C Programming: 2D arrays, pass-by-value vs. pass-by-reference

Yipeng Huang

Rutgers University

February 6, 2023

Table of contents

Canvas timed quiz 2 and programming assignment 1

Quiz 2

1. Due Friday 2/10.
2. 45 minutes.
3. Two tries.
4. Pointers, arrays, passing by value and reference.
5. Reviews recent concepts that would be fair game for exams.

Progress on Programming assignment 2?

1. Due Friday 2/10.
2. Arrays, pointers, recursion, beginning data structures.

Table of contents

Lesson 6: 2D arrays

Lesson 7: Passing-by-value

Using stack and heap picture, understand how pass by value and pass by reference are different.

- C functions are entirely pass-by-value
- `swap_pass_by_values()` doesn't actually succeed in swapping two variables.

Lesson 8: Passing-by-reference

Using stack and heap picture, understand how pass by value and pass by reference are different.

- You can create the illusion of pass-by-reference by passing pointers
- `swap_pass_by_references()` does succeed in swapping two variables.

Lesson 9: Passing an array leads to passing-by-reference

Lesson 10: How the stack works; recursion example

Low addresses		Global / static data
	Heap grows downward	Dynamic memory allocation
High addresses	Stack grows upward	Local variables, parameters

Table: Memory structure

Table of contents

`matMul.c`: Function for matrix-matrix multiplication

What to pay attention to

- How `matMulProduct` result is given back to caller of function.
- How and where memory is allocated and freed.

Table of contents

struct

arrays vs structs

- Arrays group data of the same type. The `[]` operator accesses array elements.
- Structs group data of different type. The `.` operator accesses struct elements.

These are equivalent; the latter is shorthand:

```
struct element* root;
```

- `(*root).number = value;`
- `root->number = value;`