|  |  |
| --- | --- |
| **Name** |  |
| **CMS ID** |  |
| **Marks** |  |
| **Date** |  |

Multidimensional Arrays

A multidimensional array in C++ is really an array of array(s). So a two-dimensional array is an array of arrays. When you add another dimension, it becomes an array of arrays of arrays. You can see how quickly this can spiral into something very complex. So let's start with a two-dimensional array, and go from there.

Remember that the array must contain elements of the same data type! You cannot mix and match your buckets of data.

### Two-Dimensional Arrays

A two-dimensional array or 2D array is like a table with rows and columns. In this lesson, we'll be creating and working with an array that tracks three players (rows) and their five top scores in a game (columns). For simplicity, each dimension of the array will be an integer. We have 3 players (rows) and 5 scores (columns). Therefore the code to declare the array is:

int high\_scores[3][5];

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Col 0** | **Col 1** | **Col 2** | **Col 3** | **Col 4** |
|  | Row 0 | high\_scores[0][0] | high\_scores[0][1] | high\_scores[0][2] | high\_scores[0][3] | high\_scores[0][4] |
|  | Row 1 | high\_scores[1][0] | high\_scores[1][1] | high\_scores[1][2] | high\_scores[1][3] | high\_scores[1][4] |
|  | Row 3 | high\_scores[2][0] | high\_scores[2][1] | high\_scores[2][2] | high\_scores[2][3] | high\_scores[2][4] |

int high\_scores[3][5] = {

   {500, 550, 625, 700, 850},

   {385, 425, 450, 475, 495},

   {325, 330, 360, 375, 380},

};

### Elements in a Two-Dimensional Array

In order to access elements in a 2D array, you need to provide the row and column location. Just as in Excel, you would say, Row 5, Column B. In order to get to player 2's fourth high score, the code is:

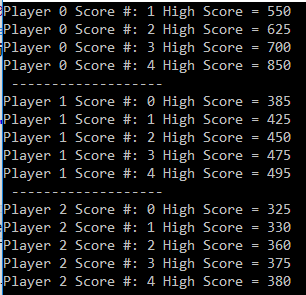
1. high\_scores[1][3];

Recall that we used a for loop to navigate a one-dimensional array. That is no different in a 2D array. Except now we have to add a nested for loop. Recall that the for loop is the counting loop; it is perfect for working with arrays that have a defined size.

This is the equivalent of walking across the table: First, you start with a row; next, you walk across all the columns; then move to the next row. Here's how it would look if you wanted to loop through and print the high scores. The code below is a fully-functioning C++ program.

1. #include <iostream>
2. using namespace std;
3. int main() {
4. // high scores for 3 players and their top scores
5. int high\_scores[3][5] = {
6. {500, 550, 625, 700, 850},
7. {385, 425, 450, 475, 495},
8. {325, 330, 360, 375, 380},
9. };
11. //start looping over rows
12. for(int i = 0; i < 3; i++) {
13. //now the rows
14. for(int j = 0; j < 5; j++) {
15. cout << "Player " << i << " Score #: " << j;
16. cout << " High Score = " << high\_scores[i][j] << endl;
17. }
18. cout << " ------------------- " << endl;
19. }
20. return 0;
21. }

The output for the code is shown in Figure 1.



Passing Array to a Function

Arrays can be passed to a function as an argument. Consider this example to pass one-dimensional array to a function:

## Example 1: Passing One-dimensional Array to a Function

**C++ Program to display marks of 5 students by passing one-dimensional array to a function.**

#include <iostream>

using namespace std;

void display(int marks[])

{

cout << "Displaying marks: "<< endl;

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

{

cout << "Student "<< i + 1 <<": "<< m[i] << endl;

}

}

int main()

{

int marks[5] = {88, 76, 90, 61, 69};

display(marks);

return 0;

}

**Output:**

When an array is passed as an argument to a function, only the name of an array is used as argument.

display(marks);

Also notice the difference while passing array as an argument rather than a variable.

void display(int m[]);

The argument marks in the above code represents the memory address of first element of array marks[5].

And the formal argument  int m[]  in function declaration converts to int\* m;. This pointer points to the same address pointed by the array marks.

That's the reason, although the function is manipulated in the user-defined function with different array name m[5], the original array marks is manipulated.

C++ handles passing an array to a function in this way to save memory and time.

Passing Multidimensional Array to a Function

Multidimensional Array can be passed in similar way as one-dimensional array. Consider this example to pass two dimensional array to a function:

### Example 2: Passing Multidimensional Array to a Function

**C++ Program to display the elements of two dimensional array by passing it to a function.**

#include <iostream>

using namespace std;

void display(int n[][2]);

int main()

{

int num[3][2] = {

{3, 4},

{9, 5},

{7, 1}

};

display(num);

return 0;

}

void display(int n[][2])

{

cout << "Displaying Values: " << endl;

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

{

for(int j = 0; j < 2; ++j)

{

cout << n[i][j] << " ";

}

}

}

**Output**

In the above program, the multi-dimensional array num is passed to the function display().

Inside, display() function, the array n (num) is traversed using a nested for loop.

The program uses 2 for loops to iterate over the elements inside a 2-dimensional array. If it were a 3-dimensional array, you should use 3 for loops.

Finally, all elements are printed onto the screen.

Note: Multidimensional array with dimension more than 2 can be passed in similar way as two dimensional array.

Q1 : Write a program to multiply the two matrices by passing a matrices to function. Program will do following tasks:

Then, it asks the user to enter the elements of two matrices and finally it multiplies two matrix and displays the result.

To perform this task three functions are made:

1. To take matrix elements from user
2. To multiply two matrix
3. To display the resultant matrix after multiplication

Q 2 : Take an array of 10 elements. Split it into middle and store the elements in two dfferent arrays. E.g.- INITIAL array :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 4 | 5 | 6 | 7 | 8 |

After Spiting

Head Array

|  |  |  |
| --- | --- | --- |
| 1 | 4 | 5 |

Tail Array

|  |  |  |
| --- | --- | --- |
| 6 | 7 | 8 |

Q 3 : Write a program C++ Program of array:

1. Make an array of integer data type and has length of 20.
2. Take value of array from user.
3. Make a function which takes array as parameters and function make following output :

Element of array that are positive

Element of array that are negative

Element of array number that are odd

Element of array number that are even

Q 4: Write a program that create two 4x5 matrices, take their values as input from user, display the one that correspond to maximum number of elements greater than 10. For example, if I have two arrays names array\_1 and array\_2, they have the following values stored in them

Since array\_1 have 9 elements that are greater than 10 whereas array\_2 have 15 elements that are greater than 10, in this case, array\_2 should be displayed. You are required to

1. Create two integer type arrays named array\_1 and array\_2 in main body
2. Pass these two arrays to a function named function\_input that takes their values from user
3. Inside the function named function\_input, you should pass these two arrays to another function named find\_maxAbove10 that calculates whether array\_1 have maximum number of values greater than 10 or array\_2 have maximum number of values greater than 10.
4. Inside the function named find\_maxAbove10, you should call another function named print\_maxAbove10 with array that corresponds to maximum number of values above 10 as actual parameter. That function should print the values of that array.