("\n Introduce los valores de [%d][%d] : ",i,j);

scanf("%d",&b[i][j]);

}

printf("\nLa matriz resultante es");

for(i=1;i<=3;i++)

for(j=1;j<=3;j++)

c[i][j]=(a[i][1]\*b[1][j])+(a[i][2]\*b[2][j])+(a[i][3]\*b[3][j]);

for(i=1;i<=3;i++)

{

for(j=1;j<=3;j++)

printf("%d ",c[i][j]);

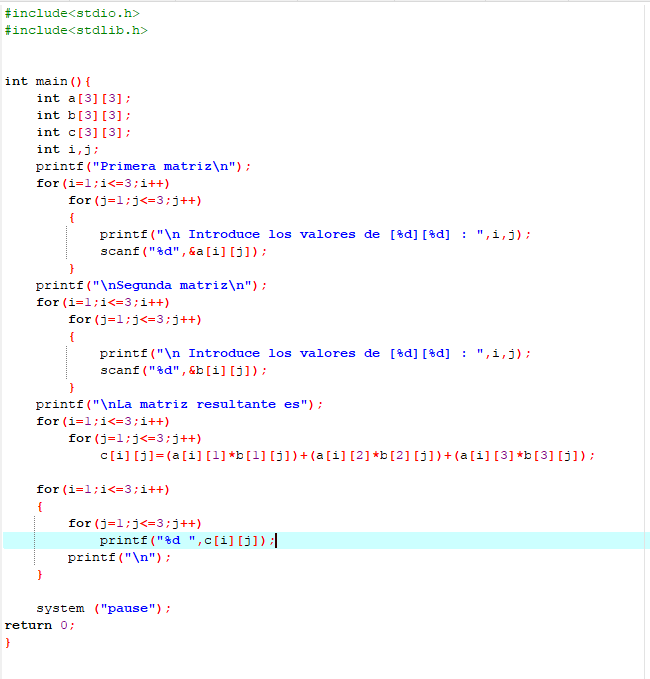
printf("\n");

}

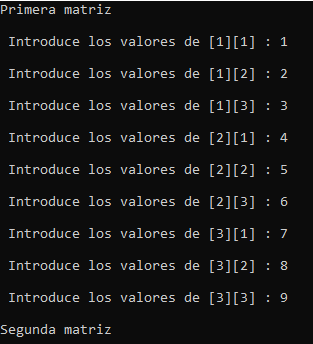
system ("pause");

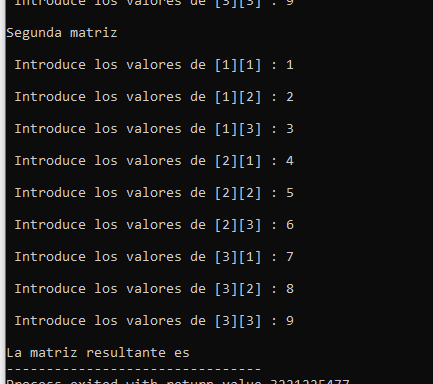
return 0;

}



Prueba de ejecución





Link de github: https://github.com/YushuaHaza/Practica-10

**Referencia**

-Facultad de Ingenieria. (2021). Laboratorio de computacion Sala A y B. Recuperado el 10 de noviembre de 2021 de http://lcp02.fi-b.unam.mx/