Let us suppose a two-dimensional array

**int** matrix[3][3];

Copy

For the above array,

matrix => Points to base address of two-dimensional array.

Since array decays to pointer.

\*(matrix) => Points to first row of two-dimensional array.

\*(matrix + 0) => Points to first row of two-dimensional array.

\*(matrix + 1) => Points to second row of two-dimensional array.

\*\*matrix => Points to matrix[0][0]

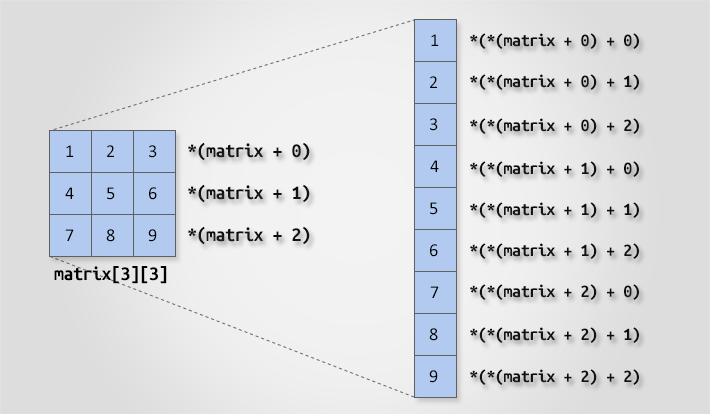
\*(\*(matrix + 0)) => Points to matrix[0][0]

\*(\*(matrix + 0) + 0) => Points to matrix[0][0]

\*(\*matrix + 1) => Points to matrix[0][1]

\*(\*(matrix + 0) + 1) => Points to matrix[0][1]

\*(\*(matrix + 2) + 2) => Points to matrix[2][2]



To access two dimensional array using pointer.

#include <stdio.h>

#define ROWS 3

#define COLS 3

void inputMatrix(int matrix[][COLS], int rows, int cols);

void printMatrix(int matrix[][COLS], int rows, int cols);

int main()

{

int matrix[ROWS][COLS];

int i, j;

printf("Enter elements in %dx%d matrix.\n", ROWS, COLS);

inputMatrix(matrix, ROWS, COLS);

printf("Elements of %dx%d matrix.\n", ROWS, COLS);

printMatrix(matrix, ROWS, COLS);

}

void inputMatrix(int matrix[][COLS], int rows, int cols)

{

int i, j;

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

{

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

{

// (\*(matrix + i) + j is equivalent to &matrix[i][j]

scanf("%d", (\*(matrix + i) + j));

}

}

}

void printMatrix(int (\*matrix)[COLS], int rows, int cols)

{

int i, j;

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

{

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

{

// \*(\*(matrix + i) + j) is equivalent to matrix[i][j]

printf("%d ", \*(\*(matrix + i) + j));

}

printf("\n");

}

}

**Declare a Two Dimensional Array of pointers**

#include <stdio.h>

int main()

{

int arr1[5][5] = { { 0, 1, 2, 3, 4 },

{ 2, 3, 4, 5, 6 },

{ 4, 5, 6, 7, 8 },

{ 5, 4, 3, 2, 6 },

{ 2, 5, 4, 3, 1 } };

int\* arr2[5][5];

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

{

for (int j = 0; j < 5; j++)

{

arr2[i][j] = &arr1[i][j];

}

}

printf("The values are\n");

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

{

for (int j = 0; j < 5; j++)

{

printf("%d ", \*arr2[i][j]);

}

printf("\n");

}

}

**Creating 2D array of pointers using Dynamic Memory allocation**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int arr1[5][5] = { { 0, 1, 2, 3, 4 },

{ 2, 3, 4, 5, 6 },

{ 4, 5, 6, 7, 8 },

{ 5, 4, 3, 2, 6 },

{ 2, 5, 4, 3, 1 } };

int\*\*\* arr2 = malloc(5 \* sizeof(int\*\*));

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

arr2[i] = malloc(5 \* sizeof(int\*));

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

{

for (int j = 0; j < 5; j++)

{

arr2[i][j] = &arr1[i][j];

}

}

printf("The values are\n");

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

{

for (int j = 0; j < 5; j++)

{

printf("%d ", \*arr2[i][j]);

}

printf("\n");

}

}

**Addition of two matrix using pointers**

#include <stdio.h>

#define ROWS 3

#define COLS 3

/\* Function declaration to input, add and print matrix \*/

void matrixInput(int mat[][COLS]);

void matrixPrint(int mat[][COLS]);

void matrixAdd(int mat1[][COLS], int mat2[][COLS], int res[][COLS]);

int main()

{

int mat1[ROWS][COLS], mat2[ROWS][COLS], res[ROWS][COLS];

printf("Enter elements in first matrix of size %dx%d: \n", ROWS, COLS);

matrixInput(mat1);

printf("\nEnter elemetns in second matrix of size %dx%d: \n", ROWS, COLS);

matrixInput(mat2);

matrixAdd(mat1, mat2, res);

printf("\nSum of first and second matrix: \n");

matrixPrint(res);

}

void matrixInput(int mat[][COLS])

{

int i, j;

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

{

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

{

// (\*(mat + i) + j) is equal to &mat[i][j]

scanf("%d", (\*(mat + i) + j));

}

}

}

void matrixPrint(int mat[][COLS])

{

int i, j;

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

{

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

{

// \*(\*(mat + i) + j) is equal to mat[i][j]

printf("%d ", \*(\*(mat + i) + j));

}

printf("\n");

}

}

void matrixAdd(int mat1[][COLS], int mat2[][COLS], int res[][COLS])

{

int i, j;

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

{

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

{

// res[i][j] = mat1[i][j] + mat2[i][j]

\*(\*(res + i) + j) = \*(\*(mat1 + i) + j) + \*(\*(mat2 + i) + j);

}

}

}