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

Yipeng Huang

Rutgers University

February 2, 2023
Table of contents

Announcements
  Final exam date and time now confirmed
  Canvas timed quiz 1 and programming assignment 1

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
  Lesson 10: How the stack works; recursion example
Final exam date and time now confirmed

Final exam to take place at registrar assigned time

1. Thursday, May 4, 8 am - 11 am.
2. First day of exam week.
3. This classroom, Hill 114.
4. The final exam schedule can be confirmed here: https://scheduling.rutgers.edu/scheduling/exam-scheduling/ final-exam-schedule/current-final-exam-exam-schedule
Canvas timed quiz 1 and programming assignment 1

Progress on Quiz 1?
1. Due tomorrow Friday 2/3.
2. 45 minutes.
3. Two tries.
4. Linux, some C.
5. Reviews recent concepts that would be fair game for exams.

Progress on Programming assignment 1?
1. Due Friday 2/10.
2. Arrays, pointers, recursion, beginning data structures.
Announcements
   Final exam date and time now confirmed
   Canvas timed quiz 1 and programming assignment 1

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
   Lesson 10: How the stack works; recursion example
From the folder 2023_0s_211, type: `git pull`
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).
Lesson 5: Pointers are just variables that live in memory

- Pointers to pointer
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 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

<table>
<thead>
<tr>
<th>Low addresses</th>
<th>Global / static data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heap grows downward</td>
<td>Dynamic memory allocation</td>
</tr>
<tr>
<td>High addresses</td>
<td>Stack grows upward</td>
</tr>
</tbody>
</table>

**Table:** Memory structure