



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación  
salas A y B

*Profesor:* Alejandro Esteban Pimentel Alarcon

*Asignatura:* Fundamentos de programación.

*Grupo:* 3

*No de Práctica(s):* 11

*Integrante(s):* Crail Avila Regina  
Ortiz Garcia Cesar Alan

*No. de Equipo de cómputo  
empleado:* 30,31

*No. de Lista o Brigada:* 8973  
9070

*Semestre:* 20-21

*Fecha de entrega:* 27/oct/19

*Observaciones:* Las actividades están incompletas, en la primer actividad falta la parte de obtener el mínimo y el máximo. Y en la segunda actividad se tenían que pedir dos matrices y mostrar el resultado de sumarlas.

CALIFICACIÓN: 6

## Arreglos c++

1. DEFINICIÓN: Un arreglo en C++ es un conjunto de datos que se almacenan en memoria de manera contigua con el mismo nombre. Para diferenciar los elementos de un arreglo se utilizan índices detrás del nombre del arreglo y encerrados por []. EJEMPLO: `int vector [10];` // array de 10 enteros: `vector[0]..vector[9]`.

2. DIMENSIONES DE LOS ARREGLOS: Arreglos de una dimensión: Un arreglo de una dimensión es una lista de variables, todas de un mismo tipo a las que se hace referencia por medio de un nombre común. Una variable individual del arreglo se llama elemento del arreglo. Para declarar un arreglo de una sola dimensión se usa el formato general: Representación gráfica de un arreglo de una dimensión: `float arreglo[6];`

3. Representación gráfica de un arreglo de dos dimensiones: `int arreglo[4][4]`

4. Representación gráfica de un arreglo de tres dimensiones: `int arreglo[4][4][3];`

5. ÍNDICE DE UN ARREGLO: Todo arreglo está compuesto por un número de elementos. El índice es un número correlativo que indica la posición de un elemento del arreglo. Los índices en C++ van desde la posición 0 hasta la posición tamaño - 1.

6. EJEMPLO DE ÍNDICE DE UN ARREGLO: Como se puede ver en el gráfico es un arreglo unidimensional de tamaño 10, pero el índice va del 0 al 9. Es por esto que al indicar su posición se debe inicializar en 0.

7. ELEMENTO DE UN ARREGLO: Un elemento de un arreglo es un valor particular dentro de la estructura del arreglo. Para acceder a un elemento del arreglo es necesario indicar la posición o índice dentro del arreglo. Ejemplo: `* arreglo[0]` // Primer elemento del arreglo `* arreglo[3]` // Cuarto elemento del arreglo

8. INICIALIZACIÓN DE ARRAYS: Los arrays pueden ser inicializados en la declaración. Ejemplos: `float R[10] = {2, 32, 4.6, 2, 1, 0.5, 3, 8, 0, 12};` `float S[] = {2, 32, 4.6, 2, 1, 0.5, 3, 8, 0, 12};` `int N[] = {1, 2, 3, 6};` `int M[][3] = { 213, 32, 32, 32, 43, 32, 3, 43, 21};` `char Mensaje[] = "Error de lectura";` `char Saludo[] = {H, o, l, a, 0};`

9. EJEMPLO : El siguiente programa carga el arreglo del número 1 al 9 y luego los muestra: `#include <iostream.h> void main(){int numero[10]; int i; for (i=1; i<11; i++){numero[i-1]=i;} for (i=0; i<10; i++){cout<<numero[i]<<endl;} getch;}`

10. DECLARACIÓN DE ARREGLOS UNIDIMENSIONALES: Para declarar un arreglo de una sola dimensión se usa el formato general: `tipo_dato identificador[tamaño];` Declaración: `int arreglo[3];` // forma un arreglo de una dimensión y de tres elementos `Nombre del arreglo` `arreglo` `Nombre de los elementos` `arreglo[0] → primer elemento` `arreglo[1] → segundo elemento` `arreglo[2] → tercer elemento`

11. DECLARACIÓN DE ARREGLOS MULTIDIMENSIONALES: La sintaxis es la siguiente: `tipo_dato identificador [dimensión1] [dimensión2] ...[dimensiónN]` ; Donde N es un número natural positivo. EJEMPLO: Declaración: `int m[2][3];` // forma una tabla de dos filas y tres columnas. // cada fila es un arreglo de una dimensión.

**12. OPERACIONES CON ARREGLOS UNIDIMENSIONALES:** Suma y Resta: Los arreglos deben tener el mismo tamaño y la suma se realiza elemento a elemento. Por ejemplo  $C = A + B$ . Donde A, B y C son arreglos de enteros de tamaño 3.

**13. OPERACIONES CON ARREGLOS MULTIDIMENSIONALES: SUMA Y RESTA:** Los arreglos deben tener el mismo orden y la suma se realiza elemento a elemento. Por ejemplo sean A, B y C arreglos de números punto flotante de orden  $2 \times 3$ . Entonces la operación  $C = A + B$  sería:

**14. PRODUCTO POR UN ESCALAR:** Dada una matriz A y un escalar c, su producto  $cA$  se calcula multiplicando el escalar por cada elemento de A, así tenemos:  $cA = cA[i, j]$ . Ejemplo:

**15. PRODUCTO DE MATRICES:** El producto de dos matrices se puede definir sólo si el número de columnas de la matriz izquierda es el mismo que el número de filas de la matriz derecha. Si A es una matriz  $m \times n$  y B es una matriz  $n \times p$ , entonces su producto matricial AB es la matriz  $m \times p$  (m filas, p columnas) dada por:  $(AB)[i, j] = A[i, 1] B[1, j] + A[i, 2] B[2, j] + \dots + A[i, n] B[n, j]$  para cada par i y j. EJEMPLO:

**16.** Las operaciones con arreglos unidimensionales o arreglos multidimensionales son de mucha utilidad, ya que nos facilitan el cálculo de operaciones muy complejas. Esto no funciona así nada más

**17. ARREGLOS DE CARACTERES MULTIDIMENSIONALES:** Los arreglos de cadenas, que a menudo se conocen como tablas de cadenas son comunes en la programación en C++. Una tabla de cadenas de dos dimensiones es creada como otro cualquier arreglo de dos dimensiones. No obstante, la forma como se conceptualizará será levemente diferente. Por ejemplo: `char nombres [10][50];` A continuación se especifica como quedará declarada el presente arreglo.

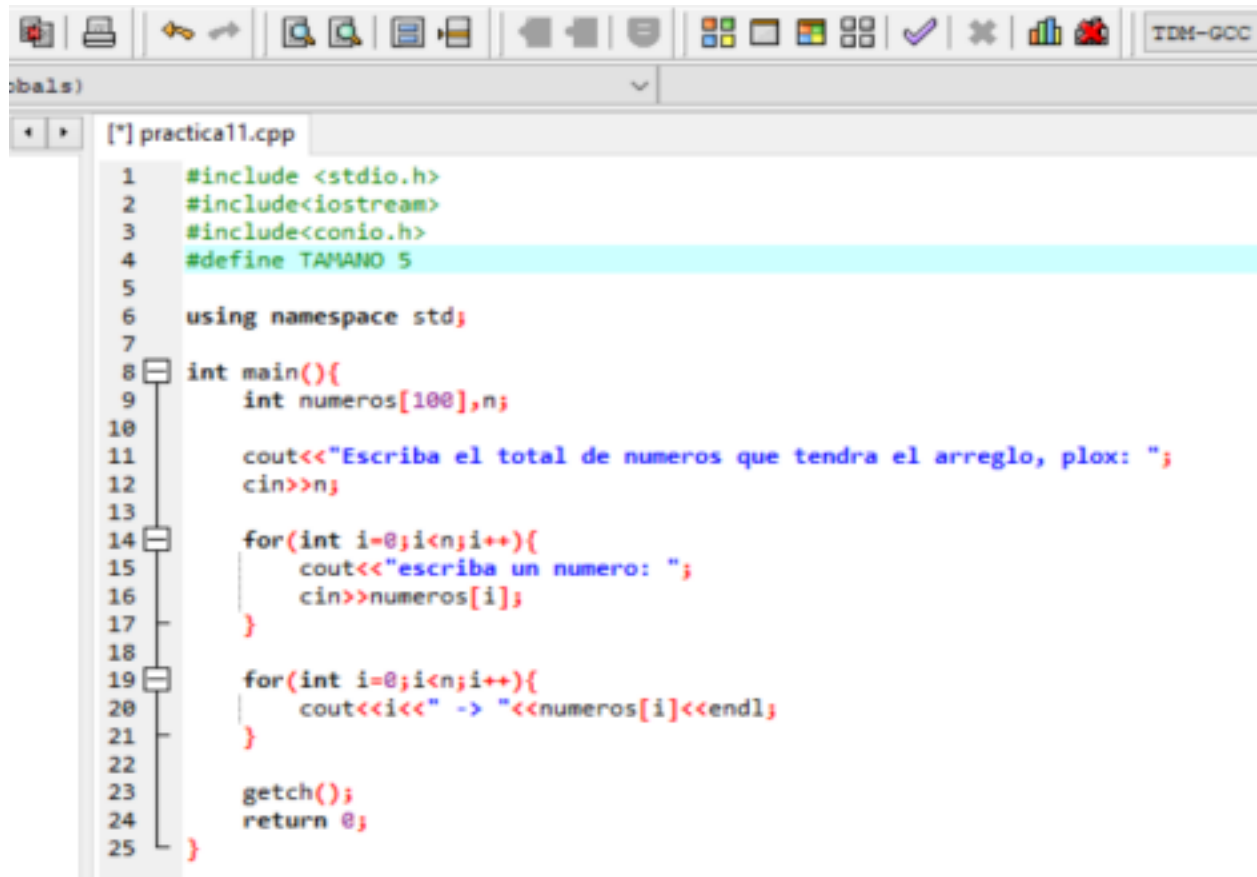
**18.** La sentencia anterior especifica una tabla que puede contener hasta 10 cadenas, cada una de hasta 50 caracteres de longitud. Para acceder a una cadena dentro de la tabla se especifica solamente el primer índice. Por ejemplo para introducir una cadena desde el teclado en la tercera cadena de nombres, se utilizaría la siguiente sentencia: `gets(nombres[2]);` De la misma manera, para dar salida a la primera cadena se utilizaría la sentencia `cout << nombres[0];`

1.\_

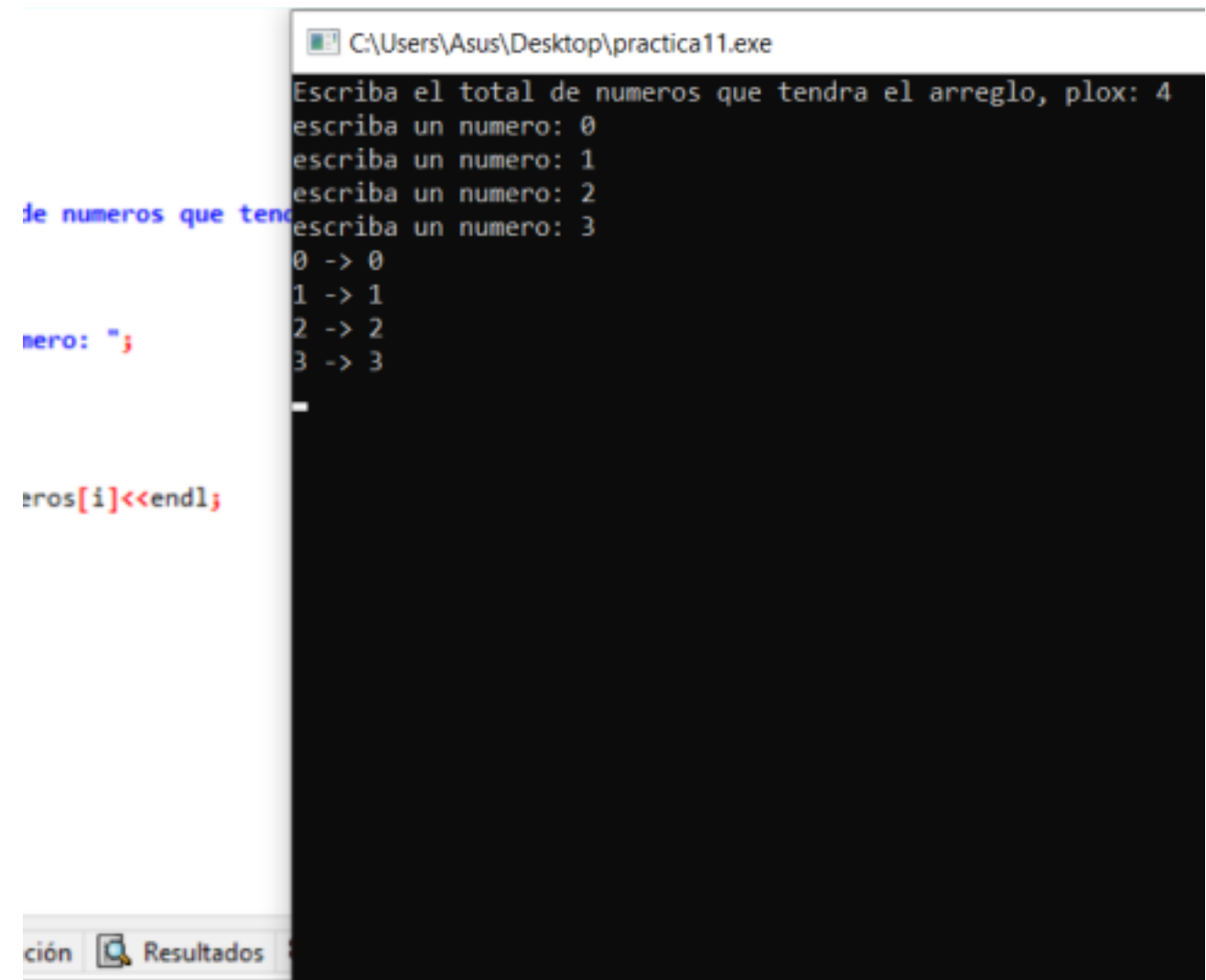
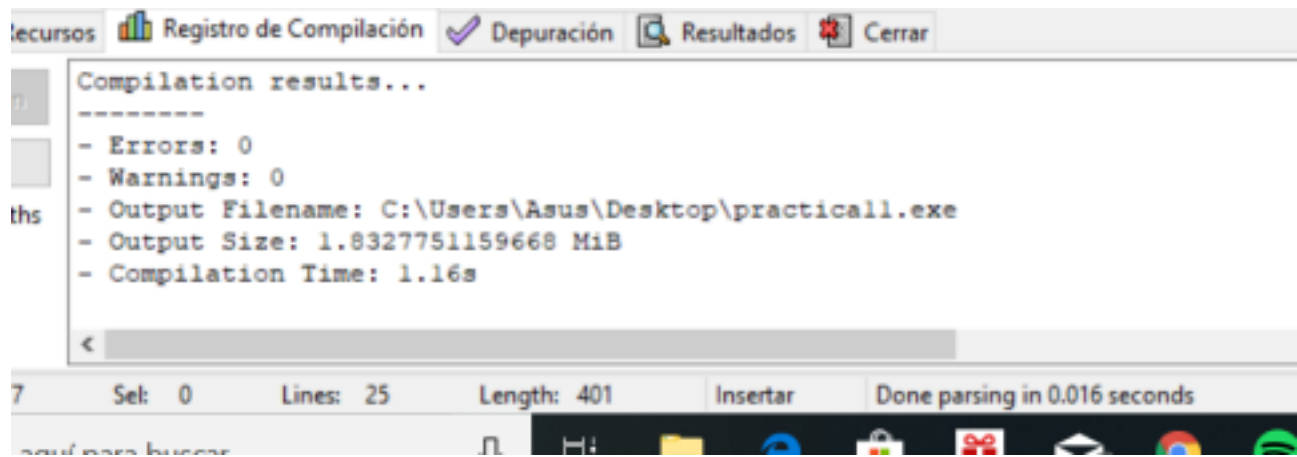
Hacer un programa que:

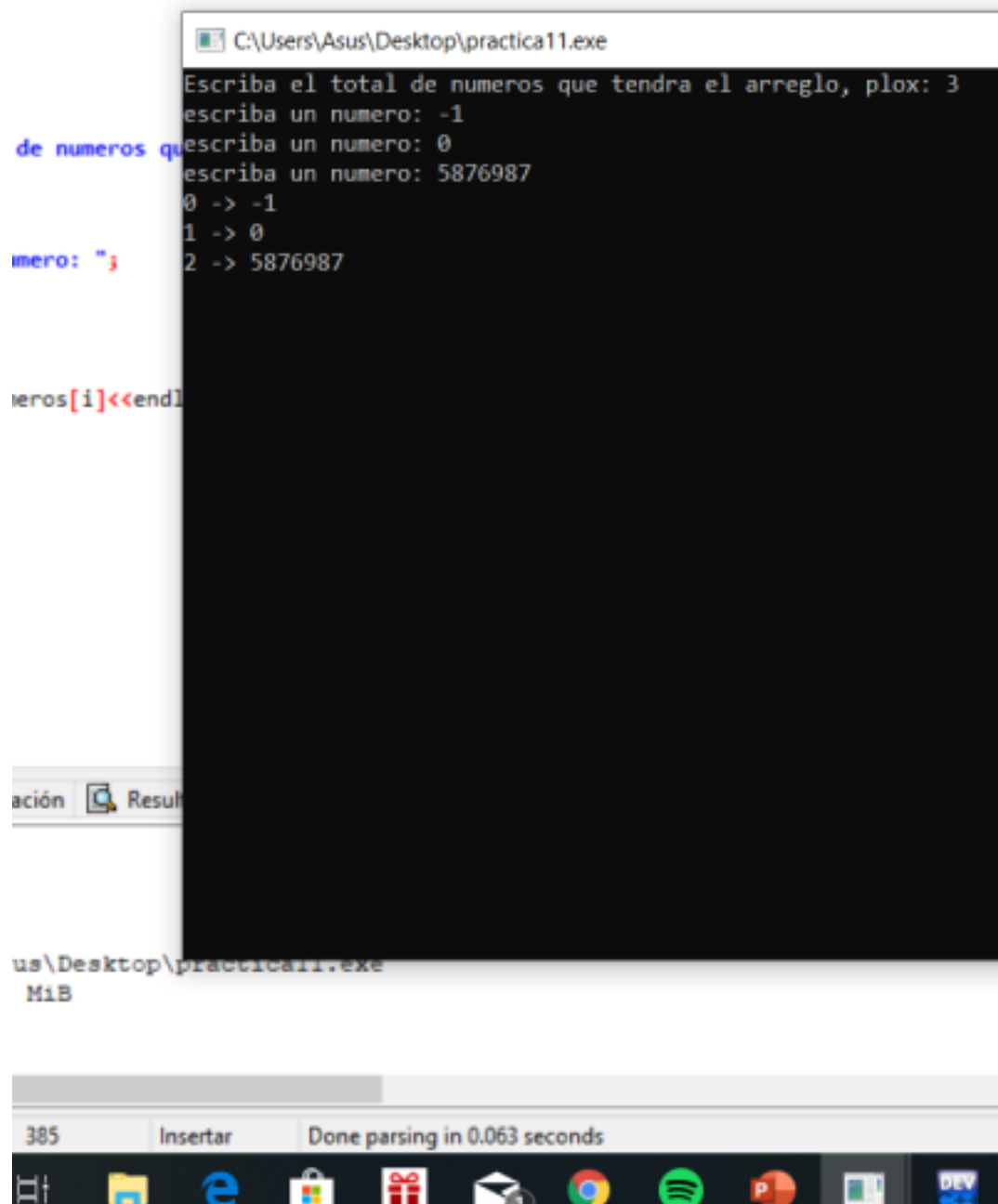
- Pida al usuario un número.
- Genere un arreglo de esa longitud.
- Pida al usuario números suficientes para llenar el arreglo.

En este programa utilizamos diferentes recursos que eran necesarios para ejecutar el programa, lo que el programa hace principalmente es pedir un número, ese número va a ser el total de elementos que tendrá el arreglo, después pedirá el total de números de el número principal ingresado haciendo una lista, esta misma hará una lista la cual contiene a todos los elementos que ingresamos en cierto orden de acomodo del mismo ingresado.

The image shows a screenshot of a code editor window with a toolbar at the top containing icons for file operations, editing, and execution. The editor displays a C++ program named 'practica11.cpp'. The code includes standard headers, defines a constant 'TAMANO' as 5, and uses the 'std' namespace. The 'main' function prompts the user for the total number of elements 'n', then uses a loop to read 'n' numbers into an array 'numeros'. Finally, it prints the array elements and waits for a key press before returning 0.

```
1  #include <stdio.h>
2  #include<iostream>
3  #include<conio.h>
4  #define TAMANO 5
5
6  using namespace std;
7
8  int main(){
9      int numeros[100],n;
10
11      cout<<"Escriba el total de numeros que tendra el arreglo, plox: ";
12      cin>>n;
13
14      for(int i=0;i<n;i++){
15          cout<<"escriba un numero: ";
16          cin>>numeros[i];
17      }
18
19      for(int i=0;i<n;i++){
20          cout<<i<<" -> "<<numeros[i]<<endl;
21      }
22
23      getch();
24      return 0;
25 }
```





Falta la parte de obtener  
el mínimo y el máximo

2.\_

Hacer un programa que:

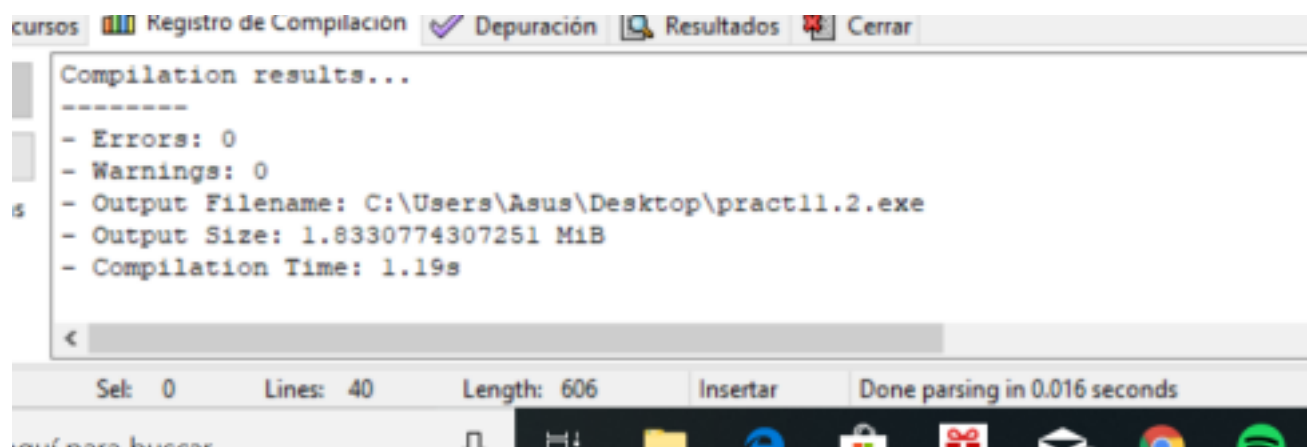
- Pida al usuario un dos números  $N$  y  $M$ .
- Genere dos matrices de  $N \times M$ .
- Pida al usuario números suficientes para llenar ambas matrices.
- Muestre al usuario la matriz resultado de sumar las dos de entrada.

Para hacer este programa usamos diferentes recursos vistos alrededor de la clase y algunos nuevo que quise y eran necesarios para realizar este programa de MATRIZ, lo que hace principalmente este programa es que pide el número de columnas que se quieran y el número de filas que se quieran también, después pide cuantos números sean necesarios para llenar las columnas y filas, posteriormente realiza la operación de la matriz.

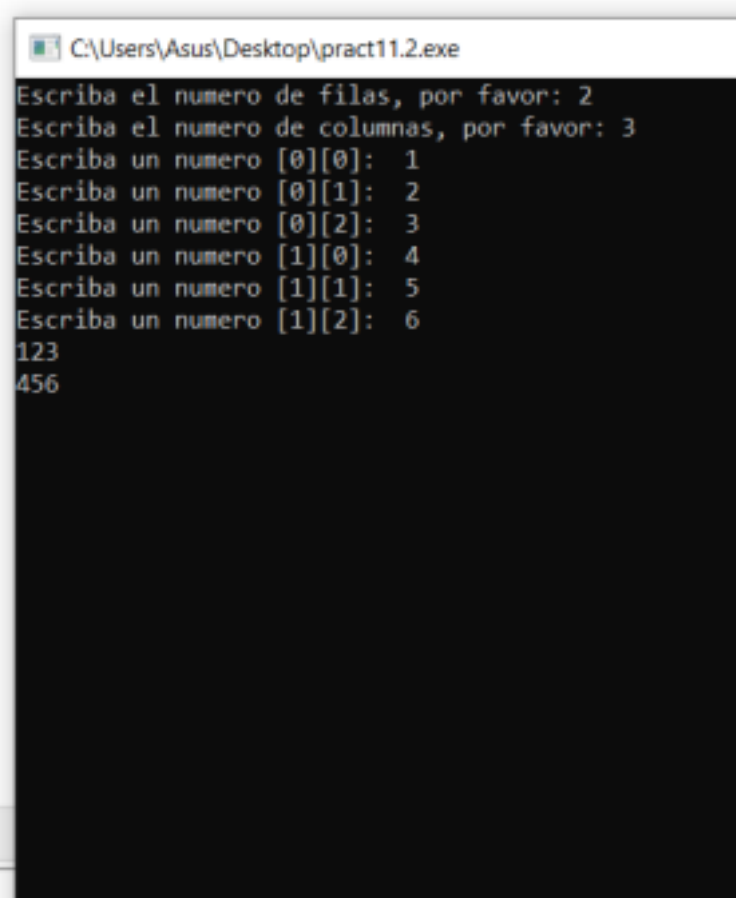
```
[*] pract11.2.cpp
1  #include<stdio.h>
2  #include<iostream>
3  #include<conio.h>
4
5  using namespace std;
6
7  int main(){
8      int numeros[100][100], filas, columnas;
9
10     cout<<"Escriba el numero de filas, por favor: ";
11     cin>>filas;
12     cout<<"Escriba el numero de columnas, por favor: ";
13     cin>>columnas;
14
15     for(int i=0;i<filas;i++){
16
17         for(int j=0;j<columnas;j++){
18             cout<<"Escriba un numero ["<<i<<"]["<<j<<": ";
19             cin>>numeros[i][j];
20         }
21     }
22
23     for(int i=0;i<filas;i++){
24         for(int j=0;j<columnas;j++){
25             cout<<numeros[i][j];
26         }
27         cout<<"\n";
28     }
29     getch();
30     return 0;
31 }
32
33
```

Registro de Compilación Depuración Resultados Cerrar

Compilation results...



```
        columnas;  
  
        , por favor: ";  
  
        columnas, por favor: ";  
  
        ["<<i<<"]("<<j<<"): ";
```



Resultados Cerrar



```
columnas;  
las, por favor: ";  
lumnas, por favor: "  
  
){  
ro ["<<i<<"]["<<j<<
```

```
C:\Users\Asus\Desktop\pract11.2.exe  
Escriba el numero de filas, por favor: 3  
Escriba el numero de columnas, por favor: 3  
Escriba un numero [0][0]: 1  
Escriba un numero [0][1]: 2  
Escriba un numero [0][2]: 3  
Escriba un numero [1][0]: 4  
Escriba un numero [1][1]: 5  
Escriba un numero [1][2]: 6  
Escriba un numero [2][0]: 7  
Escriba un numero [2][1]: 8  
Escriba un numero [2][2]: 9  
123  
456  
789  
_
```

Falta que en el proceso se pida otra matriz y se sumen.

# Conclusiones:

Como ya sabemos, los programas almacenan información en las variables. Hasta ahora, las variables utilizadas sólo guardaban un valor. Sin embargo, en muchos casos los programas necesitarán guardar muchos valores al mismo tiempo, tales como 50 calificaciones, 100 nombres de archivo o 1000 títulos de libros. Cuando los programas necesitan almacenar muchos valores definen un arreglo. Es decir que un **arreglo** es una variable capaz de guardar uno o más valores.

Un arreglo en C++ es un conjunto de datos que se almacenan en memoria de manera contigua con el mismo nombre. Para diferenciar los elementos de un arreglo se utilizan índices detrás del nombre del arreglo y encerrados por []. El elemento 5° (quinto) de un arreglo, es representado por el índice [4], ya que los índices comienzan en 0. Esto significa que un arreglo de 10 elementos tendría los índices del 0 al 9: [0 . . . 9]