**Programming Fundamentals**

|  |  |
| --- | --- |
| Lab 9 | |
| **Topic** | 2D ARRAY (single Pointer / Double Pointer) |

## Pointer to a 2D Array in C++

If we can have a pointer to an integer, a pointer to a float, a pointer to a char, then can we not have a pointer to an array? We certainly can. The following program shows how to build and use it.

#include<iostream>  
using namespace std;  
/\* Usage of pointer to an array \*/   
main( )   
{

int s[5][2] = {

{1, 2},

{1, 2},

{1, 2},

{1, 2}

} ;

int (\*p)[2] ;

int i, j;

for (i = 0 ; i <= 3 ; i++)

{

p=&s[i];

cout<<"Row"<<i<<":";

for (j = 0; j <= 1; j++)

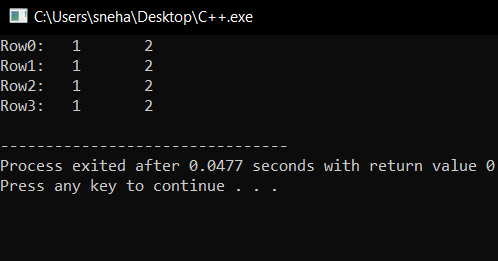
cout<<"\t"<<\*(\*p+j);

cout<<endl;

}

}

**Output**:



2D Array Pointer

Here,

* In the above code, we try to print a 2D array using pointers,
* As we earlier did, at first we initialize the 2D array, s[5][2]. And also a pointer (\*p)[2], where **p** is a pointer which stores the address of an array with 2 elements,
* As we already said, we can break down a 2D array as an array of arrays. So in this case, s is actually an array with **5** elements, which further are actually arrays with **2** elements for each row.
* We use a for loop to traverse over these 5 elements of the array, **s**. For each iteration, we assign **p** with the address of s[i],
* Further, the inner for loop prints out the individual elements of the array **s[i]** using the pointer **p**. Here, (\*p + j) gives us the address of the individual element s[i][j], so using \*(\*p+j) we can access the corresponding value.

## Passing 2-D Array to a Function

In this section, we are going to learn how to pass a **2D** array to any **function** and access the corresponding elements. In the code below, we pass the array **a**, to two functions show() and print() which prints out the passed 2D array.

#include<iostream>  
using namespace std;   
void show(int (\*q)[4], int row, int col)  
{

int i, j ;

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

{

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

cout<<"\t"<<\*(\*(q + i)+j);

cout<<"\n";

}

cout<<"\n";

}

void print(int q[][4], int row, int col)

{

int i, j;

for(i=0;i<row;i++)  
{

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

cout<<"\t"<<q[i][j];

cout<<"\n";

}

cout<<"\n";

}

int main()

{

int a[3][4] = { 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21} ;

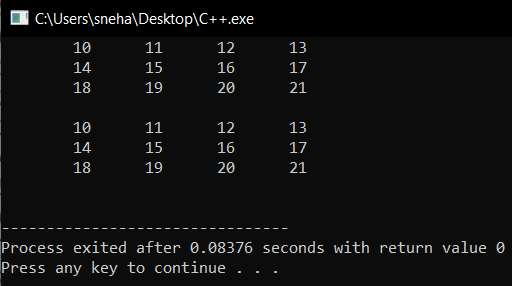
show (a, 3, 4);

print (a, 3, 4);

return 0;

}

**Output**:



Passing 2D Array To Functions

Here,

* In the show( ) function we have defined q to be a pointer to an array of **4** integers through the declaration int (\*q)[4],
* **q** holds the base address of the zeroth 1-D array
* This address is then assigned to **q**, an int pointer, and then using this pointer all elements of the zeroth 1D array are accessed.
* Next time through the loop when i takes a value **1**, the expression **q+i** fetches the address of the first 1-D array. This is because q is a pointer to the zeroth 1-D array and adding 1 to it would give us the address of the next 1-D array. This address is once again assigned to q and using it all elements of the next 1-D array are accessed
* In the second function print(), the declaration of **q** looks like this: int q[][4] ,
* This is same as **int (\*q )[4]**, where q is a pointer to an array of 4 integers. The only advantage is that we can now use the more familiar expression q[i][j] to access array elements. We could have used the same expression in show() as well but for better understanding of the use of pointers, we use pointers to access each element.

**2D dynamic memory:**

As we discuss earlier we can allocate an array dynamically using pointer. When we need multiple array we use 2D array instead of 1D array. We can also create a 2D dynamic memory.

**How can we create 2D dynamic array?**

First we need to create an array of pointers.

int \*\*ptr = new int\*[row];

Each index of array of pointers represent a row.

Now we create arrays for each row one by one.

for (int r = 0; r < row; r++)

{

ptr[r] = new int[col];

}

**Pointers and 2D dynamic Array:**

In a two dimensional array, we can access each element by using two subscripts, where first subscript represents the row number and second subscript represents the column number. The elements of 2-D array can be accessed with the help of pointer notation also. Suppose arr is a 2-D array, we can access any element arr[r][c] of the array using the pointer expression \*(\*(arr + r) + c). Now we’ll see how this expression can be derived.

**Deallocation of 2D dynamic Memory:**

We must delete the dynamic memory if there is no more use of that memory.

We follow the steps in reverse order of memory allocation.

First we delete each row than we delete the array of pointers.

**Example:**

for (int r = 0; r < row; r++)

{

delete[] ptr[r];

}

delete [] ptr;

**LAB TASKS**

**Task-1:**

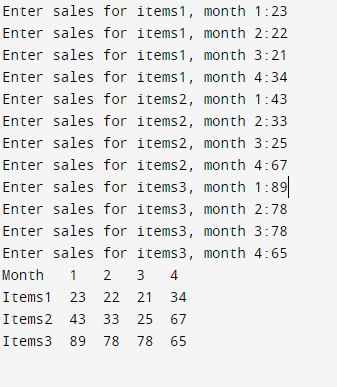
## Program to Find Largest and lowest Number in 2D Pointer Array. There should be no memory wastage.

**Task-2:**[Program to Find Largest and Second Largest Number in 2D Array.](https://t4tutorials.com/program-to-find-largest-and-second-largest-number-in-2d-array-in-cpp-c-plus-plus/)  
Note: Perform this task using Double Pointer.

**Task-3:**

Write a program to display monthly production of 3 different items for 4 months in the industry using 2D arrary.

Output:

****

**Task-4:**

Write a C++ program in which user takes integer values in a 4x6 matrix (2d Array) your task is to calculate the sum of all three rows and display the array and its sum row wise at the end of every row.