# Práctica 03. Matrices en C++

Vectores n-dimensionales en C++

U.A.Q. Fac. de Informática

**Dra. Sandra Luz Canchola Magdaleno** 

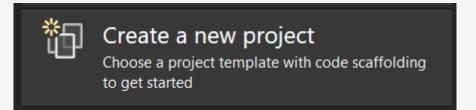
Correo: sandra.canchola@uaq.mx

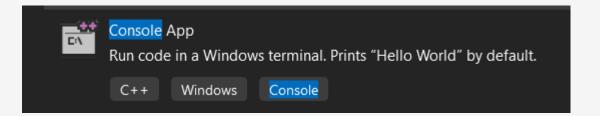
Dra. Reyna Moreno Beltrán

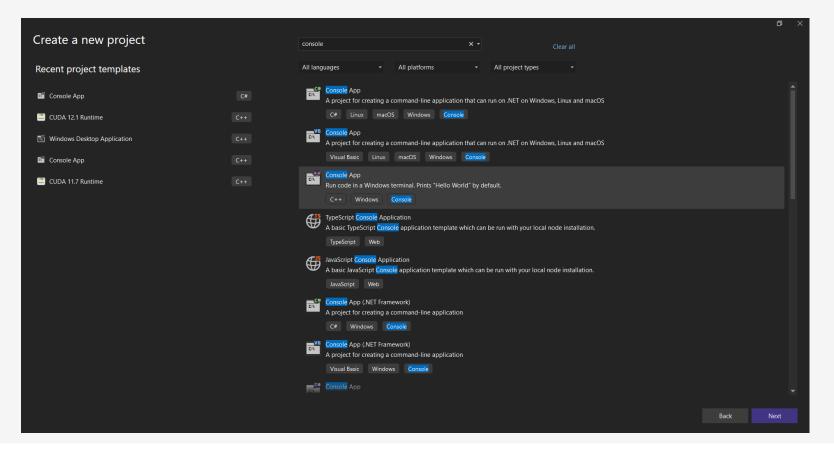
Correo: reyna.moreno@uaq.mx



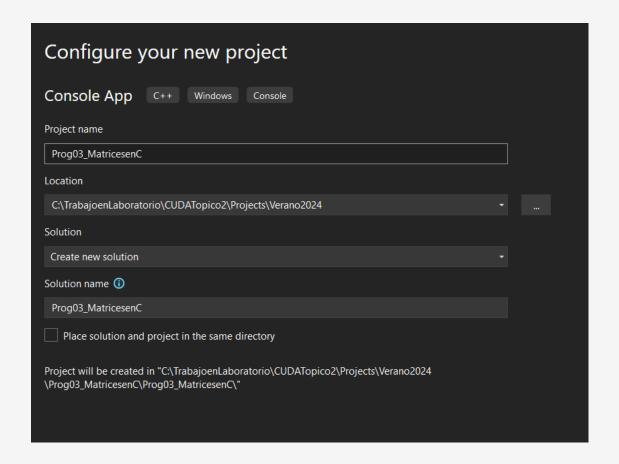
## Crear un C++ Console App

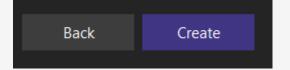






## Crear un C++ Console App





### Comandos

#### Definir una matriz

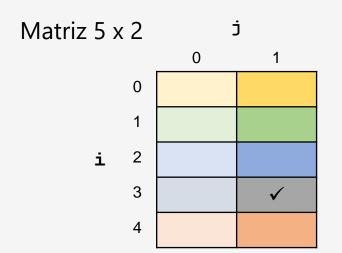
```
Type matName2D[size1][size2];
Type matName3D[size1][size2][size3];
Ejemplo:
float mat1[5][2];
```

#### Definir una matriz como apuntador

```
Type* pointerName;
pointerName = (Type *)malloc(size1 * size2 * sizeof(Type));
pointerName = (Type *)malloc(size1 * size2 * size3 * sizeof(Type));

Ejemplo:
float *array2;
mat2 = (float *)malloc(5 * 2 * sizeof(float));
```

### Elementos de matrices

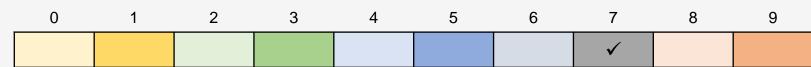


#### Índice de elementos

matriz[i][j]

Ejemplo:
mat1[3][1]

Matriz 5 x 2 como apuntador a una memoria consecutiva de 10 elementos



#### Índice de elementos

#### Ejemplo:

## Elementos de matrices

#### Matriz 5 x 2 x 4

		j												
		0 1												
		<b>k=</b> 0	1	2	3	0	1	2	3					
	0													
		0	1	2	3	0	1	2	3					
	1							✓						
i		0	1	2	3	0	1	2	3					
	2													
		0	1	2	3	0	1	2	3					
	3													
		0	1	2	3	0	1	2	3					
	4													

### Índice de elementos

matriz[i][j][k]

Ejemplo:
mat2[1][1][2]

## Elementos de matrices

#### Matriz 5 x 2 x 4 como apuntador a una memoria consecutiva de 40 elementos

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	 36	37	38	39
														✓					

#### Índice de elementos

```
indice=(i*numCol*numProf)+(j*numProf)+k
Ejemplo:
mat2[1][1][2]
indice=(1*2*4)+(1*4)+2=14
```

### Comandos

#### Acceder a un elemento de la matriz

```
matName2D[i][j]=valor;
matName3D[i][j][k]=valor;

Ejemplo:
mat1[0][1]=15;
mat2[0][1][3]=7;
```

#### Conocer la dirección de memoria de una matriz

```
&matName[0][0]
&matName[0][0][0]

Ejemplo:
&mat1[0][0]
```

### Comandos

#### Calcular la posición de elemento en una matriz

```
pos2D = (filaActual*numCol)+colActual;
pos3D = (filaActual*numCol*numProf)+(colActual*numProf)+profActual;
```

#### Acceder a un elemento de una matriz

```
*(pointerName + pos)=valor;
Ejemplo:
*(matPtr1 + step)=100;
```

Memoria

### CPU (Host)

A01	numRow	2											
A05	numCol	3											
A10	numDep	4											
A15	matriz1												
в01	matriz2												
В10	pos												
В15	matriz3												
C05													
C20	matriz1Ptr	A.	15										
C30	matriz3Ptr	В	15										
E07													

## Código

```
Program 03 Matrix
// Autor: Sandra Luz Canchola Magdaleno
// Email: sandracanchola@yahoo.com y
        sandra.canchola@uaq.mx
#include "stdafx.h"
#include <iostream>
#include <time.h>
using namespace std;
const int numRow = 2;
const int numCol = 3;
const int numDep = 4;
int main(int argc, char* argv[])
         int pos;
         float matrix1[numRow][numCol];
         float *matrix2;
         float matrix3[numRow][numCol][numDep];
         float *mat1Ptr;
         float *mat3Ptr;
         mat1Ptr = &matrix1[0][0];
         matrix2 = (float *)malloc(numRow * numCol * sizeof(float));
         mat3Ptr = &matrix3[0][0][0];
         srand((unsigned) time(NULL));
```

## Código

```
for (int i = 0; i < numRow; ++i) {
          for (int j = 0; j < numCol; ++j) {
                     pos = (i * numCol) + j;
                     matrix1[i][j] = ((float)rand() / (float)RAND MAX) * 100;
                     matrix2[pos] = ((float)rand() / (float)RAND MAX) * 100;
for (int i = 0; i < numRow; ++i) {
          for (int j = 0; j < numCol; ++j) {
                     for (int k = 0; k < numDep; ++k) {
                                matrix3[i][j][k] = ((float)rand() / (float)RAND MAX) * 100;
cout << "========\n";
cout << "The memory addresses of the variables are:\n";</pre>
cout << " A) matrix1 is: " << &matrix1[0][0] << "\n";</pre>
cout << " B) mat1Ptr is: " << &mat1Ptr << "\n";</pre>
cout << " C) matrix2 is: " << &matrix2 << "\n";</pre>
cout << " D) matrix3 is: " << &matrix3[0][0][0] << "\n";</pre>
cout << " E) mat3Ptr is: " << &mat3Ptr << "\n";</pre>
cout << "=======\n";
cout << "The contents of the pointer variables are:\n";</pre>
cout << " A) mat1Ptr is: " << mat1Ptr << "\n";</pre>
cout << " B) matrix2 is: " << matrix2 << "\n";</pre>
cout << " C) mat3Ptr is: " << mat3Ptr << "\n";</pre>
cout << "=======\n";
cout << "The contents of the matrices are:\n";</pre>
for (int i = 0; i < numRow; ++i) {
          for (int j = 0; j < numCol; ++j) {
                     pos = (i * numCol) + j;
                     cout << pos + 1 << ") matrix1[" << i << "][" << j << "]: ";</pre>
                     cout << matrix1[i][j] << " -- matrix2[" << pos << "]: ";</pre>
                     cout << matrix2[pos] << "\n";</pre>
```

```
cout << "========n";
cout << "The updated content of the matrices are:\n";</pre>
for (int i = 0; i < numRow; ++i) {
          for (int j = 0; j < numCol; ++j) {
                     pos = (i * numCol) + j;
                     matrix1[i][j] = matrix1[i][j] + 10;
                     *(mat1Ptr + pos) = *(mat1Ptr + pos) + 20;
                     matrix2[pos] = matrix2[pos] + 20;
                     \star (matrix2 + pos) = \star (matrix2 + pos) + 30;
                     cout << pos + 1 << ") matrix1[" << i << "][" << j << "]: ";</pre>
                     cout << *(mat1Ptr + pos) << " -- matrix2[" << pos << "]: ";</pre>
                     cout << matrix2[pos] << "\n";</pre>
cout << "========n";
cout << "The content of the 3D matrix is:\n";</pre>
for (int i = 0; i < numRow; ++i) {
          for (int j = 0; j < numCol; ++j) {
                     for (int k = 0; k < numDep; ++k) {
                               pos = i * (numCol * numDep) + j * (numDep) + k;
                                cout << pos + 1 << ") matrix3[" << i << "][" << j << "][";
                                cout << k << "]: " << matrix3[i][j][k] << " -- matrix3[";</pre>
                                cout << pos << "]: " << mat3Ptr[pos] << "\n";</pre>
```

```
cout << "========n";
cout << "The updated content of the matrix is:\n";</pre>
for (int i = 0; i < numRow; ++i) {
          for (int j = 0; j < numCol; ++j) {
                    for (int k = 0; k < numDep; ++k) {
                              pos = i * (numCol * numDep) + j * (numDep) + k;
                             matrix3[i][j][k] = matrix3[i][j][k] + 10;
                              mat3Ptr[pos] = mat3Ptr[pos] + 20;
                              *(mat3Ptr + pos) = *(mat3Ptr + pos) + 20;
                              cout << pos + 1 << ") matrix3[" << i << "][" << j << "][";</pre>
                              cout << k << "]: " << matrix3[i][j][k] << " -- matrix3[";</pre>
                              cout << pos << "]: " << *(mat3Ptr + pos) << "\n";</pre>
cout << "========n";
free (matrix2);
system("pause");
return 0;
```

### Corrida

```
© C:\TrabajoenLaboratorio\CUD × + v
 The memory addresses of the variables are:
 A) matrix1 is: 0000002C5C0FF698
 B) mat1Ptr is: 0000002C5C0FF768
 C) matrix2 is: 0000002C5C0FF6C8
 D) matrix3 is: 0000002C5C0FF6F0
 E) mat3Ptr is: 0000002C5C0FF788
______
The contents of the pointer variables are:
 A) mat1Ptr is: 0000002C5C0FF698
 B) matrix2 is: 0000016ECE3DA270
 C) mat3Ptr is: 0000002C5C0FF6F0
The contents of the matrices are:
1) matrix1[0][0]: 94.8973 -- matrix2[0]: 77.8375
2) matrix1[0][1]: 5.32853 -- matrix2[1]: 68.276
3) matrix1[0][2]: 70.0583 -- matrix2[2]: 62.1387
4) matrix1[1][0]: 35.7524 -- matrix2[3]: 88.9187
5) matrix1[1][1]: 86.6512 -- matrix2[4]: 34.0281
6) matrix1[1][2]: 83.404 -- matrix2[5]: 71.0593
The updated content of the matrices are:
1) matrix1[0][0]: 124.897 -- matrix2[0]: 127.837
2) matrix1[0][1]: 35.3285 -- matrix2[1]: 118.276
3) matrix1[0][2]: 100.058 -- matrix2[2]: 112.139
4) matrix1[1][0]: 65.7524 -- matrix2[3]: 138.919
5) matrix1[1][1]: 116.651 -- matrix2[4]: 84.0281
6) matrix1[1][2]: 113.404 -- matrix2[5]: 121.059
______
The content of the 3D matrix is:
1) matrix3[0][0][0]: 19.8523 -- matrix3[0]: 19.8523
2) matrix3[0][0][1]: 30.903 -- matrix3[1]: 30.903
3) matrix3[0][0][2]: 97.2594 -- matrix3[2]: 97.2594
4) matrix3[0][0][3]: 34.2051 -- matrix3[3]: 34.2051
5) matrix3[0][1][0]: 61.7664 -- matrix3[4]: 61.7664
6) matrix3[0][1][1]: 74.572 -- matrix3[5]: 74.572
7) matrix3[0][1][2]: 67.6138 -- matrix3[6]: 67.6138
8) matrix3[0][1][3]: 90.347 -- matrix3[7]: 90.347
9) matrix3[0][2][0]: 77.8558 -- matrix3[8]: 77.8558
10) matrix3[0][2][1]: 14.3712 -- matrix3[9]: 14.3712
11) matrix3[0][2][2]: 59.1937 -- matrix3[10]: 59.1937
```

```
© C:\TrabajoenLaboratorio\CUD ×
12) matrix3[0][2][3]: 57.0757 -- matrix3[11]: 57.0757
13) matrix3[1][0][0]: 30.0272 -- matrix3[12]: 30.0272
14) matrix3[1][0][1]: 70.5069 -- matrix3[13]: 70.5069
15) matrix3[1][0][2]: 43.6811 -- matrix3[14]: 43.6811
16) matrix3[1][0][3]: 72.515 -- matrix3[15]: 72.515
17) matrix3[1][1][0]: 94.406 -- matrix3[16]: 94.406
18) matrix3[1][1][1]: 93.1639 -- matrix3[17]: 93.1639
19) matrix3[1][1][2]: 84.7957 -- matrix3[18]: 84.7957
20) matrix3[1][1][3]: 52.5864 -- matrix3[19]: 52.5864
21) matrix3[1][2][0]: 51.9639 -- matrix3[20]: 51.9639
22) matrix3[1][2][1]: 3.12815 -- matrix3[21]: 3.12815
23) matrix3[1][2][2]: 79.0185 -- matrix3[22]: 79.0185
24) matrix3[1][2][3]: 81.3227 -- matrix3[23]: 81.3227
The updated content of the matrix is:
1) matrix3[0][0][0]: 69.8523 -- matrix3[0]: 69.8523
2) matrix3[0][0][1]: 80.903 -- matrix3[1]: 80.903
3) matrix3[0][0][2]: 147.259 -- matrix3[2]: 147.259
  matrix3[0][0][3]: 84.2051 -- matrix3[3]: 84.2051
5) matrix3[0][1][0]: 111.766 -- matrix3[4]: 111.766
  matrix3[0][1][1]: 124.572 -- matrix3[5]: 124.572
7) matrix3[0][1][2]: 117.614 -- matrix3[6]: 117.614
8) matrix3[0][1][3]: 140.347 -- matrix3[7]: 140.347
9) matrix3[0][2][0]: 127.856 -- matrix3[8]: 127.856
10) matrix3[0][2][1]: 64.3712 -- matrix3[9]: 64.3712
11) matrix3[0][2][2]: 109.194 -- matrix3[10]: 109.194
12) matrix3[0][2][3]: 107.076 -- matrix3[11]: 107.076
13) matrix3[1][0][0]: 80.0272 -- matrix3[12]: 80.0272
14) matrix3[1][0][1]: 120.507 -- matrix3[13]: 120.507
15) matrix3[1][0][2]: 93.6811 -- matrix3[14]: 93.6811
16) matrix3[1][0][3]: 122.515 -- matrix3[15]: 122.515
17) matrix3[1][1][0]: 144.406 -- matrix3[16]: 144.406
18) matrix3[1][1][1]: 143.164 -- matrix3[17]: 143.164
19) matrix3[1][1][2]: 134.796 -- matrix3[18]: 134.796
20) matrix3[1][1][3]: 102.586 -- matrix3[19]: 102.586
21) matrix3[1][2][0]: 101.964 -- matrix3[20]: 101.964
22) matrix3[1][2][1]: 53.1281 -- matrix3[21]: 53.1281
23) matrix3[1][2][2]: 129.019 -- matrix3[22]: 129.019
24) matrix3[1][2][3]: 131.323 -- matrix3[23]: 131.323
Presione una tecla para continuar . . .
```

## Bibliografía

- Herbert Schildt. **C++: The Complete Reference**, 4ta Edición. Edit. McGraw-Hill/Osborne, 2003. ISBN 0-07-222680-3.
- Marc Gregoire. **Professional C++.** 5ta. Edición. Edit. Wiley and sons, 2021. ISBN: 978-1-119-69540-0.
- Bjarne Stroustrup. **The C++ Programming Language**, 4ta. Edición. Addison Wesley, 2013. ISBN 978-0-321-56384-2
- Página <u>www.cplusplus.com</u>

## Gracias por su atención

U.A.Q. Fac. de Informática Campus Juriquilla

Dra. Sandra Luz Canchola Magdaleno sandra.canchola@uaq.mx Cel. 442-1369270

Dra. Reyna Moreno Beltrán reyna.moreno@uaq.mx

DRA. + Sandra Luz
CANCHOLA
MAGDALENO