Multidimensional arrays

Matt Warner

Contents

1	Overview	3
2	Declaring 2d arrays	3
3	Initilizing a 2d array	3
4	Accessing an Element of a 2d array	3
5	Iterating over the Elements of a 2D array	3
6	Passing 2D Arrays to Functions	4

1 Overview

Arrays in C++ may have more than one dimension. Two-dimensional arrays are the most common example, but larger numbers of dimensions are possible

2 Declaring 2d arrays

When a 2d array is declared, we must specify two different dimensions for the array: the number of rows, and the number of columns in each row.

```
data-type array-name[number-of-rows][number-of-columns];
```

As with a 1d array, the number of rows and the number of columns must be specified as integers. You may use an integer literal, a symbolic constant, a variable, a function call that returns an integer, or an expression that resoles to an integer. For example:

```
int numbers[2][4];  // 2d array of two rows and
                    // four columns in each row
```

3 Initilizing a 2d array

A 2d array may be Initilized at the time it is declared.

```
int numbers[2][4] = {{1,3,5,7}, {2,4,6,8}};
```

4 Accessing an Element of a 2d array

To access an individual element of a 2d array, we must code two subscripts - one to specify the row and the other to specify the column within the row

```
cout << numbers[1][2] << endl; // Prints column 2 of row 1, the
                               // integer value 6
```

5 Iterating over the Elements of a 2D array

We usually use two nested for loops to iterate over the elements of a 2D array. For example, to find the sum of all elements of the array `numbers`, we might code the following loops:

```
int sum = 0;
for (int row = 0; row < 2; row++)
{

  for (int col = 0; col < 4; col++)
  {
    sum+= numbers[row][col];
  }
}
```

Note:-

We typically loop through the rows of the array in the outer loop and the columns of a row in the inner loop, but it's certainly possible to reverse thatpattern and loop through the columns of the array in the outer loop and then the rows of a column in the inner loop.

6 Passing 2D Arrays to Functions

Like one dimensional arrays, 2D arrays are **passed to functions by address** by default. This means that the function will be able to modify the elements of the original array in the calling routine. For example, the following program:

```
#include <iostream>

using std::cout;
using std::endl;

void print_array(const int[][4]);
void increment_array(int[][4]);

int main()
{
  int numbers[2][4] = {{1,3,5,7}, {2,4,6,8}};

  print_array(numbers);

  increment_array(numbers);

  print_array(numbers);

  return 0;
}

void print_array(const int array[][4])
{
  for (int row = 0; row < 2; row++)
  {
    for (int col = 0; col < 4; col++)
      cout << array[row][col] << ' ';

    cout << endl;
  }
  cout << endl;
}

void increment_array(int array[][4])
{
  for (int row = 0; row < 2; row++)
    for (int col = 0; col < 4; col++)
    {
      array[row][col] += 10;
    }
}
```

will produce the following output:

```
1 3 5 7
2 4 6 8

11 13 15 17
12 14 16 18
```