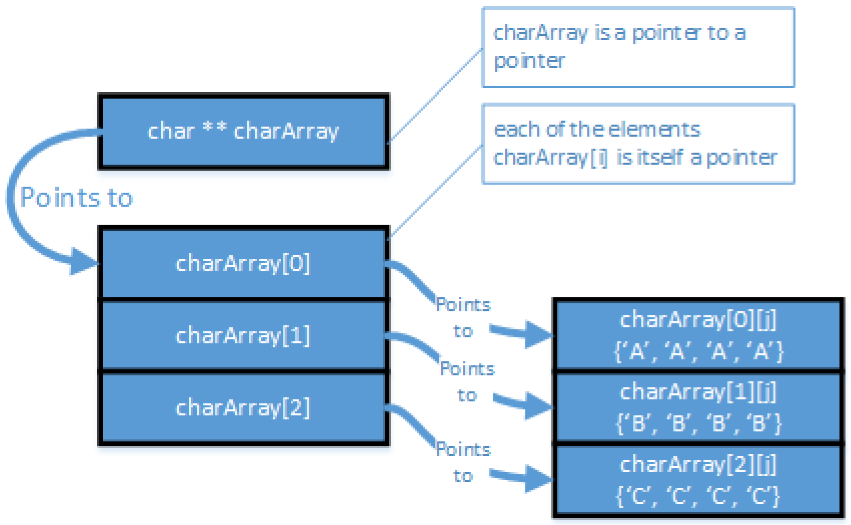
Allocating the memory for a 2-D arrayy using pointer to pointer

```
/* 2-D Dynamically allocated array of chars */

#include

using namespace std;

int main() {

int cols = 4;
int rows = 3;

// Allocate a 2-d array of ints 3 x 2
char** charArray = new char*[rows];
for(int i = 0; i < rows; ++i) {
charArray[i] = new char[cols];
}

// Fill the array
for(int i = 0; i < rows; ++i) {
for(int j = 0; j < cols; ++j) {
charArray[i][j] = char(i + 65);
}
}

// Output the array
for(int i = 0; i < rows; ++i) {
for(int j = 0; j < cols; ++j) {
cout << charArray[i][j];
}
cout << endl;
}

// Deallocate memory by deleting
for(int i = 0; i < rows; ++i) {
delete [] charArray[i];
}
delete [] charArray;
```

<u>Output</u>

```
1 AAAA
2 BBBB
3 CCCC
```

To understand this better, consider what is happening with the memory addresses:


charArray is a pointer to a pointer
each of the elements charArray[i] is itself a pointer
char ** charArray
Points to
charArray[0]
charArray[1]
charArray[2]
Points to
Points to
Points to
charArray[0][j] {'A', 'A', 'A', 'A'}
charArray[1][j] {'B', 'B', 'B', 'B'}
charArray[2][j] {'C', 'C', 'C', 'C'}