Pointers and Bubbles (A2)

1/20/2023

89/100 Points

Attempt 1

Review Feedback
1/20/2023

Attempt 1 Score:
89/100

View Feedback

Anonymous Grading: **no**

Unlimited Attempts Allowed

Details

Overview

C-style pointers are important to understand, because they live on in higher level languages. This lab will help you come to grip with pointers and how useful they can become.

The *bubble sort* is an easy-to-implement algorithm for sorting a group of elements. Here's the basic algorithm:

```
for i in list - 1
  for j in list - 1
    if list[j] > list[j + 1]
      swap list[j] and list[j+1]
```

Essentially, it looks through a list and checks each element against its neighbor. If an element is larger than the next element, the two elements should be swapped. One pass through the list is not sufficient to sort the entire list. The safest approach is to run the test in a pair of nested loops. (One could argue that this is not absolutely necessary or efficient, but that will be the subject of another lab.)

The interesting part of this process from our current perspective is the swap() function. We need to give it two variables, and it needs to change the *position* of the values in those variables.

When a function changes a single variable, we often simply have the function return a value and then we re-assign that value to the variable in question, but functions can typically only return a single value. How do you handle a case like swap where you want to change the values of two variables at once?

The solution is to pass pointers to the variables in question.

Your Task

Implement a simple bubble sort algorithm. Please begin with the following code snippet:

```
//bubble.c
//famous bubble sort
//implement the swap algorithm with pointers

#include <stdio.h>
#define MAX 9

//function prototypes
void printValues();
void sort();
void swap(int*, int*);

int values[] = {7, 3, 9, 4, 6, 1, 2, 8, 5};

int main(){
  printf("Before: \n");
  printValues();
  sort();
  printf("After: \n");
  printValues();

  return(0);
} // end main
```

Please note the following things about the code snippet:

- I've provided all the function prototypes you need. You will not need to change these prototypes (nor should you.)
- The main() function is complete. There is no need to modify the main() function.
- You will need to build the other necessary functions
- The prototypes should give you a good hint what the functions should be and how they should be written.
- The *sort()* function should follow the algorithm described above
- The *swap()* function **must use pointers.** There are ways to make this program work without pointers, but the purpose of this project is to have a real-life example of pointer work.
- See the output below for an indication how the *printValues()* function should be written.
- This program should be written in C, **NOT C++**. I want to be certain you can get around the C language if you need to.
- Please use pointers, not references or other tricks. That's what this project is about.

Output

The output of the program should look something like this:

```
  Before:
[7 3 9 4 6 1 2 8 5 ]
[3 7 9 4 6 1 2 8 5 ]
[3 7 4 9 6 1 2 8 5 ]
[3 7 4 6 9 1 2 8 5 ]
[3 7 4 6 1 9 2 8 5 ]
[3 7 4 6 1 2 9 8 5 ]
[3 7 4 6 1 2 8 9 5 ]
[3 7 4 6 1 2 8 5 9 ]
[3 4 7 6 1 2 8 5 9 ]
[3 4 6 7 1 2 8 5 9 ]
[3 4 6 1 7 2 8 5 9 ]
[3 4 6 1 2 7 8 5 9 ]
[3 4 6 1 2 7 5 8 9 ]
[3 4 1 6 2 7 5 8 9 ]
[3 4 1 2 6 7 5 8 9 ]
[3 4 1 2 6 5 7 8 9 ]
[3 1 4 2 6 5 7 8 9 ]
[3 1 2 4 6 5 7 8 9 ]
[3 1 2 4 5 6 7 8 9 ]
[1 3 2 4 5 6 7 8 9 ]
[1 2 3 4 5 6 7 8 9 ]
After:
[1 2 3 4 5 6 7 8 9 ]
```

Note that I printed the current status of the array after every swap. Also note there is a single array; The actual values are moving around in the original data structure.

Algorithm

As a part of every project this semester, you will be required to include some sort of algorithm file. The exact details for this will change as you learn new materials and techniques. For this week, we will require a simple text file (must be named *algorithm.txt or algorithm.md*) with the following concepts:

- Goals - What is the goal of the program or function?
- Input - What (if any) input does the function or program accept? Are there specific data types or ranges?
- Output - What (if any) output does the function or program return
- Steps - What are the steps (written in English) necessary to solve this problem. This should NOT be programming code, but each line should be specific enough to be translated to a line or two of code in your target language.

Please do this process for the main program (calling functions as necessary as algorithmic steps) and once for each function you anticipate in your project.

The algorithm should be completed BEFORE writing any code.

Turning in the project

- This assignment will be turned in through IU GitHub. Please name your repo CSCI24000_spring23_A2
- After you've submitted your assignment in Github, come back to this assignment page and submit the full Github URL (**https://github.iu.edu/username/reponame** (https://github.iu.edu/username/reponame)) for your repo here.
- If your repo has another name, it will not be graded.
- Your repo must be private. We will not grade projects in public repos, and we will require you to take down any homework assignments in a public repo.
- We will expect your base assignment to be in a folder called base and any blackbelt extension to be placed in a folder called blackbelt (if applicable), with instructions and a description of what you did.

Please ensure to add all the following are listed as collaborators:

- **Git Collaborators** (https://iu.instructure.com/courses/2131288/pages/assignments-submission-and-grading)

Black belt Options

There are many opportunities to extend this program. Here are a few:

- Make a version using references in C++
- Investigate some other sort algorithms (insertion sort, merge sort)
- Create a version of the program that automatically generates a random list of integers and then sorts them.

---

https://github.iu.edu/parmsing/CSCI24000_spring23_A2

(https://iu.instructure.com/courses/2131288/modules/items/28790575) (https://iu.instructure.com/courses/2131288/modules

You are unable to submit to this assignment as your enrollment in this course has been concluded.