


(6:11)

by Oliver Chang

`vector< vector<int> > M(n, vector<int>(m));`



Dynamically Allocated Matrices

- How to create and delete a dynamically allocated matrix of n by m:
 - Create
 - ```
int** M = new int*[n];
```

 // allocate an array of row pointers
    - ```
for (int i = 0; i < n; i++)
```

 // allocate the i-th row
 - ```
M[i] = new int[m];
```
  - Delete
    - ```
for (int i = 0; i < n; i++)
```

 - ```
delete[] M[i];
```

 // delete the i-th row
      - ```
delete[] M;
```

 // delete the array of row pointers
- Using STL vectors instead
 - Declare
 - ```
vector<vector<int>> M(n, vector<int>(m));
```
  - Delete: do nothing

## Two Ways to Compute Sum of Elements in a 2D Matrix

### Row sum first

```
// Calculate row sum first
int rowSum(int array[][COL]){
 int sum=0;
 for(int i=0; i<ROW; i++)
 for(int j=0; j<COL; j++) // row sum
 sum+=array[i][j];
 return sum;
}
```

Quiz: Which is faster? Why?

Remember to add compiler option for speed optimization!:

### Column sum first

```
// Calculate column sum first
int colSum(int array[][COL]){
 int sum=0;
 for(int j=0; j<COL; j++)
 for(int i=0; i<ROW; i++) // col sum
 sum+=array[i][j];
 return sum;
}
```

Code examples  
Example/twoDimArray

(12:35)

by Oliver Chang

row sum faster due to the way data stored in memory  
(continuely)(when fetch 1, data next to it is in cache also)