



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Laboratorios de computación salas A y B

*Profesor:* ALEJANDRO ESTEBAN PIMENTEL ALARCÓN

*Asignatura:* FUNDAMENTOS DE PROGRAMACIÓN

*Grupo:* 3

*No de Práctica(s):* 11

*Integrante(s):* Carlos Sotelo Leyva  
Alan Cárdenas Belmont Gerardo Gutiérrez Soto

*No. de Equipo de  
cómputo empleado:* 38

*No. de Lista o Brigada:* #2751  
#5783 #9267

*Semestre:* 2020-1

*Fecha de entrega:* 28/10/2019

*Observaciones:* En las capturas que muestran con la ejecución de los programas se puede ver claramente que no se están obteniendo los resultados esperados, deben tener errores en los programas, parece que tienen problemas de sintaxis en sus "scanf"

**CALIFICACIÓN:** 7

## INTRODUCCIÓN:

Los arreglos unidimensionales, son estructuras de datos típicamente estáticas en la mayoría de los lenguajes de programación, usan posiciones de memoria que están contiguas y que se indexan de forma numérica. Un arreglo unidimensional es una lista de valores guardados bajo el mismo nombre y del mismo tipo. Cada valor dentro del arreglo se le conoce como elemento del arreglo.

En C los arreglos se declaran de la siguiente forma:

`TipoDeDato NombreDelArreglo[TamañoDelArreglo];`

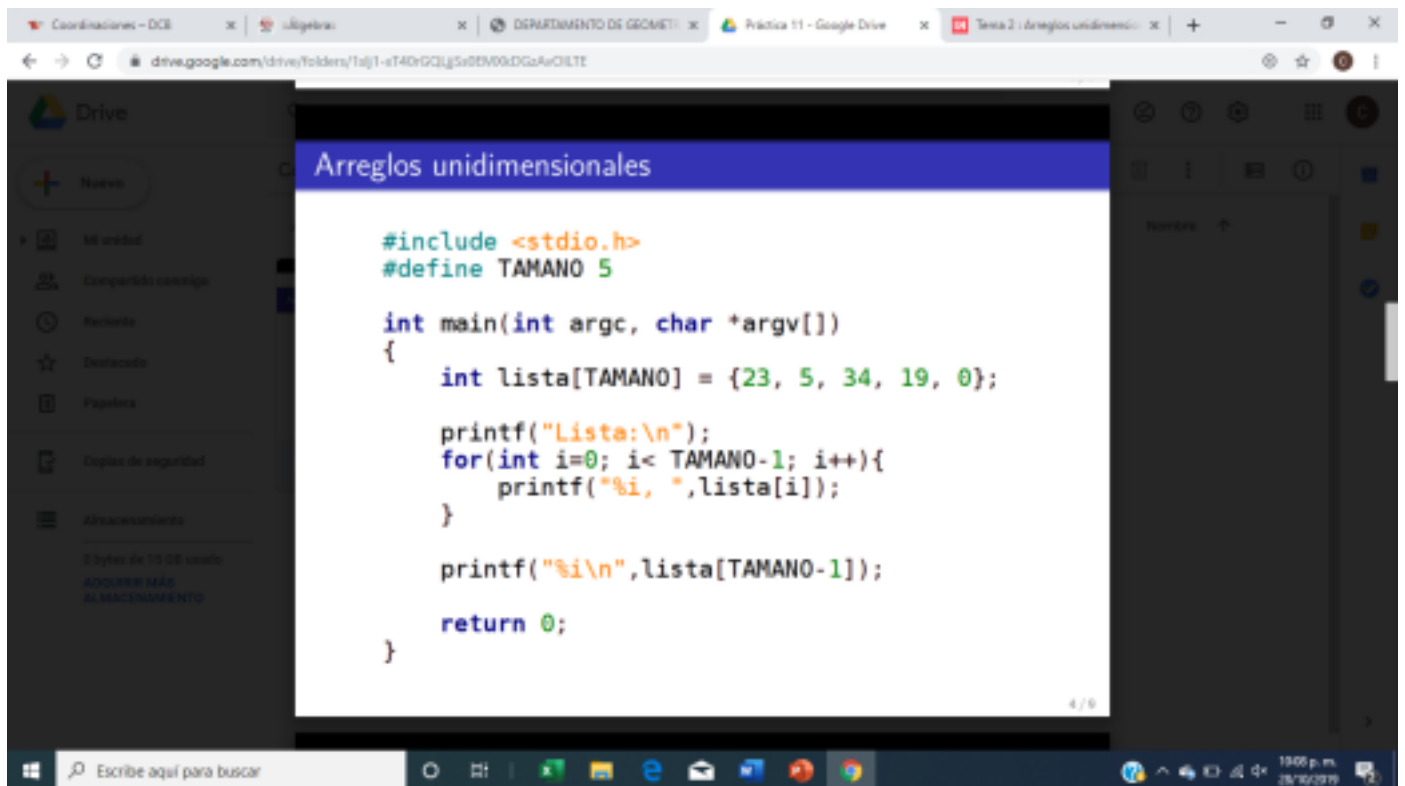
Como podemos ver, es similar a declarar una variable convencional, solo que se coloca entre corchetes el número de posiciones del arreglo, por lo que todas las posiciones serán del mismo tipo. Parra llenar todos los elementos del arreglo es común emplear un ciclo que nos permita recorrer el arreglo desde la primera hasta la última posición.

Objetivo de la práctica:

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.

## ARREGLOS UNIDIMENSIONALES:

En esta práctica veremos los arreglos unidimensionales los cuales nos sirven para poner varios valores dentro de una lista, en la cual nos diga el tamaño o de que tan extensa sea esa lista.



The screenshot shows a Google Drive presentation slide titled "Arreglos unidimensionales". The slide contains the following C code:

```
#include <stdio.h>
#define TAMANO 5

int main(int argc, char *argv[])
{
    int lista[TAMANO] = {23, 5, 34, 19, 0};

    printf("Lista:\n");
    for(int i=0; i< TAMANO-1; i++){
        printf("%i, ",lista[i]);
    }

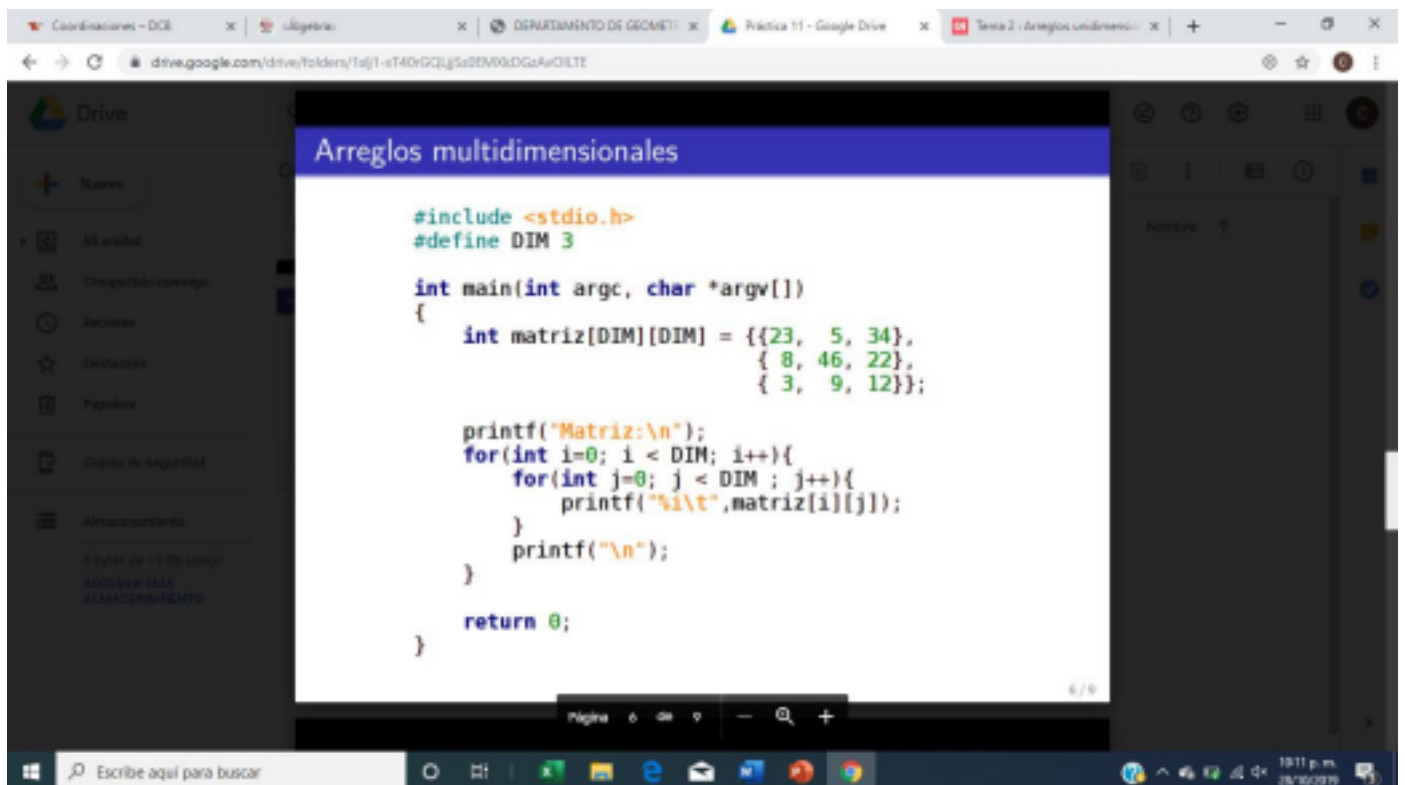
    printf("%i\n",lista[TAMANO-1]);

    return 0;
}
```

The code defines a constant `TAMANO` as 5, declares an array `lista` of size `TAMANO` with values {23, 5, 34, 19, 0}, and prints the elements of the array. The presentation interface includes a sidebar with "Nuevo" and "Mi unidad", and a taskbar at the bottom with various application icons and the system clock showing 10:00 p.m. on 25/10/2019.

## ARREGLOS MULTIDIMENSIONALES:

Como su nombre lo dice, en este tipo de arreglos Podemos meter varias dimensiones o listas dentro de un mismo programa poniendo las variables correspondientes a cada uno.



The screenshot shows a Google Drive presentation slide titled "Arreglos multidimensionales". The slide contains the following C code:

```
#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;
}
```

The code defines a constant `DIM` as 3, declares a 2D array `matriz` of size `DIM` by `DIM` with values {{23, 5, 34}, { 8, 46, 22}, { 3, 9, 12}}, and prints the elements of the matrix. The presentation interface includes a sidebar with "Nuevo" and "Mi unidad", and a taskbar at the bottom with various application icons and the system clock showing 10:11 p.m. on 25/10/2019.

## ACTIVIDAD 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.
- Muestre al usuario el número menor y el mayor de dicho arreglo.

```
1 #include <stdio.h>
2 int main ()
3 {
4     printf("agregue un numero para la longitud deseada\n");
5     scanf("%i", &num);
6     int lista[num];
7
8     for(int i=0; i<num; i++)
9     {
10         printf("lista[%i]\n", i);
11         scanf("%i", &lista[i]);
12     }
13
14     int a;
15     a=lista[0];
16     for(int i=1; i<num; i++)
17     {
18         if(lista[i]<a)
19         {
20             a=lista[i];
21         }
22     }
23
24     int b;
25     b=lista[0];
26     for(int i=1; i<num; i++)
27     {
28         if(lista[i]>b)
29         {
30             b=lista[i];
31         }
32     }
33
34     printf("el numero menor es %i\n", a);
35     printf("el numero mayor es %i\n", b);
36
37     return 0;
38 }
```

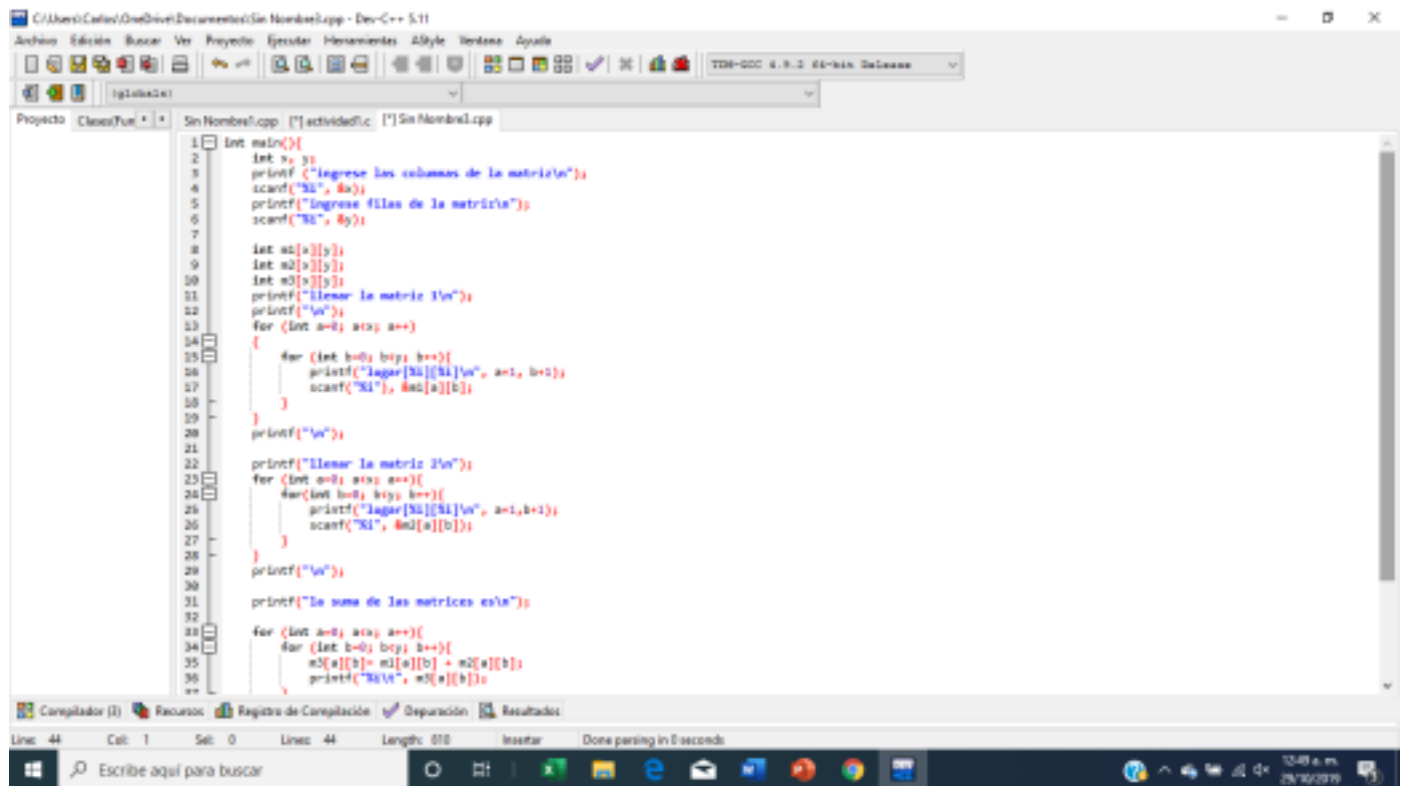
No esta funcionando la parte de mayor y menor

```
agregue un numero para la longitud deseada
lista[0]
1
lista[1]
2
lista[2]
3
lista[3]
4
lista[4]
5
lista[5]
6
lista[6]
7
lista[7]
8
.....
Process exited after 14.19 seconds with return value 0
Presione una tecla para continuar . . . . .
```

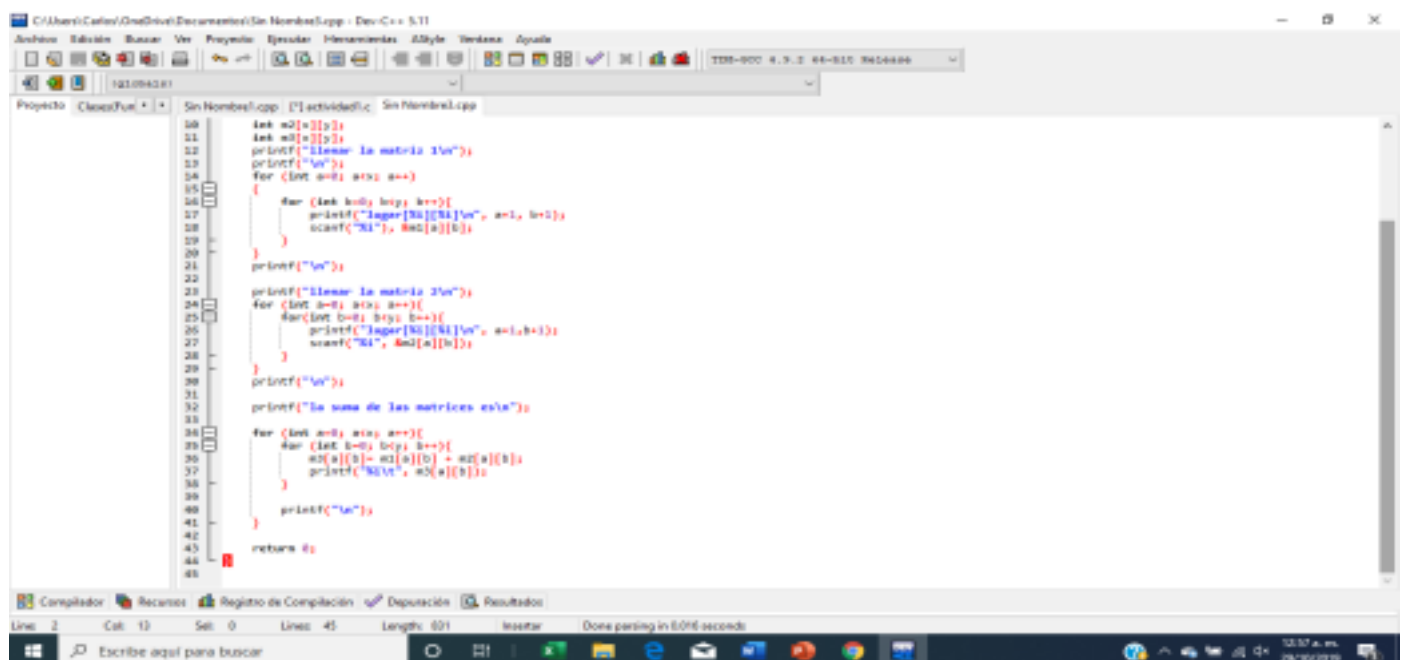
## ACTIVIDAD 2:

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.



```
1 int main(){
2     int n, m;
3     printf("Ingrese las columnas de la matriz\n");
4     scanf("%i", &n);
5     printf("Ingrese filas de la matriz\n");
6     scanf("%i", &m);
7
8     int m1[n][m];
9     int m2[n][m];
10    int m3[n][m];
11    printf("Llenar la matriz 1\n");
12    printf("\n");
13    for (int a=0; a<n; a++){
14        for (int b=0; b<m; b++){
15            printf("Ingrese valor para m1[%i][%i]: ", a, b);
16            scanf("%i", &m1[a][b]);
17        }
18    }
19    printf("\n");
20
21    printf("Llenar la matriz 2\n");
22    for (int a=0; a<n; a++){
23        for (int b=0; b<m; b++){
24            printf("Ingrese valor para m2[%i][%i]: ", a, b);
25            scanf("%i", &m2[a][b]);
26        }
27    }
28    printf("\n");
29
30    printf("La suma de las matrices es\n");
31
32    for (int a=0; a<n; a++){
33        for (int b=0; b<m; b++){
34            m3[a][b] = m1[a][b] + m2[a][b];
35            printf("%i\t", m3[a][b]);
36        }
37    }
38}
```



```
39
40
41    printf("\n");
42
43    return 0;
44}
```

```
C:\Users\Carlos\OneDrive\Documents\Sin Nombres3.exe
lugar[1][1]
2
lugar[2][4]
10
lugar[2][2]
12
lugar[2][3]
15
lugar[1][1]
11
lugar[1][2]
45
lugar[1][3]
89
lugar[2][1]
85
lugar[2][2]
86
lugar[2][3]
88
la suma de las matrices es
4214829 45 6487505
85 86 88
Process exited after 39.35 seconds with return value 0
Presione una tecla para continuar . . .
```

Claramente aquí algo no está funcionando

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Carlos\OneDrive\Documents\Sin Nombres3.exe
- Output Size: 131.4623928125 KiB
- Compilation Time: 0.31s

## Reporte de la práctica:

En esta práctica empleamos los arreglos unidimensionales y los arreglos multidimensionales, ya que para cada una de las actividades nos pedían programas con los diferentes arreglos, y se pudieron realizar a base de los conocimientos aprendidos en clases y practicas anteriores, lo nuevo de esta práctica son los arreglos ya que nosotros no habíamos escuchado sobre esto nunca, y hemos visto que son de gran utilidad para hacer operaciones mas largas simplemente utilizando un arreglo ya se unidimensional o multidimensional.