	Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia	

Laboratorios de computación salas A y B

Profesor: Karina García Morales

Asignatura: Fundamentos de programación

Grupo: 20

No. de práctica(s): 10

Integrante(s): Suzán Herrera Álvaro

No. de lista o brigada: 49

Semestre: 1

Fecha de entrega: 6 de diciembre 2022

Observaciones:

CALIFICACIÓN: _____

Objetivo: El alumno utilizará arreglos de dos dimensiones en la elaboración de programas que resuelvan problemas que requieran agrupar datos del mismo tipo, en estructuras que utilicen dos índices.

Conceptos:

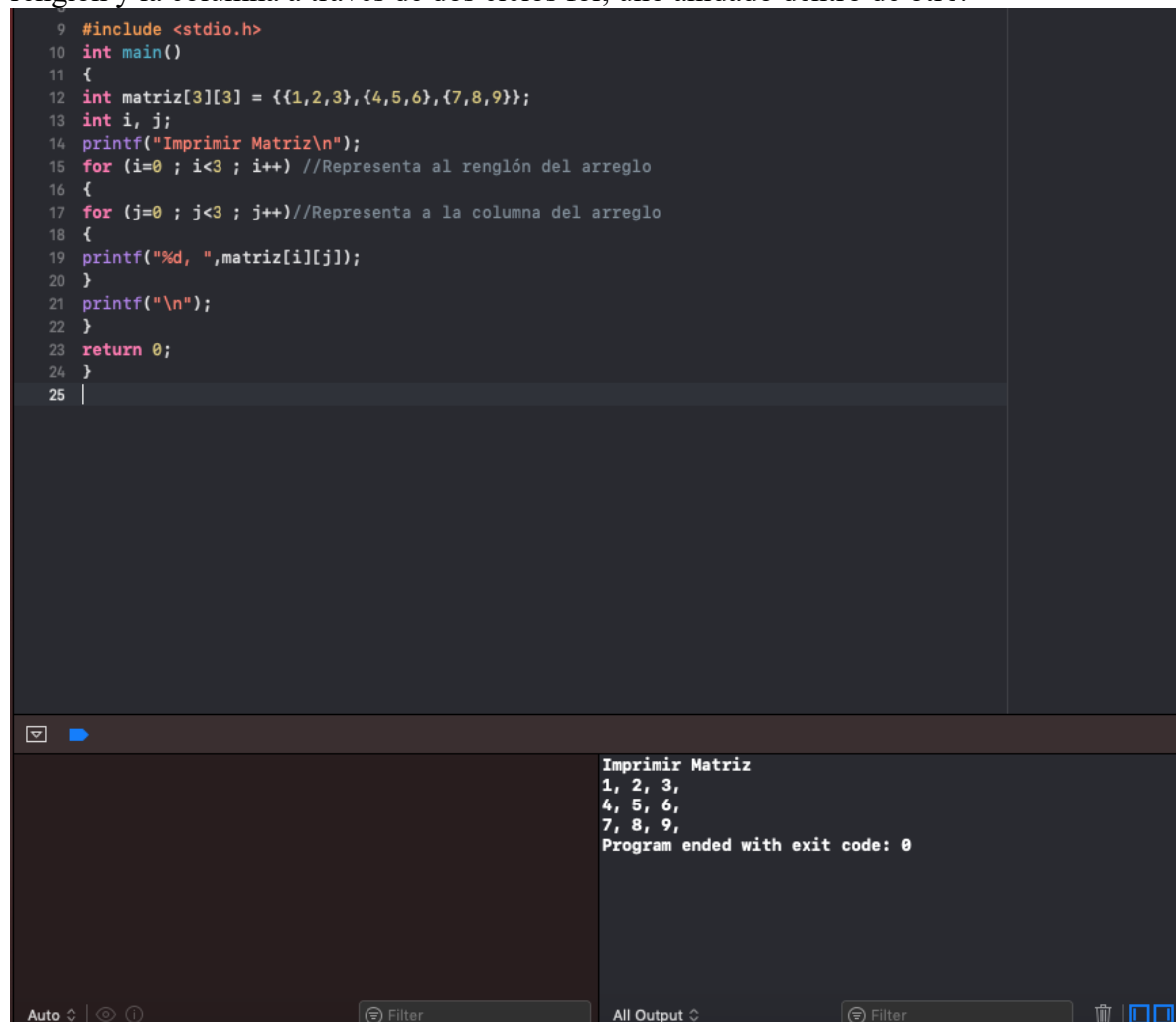
Un apuntador es una variable como cualquier otra. Una variable apuntador contiene una dirección que apunta a otra posición en memoria. En esa posición se almacenan los datos a los que apunta el apuntador. Un apuntador apunta a una variable de memoria.

Los arreglos multidimensionales son una ampliación de las matrices de dos dimensiones y utilizan subíndices adicionales para la indexación. Un arreglo 3D, por ejemplo, utiliza tres subíndices. Los dos primeros son como una matriz, pero la tercera dimensión representa páginas u hojas de elementos.

Códigos (arreglos multidimensionales usando for)

A continuación, se observa un programa que genera un arreglo de dos dimensiones (arreglo multidimensional) y accede a cada uno de sus elementos por la posición que indica el renglón y la columna a través de dos ciclos for, uno anidado dentro de otro.

```
9  #include <stdio.h>
10 int main()
11 {
12     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
13     int i, j;
14     printf("Imprimir Matriz\n");
15     for (i=0 ; i<3 ; i++) //Representa al renglón del arreglo
16     {
17         for (j=0 ; j<3 ; j++)//Representa a la columna del arreglo
18         {
19             printf("%d, ",matriz[i][j]);
20         }
21         printf("\n");
22     }
23     return 0;
24 }
25 |
```



```
Imprimir Matriz
1, 2, 3,
4, 5, 6,
7, 8, 9,
Program ended with exit code: 0
```

El siguiente programa genera un arreglo de dos dimensiones (arreglo multidimensional) y accede a sus elementos por la posición que indica el renglón y la columna a través de dos ciclos for, uno anidado dentro de otro, el contenido de cada elemento de este arreglo es la suma de sus índices.

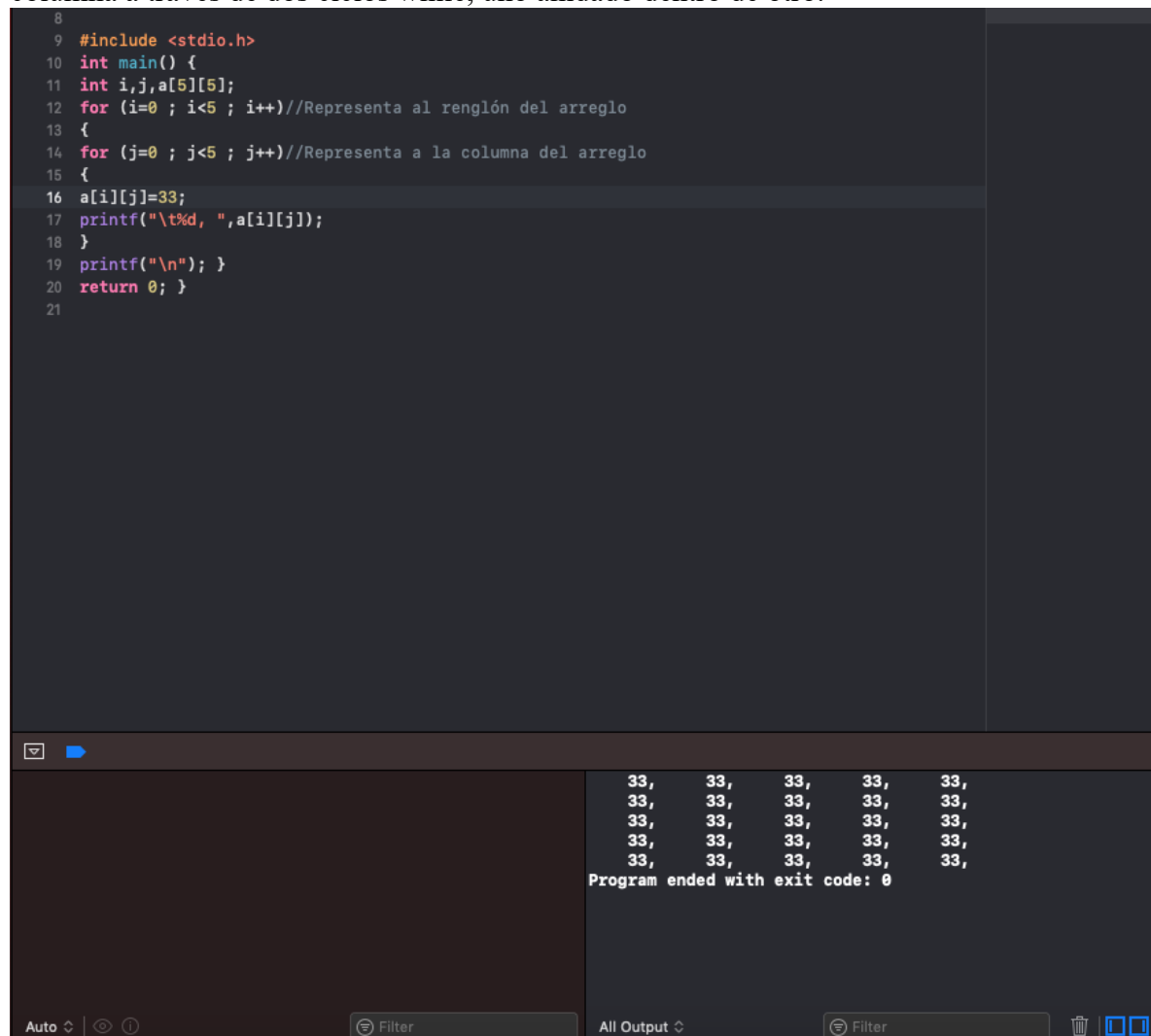
```
8
9  #include <stdio.h>
10 int main() {
11  int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}}; int i, j;
12  printf("Imprimir Matriz\n");
13  i=0;
14  while(i<4) //Representa al renglón del arreglo
15  {
16  j=0;
17  while (j<4)//Representa a la columna del arreglo
18  {
19  printf("%d, ",matriz[i][j]);
20  j++; }
21  printf("\n");
22  i++; }
23  return 0; }
24
```

Imprimir Matriz
1, 2, 3, 4,
4, 5, 6, 7,
7, 8, 9, 32766,
32766, 1776812206, 41979416, -272632544,
Program ended with exit code: 0

Códigos (arreglos multidimensionales usando while)

El código que se observa a continuación genera un arreglo de dos dimensiones (arreglo multidimensional) y accede a sus elementos por la posición que indica el renglón y la columna a través de dos ciclos while, uno anidado dentro de otro.

```
8
9 #include <stdio.h>
10 int main() {
11     int i,j,a[5][5];
12     for (i=0 ; i<5 ; i++)//Representa al renglón del arreglo
13     {
14         for (j=0 ; j<5 ; j++)//Representa a la columna del arreglo
15         {
16             a[i][j]=33;
17             printf("\t%d, ",a[i][j]);
18         }
19         printf("\n"); }
20     return 0; }
21
```



The screenshot shows a C program in a code editor and its execution output. The code defines a 5x5 integer array 'a' and fills it with the value 33 using nested for loops. The output window displays the array contents as a 5x5 grid of '33,' values, followed by the message 'Program ended with exit code: 0'.

33,	33,	33,	33,	33,
33,	33,	33,	33,	33,
33,	33,	33,	33,	33,
33,	33,	33,	33,	33,
33,	33,	33,	33,	33,

Program ended with exit code: 0

```
8
9  #include <stdio.h>
10 int main() {
11     int i,j,a[5][5];
12     for (i=0 ; i<5 ; i++)//Representa al renglón del arreglo
13     {
14         for (j=0 ; j<5 ; j++)//Representa a la columna del arreglo
15         {
16             a[i][j]=i+j+12;
17             printf("\t%d, ",a[i][j]);
18         }
19         printf("\n"); }
20     return 0; }
21
```

```
12,    13,    14,    15,    16,
13,    14,    15,    16,    17,
14,    15,    16,    17,    18,
15,    16,    17,    18,    19,
16,    17,    18,    19,    20,
Program ended with exit code: 0
```

Auto ↕ | 🔍 ⓘ

Filter

All Output ↕

Filter



```
9  #include <stdio.h>
10 int main() {
11     int i,j,a[5][5];
12     for (i=0 ; i<5 ; i++)//Representa al renglón del arreglo
13     {
14         for (j=0 ; j<5 ; j++)//Representa a la columna del arreglo
15         {
16             a[i][j]=i+j;
17             printf("\t%d, ",a[i][j]);
18         }
19         printf("\n"); }
20     return 0; }
21
```

```
0, 1, 2, 3, 4,
1, 2, 3, 4, 5,
2, 3, 4, 5, 6,
3, 4, 5, 6, 7,
4, 5, 6, 7, 8,
Program ended with exit code: 0
```

Enseguida se muestra el código de un programa que permite generar un arreglo de dos dimensiones (arreglo multidimensional) y accede a sus elementos por la posición que indica el renglón y la columna a través de dos ciclos while, uno anidado dentro de otro, el contenido de cada elemento de este arreglo es la suma de sus índices.

```
8
9  #include <stdio.h>
10 int main() {
11     int matriz[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{12,14,15,16}}; int i, j;
12     printf("Imprimir Matriz\n");
13     i=0;
14     while(i<4) //Representa al renglón del arreglo
15     {
16         j=0;
17         while (j<4)//Representa a la columna del arreglo
18         {
19             printf("%d, ",matriz[i][j]);
20             j++; }
21             printf("\n");
22             i++; }
23     return 0; }
24
```



```
Imprimir Matriz
1, 2, 3, 4,
5, 6, 7, 8,
9, 10, 11, 12,
12, 14, 15, 16,
Program ended with exit code: 0
```

Auto ↺ | ⌂ ⓘ

Filter

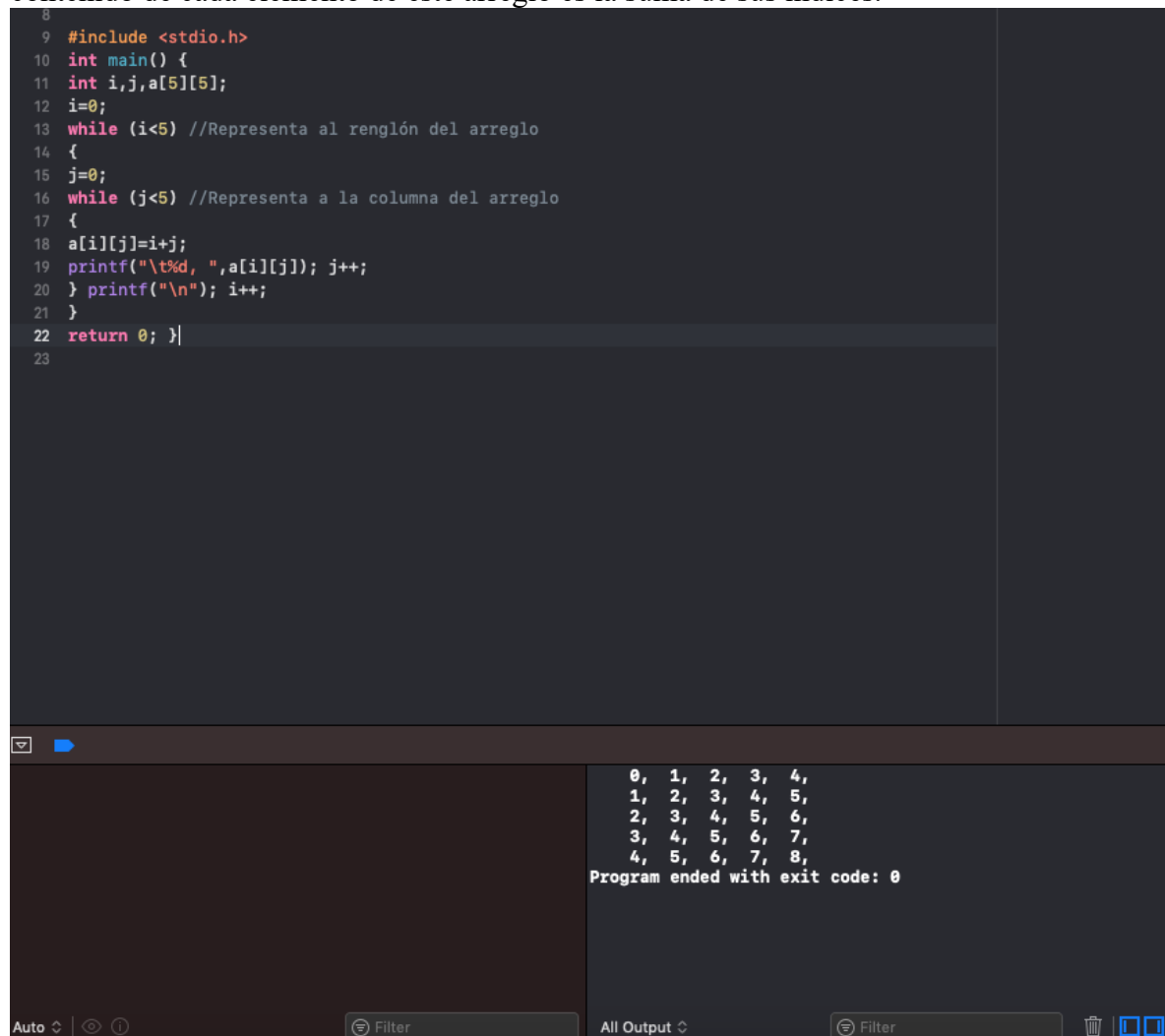
All Output ↺

Filter



Enseguida se muestra el código de un programa que permite generar un arreglo de dos dimensiones (arreglo multidimensional) y accede a sus elementos por la posición que indica el renglón y la columna a través de dos ciclos while, uno anidado dentro de otro, el contenido de cada elemento de este arreglo es la suma de sus índices.

```
8
9  #include <stdio.h>
10 int main() {
11     int i,j,a[5][5];
12     i=0;
13     while (i<5) //Representa al renglón del arreglo
14     {
15         j=0;
16         while (j<5) //Representa a la columna del arreglo
17         {
18             a[i][j]=i+j;
19             printf("\t%d, ",a[i][j]); j++;
20         } printf("\n"); i++;
21     }
22     return 0; }|
23
```

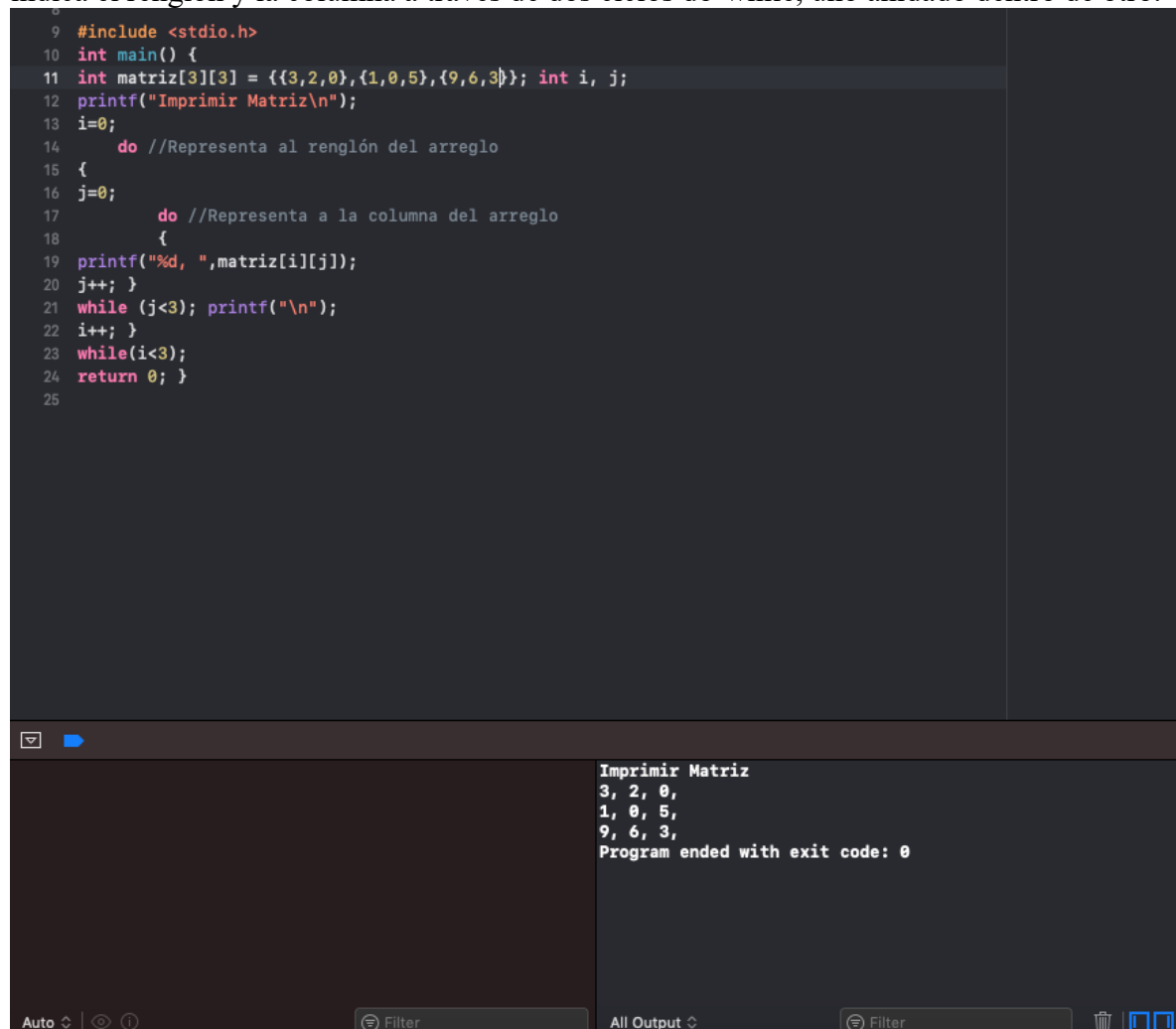


The image shows a code editor with a C program and its output. The program uses two nested while loops to iterate over a 5x5 array 'a'. The outer loop 'i' represents the row index (0 to 4), and the inner loop 'j' represents the column index (0 to 4). For each element 'a[i][j]', the value is calculated as 'i+j' and printed with a tab separator. After each row is completed, a newline is printed. The output window shows the resulting 5x5 grid of values, where each row contains consecutive integers starting from the row index. The program ends with an exit code of 0.

```
0, 1, 2, 3, 4,
1, 2, 3, 4, 5,
2, 3, 4, 5, 6,
3, 4, 5, 6, 7,
4, 5, 6, 7, 8,
Program ended with exit code: 0
```


Códigos (arreglos multidimensionales usando do-while) A continuación, se presenta un código que permite la generación de un arreglo de dos dimensiones (arreglo multidimensional) y se puede acceder a cada uno de sus elementos por la posición que indica el renglón y la columna a través de dos ciclos do-while, uno anidado dentro de otro.

```
8
9 #include <stdio.h>
10 int main() {
11     int matriz[3][3] = {{3,2,0},{1,0,5},{9,6,3}}; int i, j;
12     printf("Imprimir Matriz\n");
13     i=0;
14     do //Representa al renglón del arreglo
15     {
16         j=0;
17         do //Representa a la columna del arreglo
18         {
19             printf("%d, ",matriz[i][j]);
20             j++; }
21         while (j<3); printf("\n");
22         i++; }
23     while(i<3);
24     return 0; }
25
```



The screenshot shows a C program in a code editor. The program defines a 3x3 matrix and uses nested do-while loops to iterate through each element, printing them in a formatted way. The output window shows the matrix being printed row by row, followed by a message indicating the program ended successfully with exit code 0.

Imprimir Matriz
3, 2, 0,
1, 0, 5,
9, 6, 3,
Program ended with exit code: 0

```
8
9  #include <stdio.h>
10 int main() {
11     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}}; int i, j;
12     printf("Imprimir Matriz\n");
13     i=0;
14     do //Representa al renglón del arreglo
15     {
16         j=0;
17         do //Representa a la columna del arreglo
18         {
19             printf("%d, ",matriz[i][j]);
20             j++; }
21         while (j<3); printf("\n");
22         i++; }
23     while(i<3);
24     return 0; }
25
```

```
Imprimir Matriz
1, 2, 3,
4, 5, 6,
7, 8, 9,
Program ended with exit code: 0
```

Auto ↕ | 🔍 ⓘ

Filter

All Output ↕

Filter

🗑️ 📄 📄

Como puede observarse en el programa que se lee a continuación, se genera un arreglo de dos dimensiones (arreglo multidimensional) y se accede a todos sus elementos por la posición que indica el renglón y la columna a través de dos ciclos do-while, uno anidado dentro de otro, el contenido de cada uno de los elementos de este arreglo es la suma de sus índices.

```
9  #include <stdio.h>
10 int main() {
11     int i,j,a[5][5];
12     i=0;
13     do //Representa al renglón del arreglo
14     {
15         j=0;
16         do //Representa a la columna del arreglo
17         {
18             a[i][j]=i+j; printf("\t%d, ",a[i][j]); j++;
19         }
20         while (j<5); printf("\n"); i++;
21     }
22     while (i<5); return 0;
23 }
24
```

```
0, 1, 2, 3, 4,
1, 2, 3, 4, 5,
2, 3, 4, 5, 6,
3, 4, 5, 6, 7,
4, 5, 6, 7, 8,
Program ended with exit code: 0
```

A continuación, se muestra un programa que genera un arreglo multidimensional de máximo 10 renglones y 10 columnas, para poder almacenar datos en cada elemento y posteriormente mostrar el contenido de esos elementos se hace uso de ciclos for anidados respectivamente

```
8
9 #include <stdio.h>
10 int main () {
11     int lista[3][3]; // Se declara el arreglo multidimensional
12     int i,j;
13     int renglon,columna;
14     printf("\nDa el número de renglones y columnas separados con coma\n");
15     scanf("%d,%d",&renglon,&columna);
16     if(((renglon>=1) && (renglon<=3))&&((columna>=1) && (columna<=3)))
17     {
18         // Acceso a cada elemento del arreglo multidimensional usando for
19         for (i= 0 ; i <= renglon-1 ; i++) {
20             for(j= 0 ; j <= columna-1 ; j++) {
21                 printf("\nNúmero para el elemento %d,%d del arreglo", i,j );
22                 scanf("%d",&lista[i][j]);
23             }
24             printf("\nLos valores dados son: \n");
25             // Acceso a cada elemento del arreglo multidimensional usando for
26             for (i= 0 ; i <= renglon-1 ; i++) {
27                 for(j= 0 ; j <= columna-1 ; j++) {
28                     printf("%d ", lista[i][j]);
29                 }
30                 printf("\n");
31             }
32         }
33     else printf("Los valores dados no es válido"); printf("\n");
34     return 0;
35 }
36
```

Da el número de renglones y columnas separados con coma
3

Número para el elemento 0,0 del arreglo3

Número para el elemento 1,0 del arreglo3

Número para el elemento 2,0 del arreglo3

Los valores dados son:

3
3
3

Program ended with exit code: 0

```

8
9 #include <stdio.h>
10 int main() {
11     int matriz[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
12     int i, cont=0, *ap;
13     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0]; printf("Imprimir Matriz\n");
14     i=0;
15     while (i<9)
16     {
17         if (cont == 4) //Se imprimió un renglón y se hace un salto de línea {
18             printf("\n");
19             cont = 0; //Inicia conteo de elementos del siguiente renglón }
20             printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz
21             cont++;
22             i++; }
23     printf("\n");
24     return 0; }
25

```



```

1 2 3 4 5 6 7 8 9
Program ended with exit code: 0

```

Auto ↕

Filter

All Output ↕

Filter



```

8
9  #include <stdio.h>
10 int main() {
11     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
12     int i, cont=0, *ap;
13     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0];
14     printf("Imprimir Matriz\n");
15     i=0;
16     do
17     {
18         if (cont == 3) //Se imprimió un renglón y se hace un salto de línea //ese if sirve para
            salto de renglón
19         {
20             printf("\n");
21             cont = 0; //Inicia conteo de elementos del siguiente renglón
22         }
23         printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz cont++;
24         i++;
25     }
26     while (i<9); //El argumento de este while depende del argumneto que pusimos en if, siendo
        este el cuadrado del argumento de if
27     printf("\n");
28     return 0; }
29

```



```

Imprimir Matriz
1  2  3  4  5  6  7  8  9
Program ended with exit code: 0

```

Auto ↕ | 🔍 ⓘ

⊖ Filter

All Output ↕

⊖ Filter



```

8
9 #include <stdio.h>
10 int main() {
11     int matriz[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
12     int i, cont=0, *ap;
13     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0];
14     printf("Imprimir Matriz\n");
15     i=0;
16     while (i<16)
17     {
18         if (cont == 4) //Se imprimió un renglón y se hace un salto de línea
19         {
20             printf("\n");
21             cont = 0; //Inicia conteo de elementos del siguiente renglón
22         }
23         printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz
24         cont++;
25         i++; }
26     printf("\n");
27     return 0; }
28
29

```



```

Imprimir Matriz
1  2  3  4
5  6  7  8
9  10 11 12
13 14 15 16
Program ended with exit code: 0

```

Auto ↕ | 🔍 ⓘ

🔍 Filter

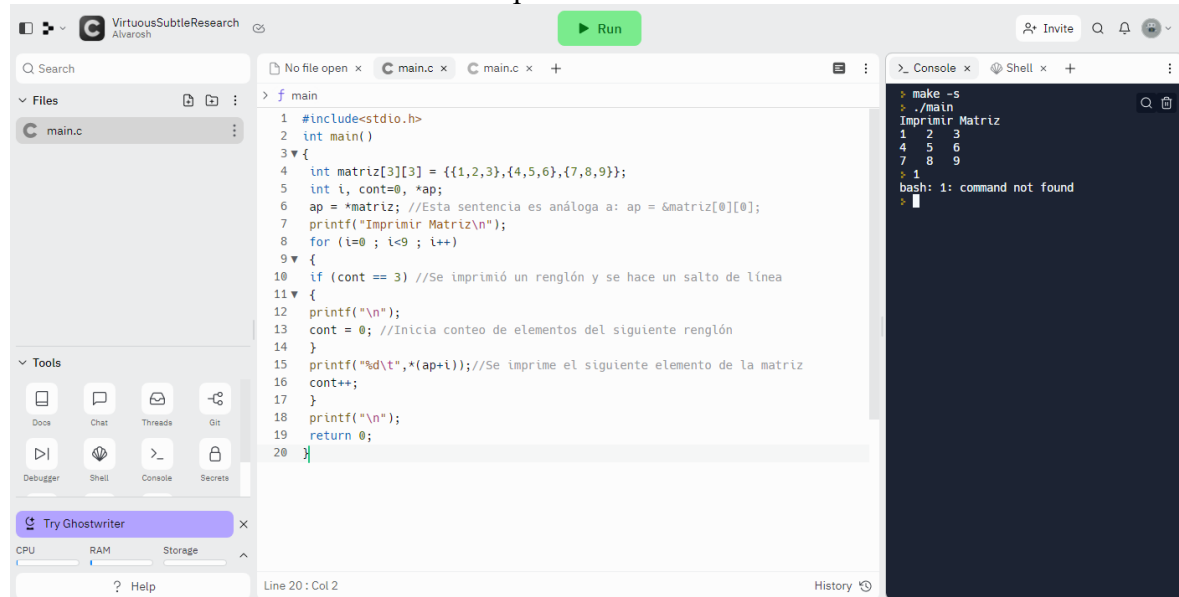
All Output ↕

🔍 Filter



Códigos (arreglos multidimensionales con apuntadores)

El programa siguiente genera un arreglo de dos dimensiones (arreglo multidimensional) y accede a sus elementos a través de un apuntador utilizando un ciclo for.

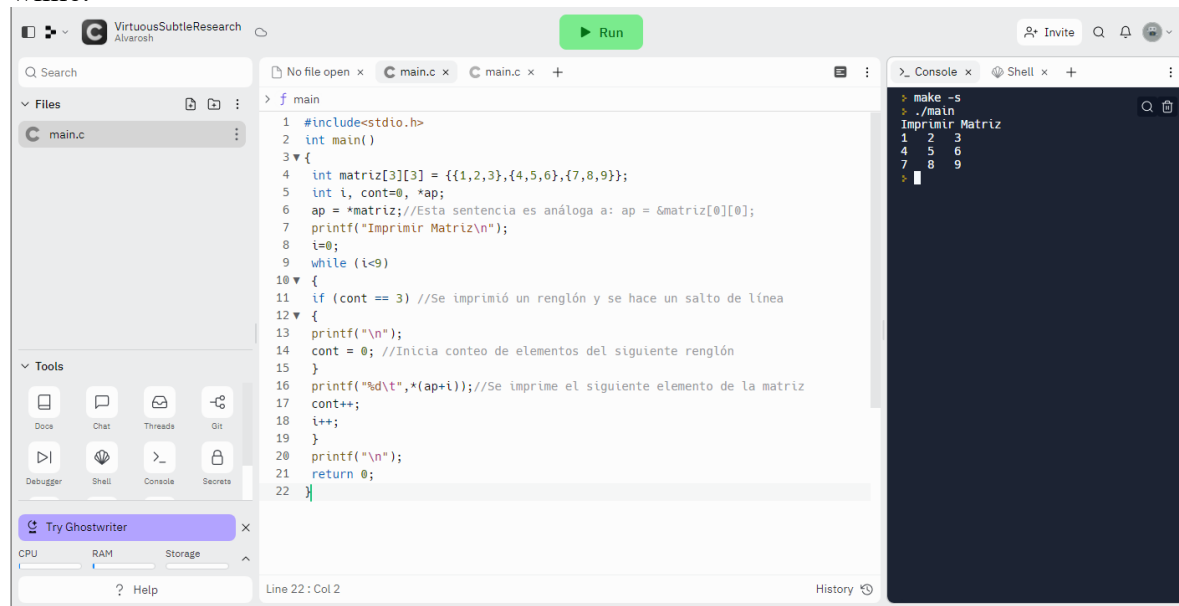


The screenshot shows a code editor with a C program. The program defines a 3x3 matrix and uses a for loop to iterate through its elements, printing them. The console output shows the matrix being printed.

```
1 #include<stdio.h>
2 int main()
3 {
4     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
5     int i, cont=0, *ap;
6     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0];
7     printf("Imprimir Matriz\n");
8     for (i=0 ; i<9 ; i++)
9     {
10        if (cont == 3) //Se imprimió un renglón y se hace un salto de línea
11        {
12            printf("\n");
13            cont = 0; //Inicia conteo de elementos del siguiente renglón
14        }
15        printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz
16        cont++;
17    }
18    printf("\n");
19    return 0;
20 }
```

```
> make -s
> ./main
Imprimir Matriz
1 2 3
4 5 6
7 8 9
>
bash: 1: command not found
```

El código del siguiente programa genera un arreglo de dos dimensiones (arreglo multidimensional) y accede a sus elementos a través de un apuntador utilizando un ciclo while.

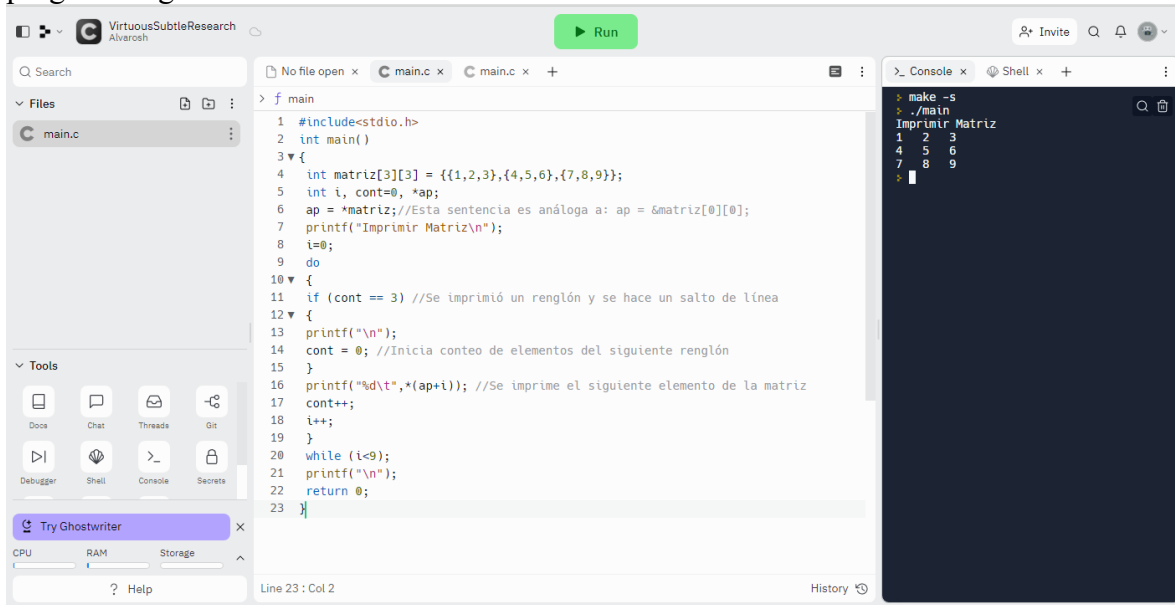


The screenshot shows a code editor with a C program. The program defines a 3x3 matrix and uses a while loop to iterate through its elements, printing them. The console output shows the matrix being printed.

```
1 #include<stdio.h>
2 int main()
3 {
4     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
5     int i, cont=0, *ap;
6     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0];
7     printf("Imprimir Matriz\n");
8     i=0;
9     while (i<9)
10    {
11        if (cont == 3) //Se imprimió un renglón y se hace un salto de línea
12        {
13            printf("\n");
14            cont = 0; //Inicia conteo de elementos del siguiente renglón
15        }
16        printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz
17        cont++;
18        i++;
19    }
20    printf("\n");
21    return 0;
22 }
```

```
> make -s
> ./main
Imprimir Matriz
1 2 3
4 5 6
7 8 9
>
bash: 1: command not found
```


La generación de un arreglo de dos dimensiones (arreglo multidimensional) y el acceso a sus elementos a través de un apuntador utilizando un ciclo do-while se puede observar en el programa siguiente:



The screenshot shows a code editor with a C program that generates and prints a 3x3 matrix. The program uses a do-while loop to iterate through the rows and a pointer to access the elements of the matrix. The output of the program is displayed in the console window.

```
1 #include<stdio.h>
2 int main()
3 {
4     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
5     int i, cont=0, *ap;
6     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0];
7     printf("Imprimir Matriz\n");
8     i=0;
9     do
10    {
11        if (cont == 3) //Se imprimió un renglón y se hace un salto de línea
12        {
13            printf("\n");
14            cont = 0; //Inicia conteo de elementos del siguiente renglón
15        }
16        printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz
17        cont++;
18        i++;
19    }
20    while (i<9);
21    printf("\n");
22    return 0;
23 }
```

The console output shows the matrix being printed:

```
> make -s
> ./main
Imprimir Matriz
1 2 3
4 5 6
7 8 9
```

Tarea:

Arreglo multidimensional

Ej: Realiza un programa que muestre tu nombre
y número de cuenta
include <stdio.h>

```
int main()
{
    int num[9] = {3, 2, 0, 1, 0, 5, 9, 6, 4};
    char nom[5] = {"Alvaro"};
    int i, j = 0;
    printf("el nombre del alumno es: \n");
    for (j = 0; i < 5; i++)
    {
        printf("%c", nom[i]);
    }
    printf("\n el número de cuenta es: \n");
    while (j < 9)
    {
        printf("%d", num[j]);
        j++;
    }
    return 0;
}
```

Modifica el ejercicio 1 con cantidades

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
int nom[9] = {3, 2, 0, 1, 0, 5, 9, 6, 9};
```

```
char nom[5] = {'A', 'l', 'v', 'a', 'r', 'o'};
```

```
int *ap[9];
```

```
int i, j;
```

```
printf("El nombre del alumno es: \n");
```

```
for(i=0; i<5; i++)
```

```
{
```

```
printf("%c", nom[i]);
```

```
}
```

```
printf("\n el número de cuenta es: \n");
```

```
j = *ap
```

```
printf("%d", j);
```

```
j = *(ap+1);
```

```
printf("%d", j);
```

```
j = *(ap+2);
```

```
printf("%d", j);
```

```
j = *(ap+3);
```

```
printf("%d", j);
```

```
j = *(ap+4);
```

```
printf("%d", j);
```

```
j = *(ap+5);
```

```
printf("%d", j);
```

```
j = *(ap+6);
```

```
printf("%d", j);
```

```
j = *(ap+7);
```

```
printf("%d", j);
```

```
j = *(ap+8);
```

```
printf("%d", j);
```

```
return 0;
```

```
}
```


E. 3 Corrigé

```
#include <stdio.h>

main()
{
    int i, j, cont=0, n;
    float m[2][2], s=0, *ap;
    ap = m;
    for(j=0; j<=1; j++)
    {
        for(i=0; i<=1; i++)
        {
            printf("\t\t Teclen el elemento i.d.j del (%d,%d)", i, j+1);
            scanf("%f", &m[i][j]);
            s += m[i][j];
        }
    }
    printf("\t\t La matriz es = \n");
    for(n=0; n<=1; n++)
    {
        printf("%f\t\t", m[n][0]);
        printf("%f\t\t", m[n][1]);
        printf("\n");
        cont++;
    }
    printf("\t\t La suma de los elementos es = %f", s);
    return 0;
}
```

```

#include <stdio.h>
int main()
{
    int i=0, j=0, cont=0, n;
    float M[2][2], s=0, *ap;
    ap = *M;
    for(i=0; i<2; i++)
    {
        for(j=0; j<2; j++)
        {
            printf("\t In Teclear el elemento i,j: ");
            scanf("%f", &M[i][j]);
            s += M[i][j];
        }
    }
    printf("\t La Matriz es: \n");
    for(n=0; n<4; n++)
    {
        if(cont==2)
        {
            printf("\t %.2f \t", *(ap+n));
            cont++;
        }
    }
    printf("\t \n La suma de los elementos = %.2f", s);
    return 0;
}

```

Norma

Conclusiones:

El "overhead" es mucho menor al utilizar punteros a diferencia de la cantidad de "overhead" que podría presentarse al utilizar otros operadores. Esto mejora la administración de memoria de cualquier dispositivo como puede ser un microcontrolador. La asignación de memoria dinámica es otro uso potente que mejora la administración de memoria del sistema ya que la reserva de memoria se realiza en tiempo de ejecución, es decir cuando el programa se está ejecutando.

Los punteros crean código eficiente y rápido ya que están más cerca del hardware. Esto significa que el compilador puede traducir más fácilmente la operación en código de terminal.

Los punteros son importantes para crear aplicaciones, pero son complicados de usar.

<https://github.com/3201050964/Pr-ctica-10>