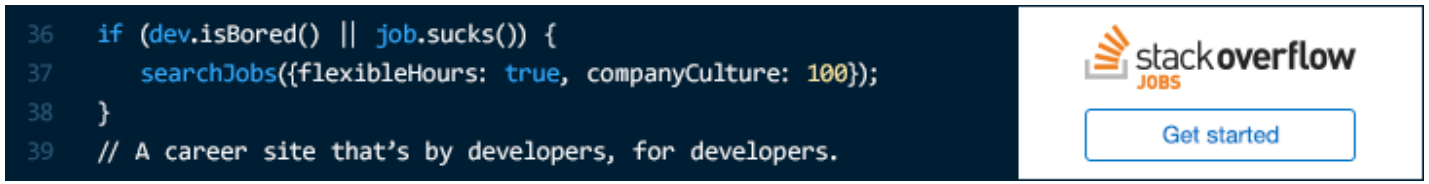
Multi-dimensional vector



Hai C++

How to create a 2D vector where in like 2D array

```
a[0][1]=98;
a[0][2]=95;
a[0][3]=99;
a[0][4]=910;

a[1][0]=98;
a[1][1]=989;
```

```
a[1][2]=981;
a[1][3]=987;
```

how do the same in vector? Thank you in advance.

c++ vector

edited Aug 9 '10 at 8:34

abatishchev
60.2k 60 223 367

asked May 5 '09 at 6:10

karthigeyan

duplicate?: stackoverflow.com/questions/741190/multi-dimensional-array-c – lothar May 5 '09 at 6:14

5 I don't think it is an *exact* duplicate. This question is about the specifics of using vectors of vectors and not raw memory. – David Rodríguez - dribeas May 5 '09 at 6:35

7 Answers

```
vector<vector<int> > a;
```

answered May 5 '09 at 6:12

Ari
1,858 1 14 23

1 How you define the rows and columns? – Rosenthal Nov 24 '14 at 4:25

Up to you to decide. – Ari Dec 22 '14 at 8:51


```
std::vector< std::vector< int > > a; // as Ari pointed
```

Using this for a growing matrix can become complex, as the system will not guarantee that all internal vectors are of the same size. Whenever you grow on the second dimension you will have to explicitly grow all vectors.

```
// grow twice in the first dimension
a.push_back( vector<int>() );
a.push_back( vector<int>() );

a[0].push_back( 5 ); // a[0].size() == 1, a[1].size()==0
```

If that is fine with you (it is not really a matrix but a vector of vectors), you should be fine. Else you will need to put extra care to keep the second dimension stable across all the vectors.

If you are planing on a fixed size matrix, then you should consider encapsulating in a class and overriding operator() instead of providing the double array syntax. Read the C++ FAQ regarding this here

edited Jul 7 '16 at 11:40

NathanOliver
50.7k 12 72 111

answered May 5 '09 at 6:27

David Rodríguez - dribeas
156k 12 194 396

1 FAQ Lite link is now here – sumodds Sep 24 '13 at 19:17

```
std::vector< std::vector<int> > a;

    //m * n is the size of the matrix

    int m = 2, n = 4;
    //Grow rows by m
    a.resize(m);
    for(int i = 0 ; i < m ; ++i)
    {
        //Grow Columns by n
        a[i].resize(n);
    }
    //Now you have matrix m*n with default values

    //you can use the Matrix, now
    a[1][0]=98;
    a[1][1]=989;
```

```
    a[1][2]=981;
    a[1][3]=987;

//OR
for(i = 0 ; i < m ; ++i)
{
    for(int j = 0 ; j < n ; ++j)
    {      //modify matrix
        int x = a[i][j];
    }

}
```

answered May 5 '09 at 6:53

aJ.
21.8k 16 67 107

If you don't *have* to use vectors, you may want to try Boost.Multi_array. Here is a link to a short example.

answered May 5 '09 at 8:39

Benoît
13.7k 5 34 61

---

dribeas' suggestion is really the way to go.

Just to give a reason why you might want to go the operator() route, consider that for instance if your data is sparse you can lay it out differently to save space internally and operator() hides that internal implementation issue from your end user giving you better encapsulation and allowing you to make space or speed improving changes to the internal layout later on without breaking your interface.

answered May 5 '09 at 7:19

Robert S. Barnes
23.9k 20 104 157

As Ari pointed, vector< vector< int>> is the right way to do it.

In addition to that, in such cases I always consider wrapping the inner vector (actually, whatever it represents) in a class, because complex STL structures tend to become clumsy and confusing.

answered May 5 '09 at 8:47

Igor Oks
13.8k 18 65 103

Just use the following method to use 2-D vector.

```
int rows, columns;

// . . .

vector < vector < int > > Matrix(rows, vector< int >(columns,0));

                                Or

vector < vector < int > > Matrix;
Matrix.assign(rows, vector < int >(columns, 0));

// Do your stuff here...
```

This will create a Matrix of size rows * columns and initializes it with zeros because we are passing a zero(0) as a second argument in the constructor i.e vector < int > (columns, 0).

answered Jul 6 '16 at 15:54

Aditya Goel
41 3
