

# C Programming: Pointers recap, pass-by-value vs. pass-by-reference

Yipeng Huang

Rutgers University

February 1, 2022

# Table of contents

## Announcements

Quiz now live & programming assignment

`pointers.c`: A lab exercise for pointers, arrays, and memory

Lesson 5: Pointers are just variables that live in memory

Lesson 6: Arrays are just places in memory

Lesson 7: Passing-by-value

Lesson 8: Passing-by-reference

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

# Quiz now live & programming assignment

## Quiz now live

- ▶ Due on Thursday, 11:59 pm.
- ▶ Two tries, 45 minutes each, future quizzes may shorten to 30 minutes.
- ▶ Individual work. Open book. Experiment on iLab.

## Programming assignment

- ▶ PA0 due tonight, currently  $\approx 80\%$  submitted.
- ▶ PA1 due in one week.
- ▶ As of today, basic knowledge needed for both pieces of PA1 covered.
- ▶ Goal for today, Tuesday: Solidify our discussion about pointers.
- ▶ Goal for Thursday: More details about algorithms.

# Table of contents

## Announcements

Quiz now live & programming assignment

## `pointers.c`: A lab exercise for pointers, arrays, and memory

Lesson 5: Pointers are just variables that live in memory

Lesson 6: Arrays are just places in memory

Lesson 7: Passing-by-value

Lesson 8: Passing-by-reference

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

# Why pointers?

Pointers underlie almost every programming language feature:

- ▶ arrays
- ▶ pass-by-reference
- ▶ data structures

Vital reason why C is a low-level, high-performance, systems-oriented programming language (why we use it for this class, computer architecture).

# git pull

- ▶ From the folder `2022_0s_211`, type: `git pull`.
- ▶ By now we have several example codes: `isPrime.c`, `numList.c`, `pointers.c`, `dotProduct.c`.
- ▶ This hands-on-lab is in `pointers.c`.

## Lesson 5: Pointers are just variables that live in memory

- ▶ Pointers to pointer

## Lesson 6: Arrays are just places in memory

- ▶ name of array points to first element
- ▶ `malloc()` and `free()`
- ▶ stack and heap
- ▶ using pointers instead of arrays
- ▶ pointer arithmetic
- ▶ `char* argv[]` and `char** argv` are the same thing



## 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