Week 7

Diana Gage

www-personal.umich.edu/~drgage

slides by Leah Bar-On Simmons and Diana Gage

Course Evals

- Thanks to those of you who filled them out!
- I got some good feedback…
  - More practice problems!
  - Do you guys want to do some 2D past exam questions today?

How's Project 3 Going?

- Any general questions I can answer?
- Specific questions?
  - Assignment 3 should be very helpful

Projects going forward

- For project 4 you can also work with a partner!
- After that, there is the Final Project, which is in groups of 4

Discussion Plan

- Any exam questions?
- 2-dimensional arrays review
  - Practice problems
- File Input/Output Intro

2 Dimensional Arrays

- You can visualize these like a matrix, or game board!
  - ALWAYS: arr[row][col]
  - *You'll often see 'col' in place of column*

Initializing all to 0

- 1-dimensional array:

  int board[3] = {0};

- **2-dimensional array:**

  int board[3][5] = {0};

- Both of these initialize every element in the array to 0
- this only works with 0 – any other number within the curly braces is interpreted as just setting the first element in the array

Initializing directly

```
const int HEIGHT = 3;
const int WIDTH = 4;

int board[HEIGHT][WIDTH] = {
                                {1, 2, 3, 4},
                                {5, 6, 7, 8},
                                {9, 10 , 11, 12}
                           };
```

- You can also initialize all to the same thing, using nested loops

When initializing…

Which of these is **invalid**?

```
int data[10][2];

int data [4][];

int data [][8];
```

When initializing…

Which of these is **invalid**?

int data[10][2];

int data [4][]; //compile error

int data [][8];

When initializing…

```
int data[10][2];
int data [4][]; //compile error
int data [][8];
```

- With a 2-dimensional array, the compiler absolutely needs to know the **column** (the second parameter) **size** at compile time
- **The row is optional**

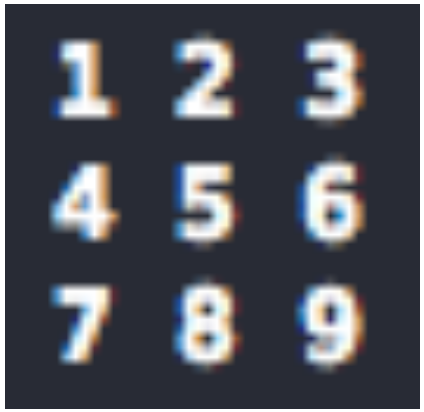
How do we iterate through a 2-d array

- What if we want to print every element in as in the picture?

How do we iterate through a 2-d array

- What if we want to print every element as in the picture. We print each element in each row, and start a new line for each row so it looks like a matrix.

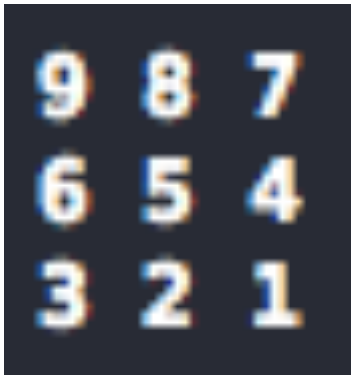
```
for (int i = 0; i < 3; ++i){
      for (int j = 0; j < 3; ++j){
          cout << my_arr[i] [j] << " ";
      }
      cout << endl;
 }
```



How do we iterate through a 2-d array

- Now what if we want to print the values **backwards,** keeping the matrix form?

```
int my_arr[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
```

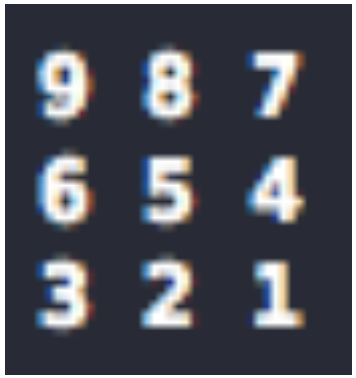


How do we iterate through a 2-d array

- Now what if we want to print the values **backwards,** keeping the matrix form?

```
int my_arr[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };

for (int i = 2; i >= 0; --i){
        for (int j = 2; j >= 0; --j){
            cout << my_arr[i][j] << " ";
        }
        cout << endl;
}
```



How do we iterate through a 2-d array

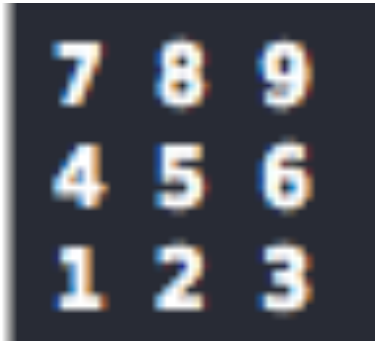
- Now what if we want to **only reverse each column**, not the entire array?
- What is this equivalent to? What will be the net result?

```
int my_arr[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
```

How do we iterate through a 2-d array

- Now what if we want to only **reverse each column**, not the entire array?
- What is this equivalent to? What will be the net result? Reversing the order of the rows

```
int my_arr[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
```

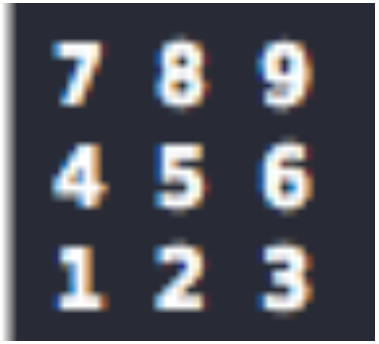


How do we iterate through a 2-d array

- Now what if we want to only **reverse each column**, not the entire array?

```
int my_arr[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };

for (int i = 2; i >= 0; --i){
    for (int j = 0; j < 3; ++j){
        cout << my_arr[i][j] << " ";
    }
    cout << endl;
}
```



File I/O

- Will be key in your next project/some of the final projects!
- Alternative to standard I/O
  - Reading in from keyboard, printing to screen
- File input – reading in from a file
- File output – writing to a file

File I/O

- **#include <fstream>** to have access to these datatypes:
  - Ifstream
  - Ofstream
- With these, you can declare variables so that you can read from/write to files!

<iostream> vs <fstream>

```
#include <iostream>
using namespace std;

int main() {
    int x;


    cin >> x;

}
```

```
#include <fstream>
using namespace std;

int main() {
    int x;
    ifstream input_file;
    input_file.open("filename");
    input_file >> x;
    input_file.close();
}
```

My suggestion: name your ifstream "**fin**"

What about writing to files?

```
#include <fstream>
using namespace std;

int main() {
    int x = 42;
    ofstream output_file;
    output_file.open("filename");
    output_file << x;
    output_file.close();
}
```

My suggestion: name your ofstream "**fout**"