## 2-D arrays with variable dimensions

- Sometimes we may wish to create an array where both dimensions, i.e. rows and columns, are variable.
- To conceptualize a two-dimensional array, consider a one-dimensional array of one-dimensional arrays.
- To construct a two-dimensional array,first allocate an array of pointers to pointers. Then, index through the first array and allocate additional one-dimensional arrays.
- The following example dynamically creates a two-dimensional array of integers where user input specifies the size of each dimension. The example also initialises the array to contain random values and displays the contents of the array.

You need these include files:
<iostream.h> <stdlib.h> and <time.h>

```
int **MyArray;
int d1, d2, ndx1, ndx2;
  cout << "Enter the first array dimension: ";
  cin >> d1;
  cout << "Enter the second array dimension: ";
  cin >> d2;

  // Allocate memory for the array
  MyArray = new int*[d1];
  for (ndx1 = 0; ndx1 < d1; ndx1++)
    MyArray[ndx1] = new int[d2];

// Fill the array with random integer values between 0 and 9
  srand((unsigned)time(NULL));
  for (ndx1 = 0; ndx1 < d1; ndx1++)
    for (ndx2 = 0; ndx2 < d2; ndx2++)
      MyArray[ndx1][ndx2] = rand() % 10;
```

```
// Display the array
  for (ndx2 = 0; ndx2 < d2; ndx2++)
    cout << '\t' << "col " << ndx2;   // col #'s
  cout << endl;
  for (ndx1 = 0; ndx1 < d1; ndx1++)
    {
    cout << "row " << ndx1;          // row #'s
    for (ndx2 = 0; ndx2 < d2; ndx2++)
      {
      cout << '\t';
      cout << MyArray[ndx1][ndx2];
      }
    cout << endl;
    }
  cin.get();
  cin.get();
  // Delete the array
  for (ndx1 = 0; ndx1 < d1; ndx1++)
    delete [] MyArray[ndx1];
  delete [] MyArray;
```

**Output from Example:**

```
Enter the first array dimension: 5
Enter the second array dimension: 6
      col 0   col 1   col 2   col 3   col 4   col 5
row 0   3       8       6       8       6       1
row 1   1       4       4       5       7       0
row 2   2       1       9       6       6       9
row 3   2       7       7       7       0       1
row 4   3       0       2       6       9       7
```

## Starmap revisited.

Remember this?

```
#define map_size 5

class starmap{
int stars[map_size][map_size];
public:
        starmap(int sarray[map_size][map_size]);
        void display();
        void flip_hor();
        void flip_vert();
        };
```

```
starmap::starmap(int sarray[map_size][map_size])
{
int i,j;
for(i=0;i<map_size;i++)
        for(j=0;j<map_size;j++)
                stars[i][j]=sarray[i][j];
}
void starmap::display()
{
int i,j;
for(i=0;i<map_size;i++)
        {
        for(j=0;j<map_size;j++)
                if(stars[i][j])
                        cout << "*";
                else
                        cout << " ";
        cout << endl;
        }
cout <<endl <<endl;
}
```

```
void main()
{
int s[map_size][map_size] = { {1,1,1,1,1},
                              {1,0,0,0,0},
                              {1,1,1,1,1},
                              {1,0,0,0,0},
                              {1,0,0,0,0}};

starmap  map1(s);
map1.display();
cin.get();
}
```

Let's rewrite the starmap program, so that it dynamically allocates the memory for the starmap.

```
#include <iostream.h>

class starmap{
int **stars;
int numRows;
int numColumns;
public:
        starmap(int *sarray, int nRow, int nCol);
        void display();
};
```

```
starmap::starmap(int *sarray, int nRow, int nCol)
{
int i,j;
numRows = nRow;
numColumns = nCol;
// Allocate memory for the array
        stars = new int*[numRows];
        for (i = 0; i < numRows; i++)
                stars[i] = new int[numColumns];

for(i=0;i<numRows;i++)
        for(j=0;j<numColumns;j++)
                stars[i][j]=*(sarray++);
}
```

```
void starmap::display()
{
int i,j;
for(i=0;i<numRows;i++)
        {
        for(j=0;j<numColumns;j++)
                if(stars[i][j])
                        cout << "*";
                else
                        cout << " ";
        cout << endl;
        }
cout <<endl <<endl;
}
```

```
void main()
{
int s[5][5] = { {1,1,1,1,1},
                {1,0,0,0,0},
                {1,1,1,1,1},
                {1,0,0,0,0},
                {1,0,0,0,0}  };

starmap  map1(&s[0][0],5,5);
map1.display();

cin.get();
}
```