C Programming: Arrays, Functions

Yipeng Huang

Rutgers University

September 18, 2025

Table of contents

Canvas timed quiz 2 and programming assignment 1

Programming assignment 1

1. Due Friday 9/26.
2. Arrays, pointers, recursion, and beginning data structures.

Table of contents

Lesson 6: Arrays are just places in memory

- Three types of array in C: Fixed length, variable length, heap-allocated.
- name of array points to first element
- stack and heap
- `malloc()` and `free()`
- using pointers instead of arrays
- pointer arithmetic
- `char* argv[]` and `char** argv` are the same thing

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.

Why matMul() is written that way

The matMul function signature in the provided example code. Caller of matMul allocates memory.

```
1 void matMul (
2     unsigned int l,
3     unsigned int m,
4     unsigned int n,
5     int** matrix_a,
6     int** matrix_b,
7     int** matMulProduct
8 );
```

Suppose we want matMul() to be in charge of allocating memory. How to implement?

```
1 void matMul (
2     unsigned int l,
3     unsigned int m,
4     unsigned int n,
5     int** matrix_a,
6     int** matrix_b,
7     int*** matMulProduct
8 );
```

Why matMul() is written that way

The matMul function signature in the provided example code.

```
void matMul (
    unsigned int l,
    unsigned int m,
    unsigned int n,
    int** matrix_a,
    int** matrix_b,
    int** matMulProduct
);
```

A more "natural" function signature with return. How to implement?

```
int** matMul (
    unsigned int l,
    unsigned int m,
    unsigned int n,
    int** matrix_a,
    int** matrix_b
);
```