CIS241

System-Level Programming and Utilities

C - 2D Arrays

Erik Fredericks, frederer@gvsu.edu

Fall 2025

> Based on material provided by Erin Carrier, Austin Ferguson, and Katherine Bowers

Expanding dimensions

We've done 1-D arrays so far

0	1	2	3

What's a 2-D array called?

- A matrix!

0	1	2	3
0	1	2	3
0	1	2	3

Declaring

We can declare a 2D array like this:

- `int mat[2][3];`

What are the dimensions?

- 2 rows, 3 cols (row-major)

Where does this memory live?

- On the stack!

It’s one contiguous block of memory

- `int mat[2][3];`
- **What does `mat[1][0]` refer to?**

0	1	2
3	4	5

- Or `mat[1][2]`?

Dynamic allocation (easy way)

There are multiple ways to *dynamically* allocate a matrix

Here, we'll walk through the easiest - treat a 2D array as a 1D array!

```
int* mat = (int*)malloc(w * h * sizeof(int));
```

How to access the c-th column of the r-th row?

- `mat[r * w + c]`

This is a cross-language way to handle multi-dimensional arrays

- Can't use `[][]` notation, though

What if I want `[ ][ ]` syntax?

Option 1: Allocate individual arrays

```c
int** mat = (int**) malloc(h * sizeof(int*));
for(int i = 0; i < h; i++){
    mat[i] = (int*) malloc(w * sizeof(int));
}
```

Can use `[][]`, but no longer contiguous in memory.

Option 2: Assign pointers into an array

Assigning pointers into an array

```
int* arr = (int*)malloc(w * h * sizeof(int));
int** mat = (int**) malloc(h * sizeof(int*));
for(int i = 0; i < h; i++){
    mat[i] = arr + i * w;
}
```

Can use `[][]` *AND* is contiguous