LAB 4: 2D array, Structures, Pointer basics

Released: Oct. 21 Due: Oct. 29(Tuesday) 11:59 PM Total mark: 100 pts

Problem A: 2D array (20 pts)

Subject

Similarities and differences between 2D char array and array of char pointers, both of which can be used to store rows of input strings. (The array of char pointers will cover later)

Specification

Write an ANSI-C program that uses 2D array to read and store user input strings line by line, until a line of xxx is entered. The program then reorders the rows of inputs.

Implementation

Download the file **lab4A.c** to start off. Assume that there are at least 4 lines of inputs (excluding the terminator line xxx) and there are no more than 30 lines of inputs. Also, assume that each line contains no more than MAXCOLS characters. Note: each line of inputmay contain spaces.

- Use a table-like **2D array** to store the user inputs. That is, define char inputs [MAXLINES][MAXCOLS].
- First print the memory allocation for the 2D array using sizeof.You should get 1500 (bytes), as shown in the sample output.
- Use fgets to read in a line into the table row directly. Note that a trailing \n is read in.
- When all the inputs have been read in (indicated by input line xxx), exchange data in row 0 and row 1 in main (), and then send the array to a function exchange2D() to exchange the data in the other rows, as shown below. Assume the inputs contain at least 4 lines.
- Define a function **void exchange2D(char[][MAXCOLS], int n)**which takes as argument an 2D array, and swaps the data in the first n adjacent rows, except the first two rows, i.e., starting from 3rd row. Specifically, swaps the 3rd row with the 4th row, swaps the 5th row with the 6th row and so on. If n is an odd number, then the last row is not swapped.
- Define a function **void print2D(char[][MAXCOLS], int n)**which takes as argument a 2D array, and then prints the first n rows of the array on stdout.
- Use this function in main to display all the stored rows of the array, **both before and after swapping and sorting.**

Notice that for the 2D array, similar to general 1D array, when passing to a function, we also need an extra argument nto indicate the function where to stop the processing, as there is no "terminator row" in the 2D array.

LAB 4: 2D array, Structures, Pointer basics

Released: Oct. 21 Due: Oct. 29(Tuesday) 11:59 PM Total mark: 100 pts

Sample Inputs/Outputs:

```
sizeof inputs: 1836

Enter string: Giraffes 0
Enter string: Zebras 1
Enter string: Monkeys 2
Enter string: Kangaroos 3
Enter string: Do You like them? 4
Enter string: Yes 5
Enter string: Thank You 6
Enter string: Bye 7
Enter string: xxx

After while loop Count: 8
[0]: Giraffes 0
[1]: Zebras 1
[2]: Monkeys 2
[3]: Kangaroos 3
[4]: Do You like them? 4
[5]: Yes 5
[6]: Thank You 6
[7]: Bye 7

== After swapping ==
[0]: Zebras 1
[1]: Giraffes 0
[2]: Kangaroos 3
[3]: Monkeys 2
[4]: Yes 5
[5]: Do You like them? 4
[6]: Bye 7
[7]: Thank You 6
```

a.exe < inputB.txt
sizeof inputs: 1836

Enter string: Enter string: Enter string: Enter string: Enter string: Enter string: Enter string: Enter string: Enter string: Enter string: Enter string: Enter string:
After while loop Count: 11
[0]: giraffes are high 0
[1]: mosquitos are annoying 1
[2]: monkeys are smart 2
[3]: kangaroos are funny 3
[4]: dogs are friendly 4
[5]: hippos are huge 5
[6]: cobras are fearsome 6
[7]: foxes 7
[8]: elephants 8
[9]: hens 9

[10]: bisons 10

== After swapping ==
[0]: mosquitos are annoying 1
[1]: giraffes are high 0
[2]: kangaroos are funny 3
[3]: monkeys are smart 2
[4]: hippos are huge 5
[5]: dogs are friendly 4
[6]: foxes 7
[7]: cobras are fearsome 6
[8]: hens 9
[9]: elephants 8
[10]: bisons 10

After you submitted, as an additional practice, change the formal argument in one of the function definitions (and the corresponding declaration) from char[][MAXCOLs] to char[][], for example, void exchange2D(char[][]), and compile. What do you get?

Problem B – Structures (20 pts)

Subject: Structure and functions, array of structures, pointer and malloc for structures.

Implementation

Download file **lab4B.c**. Complete the program, and observe,

- how a function can be declared to return a structure. By returning a structure, the function can return more than one value by encapsulating multiple values into a struct.
- Implement function **struct results getSumDiff(int, int),** which calculates and returns the sum and difference of the two argument integers, as struct results.
- Implement function **void printStruct(struct results s)** that takes a structure results, and print the members.

Sample Inputs/Outputs:

```
Struct a before processStruc function: 100 4
Struct a in the processStruc function: 101 104
Struct a after  processStruc function: 100 4

Enter two integers: 45 67
Sum is: 112, Diff is -22

Enter two integers: 23 12
Sum is: 35, Diff is 11
```

LAB 4: 2D array, Structures, Pointer basics

Released: Oct. 21 Due: Oct. 29(Tuesday) 11:59 PM Total mark: 100 pts

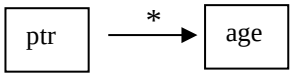
Problem C Pointer 101 (20 pts)

Specification

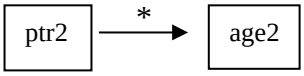
Write your first (short) program that uses pointers. Name it as **lab4C.c**

Implementation

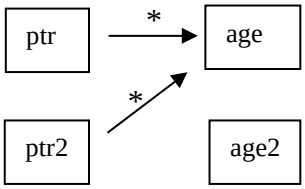
- define an integer age and initialize it to 10. Define another integer age2, which is initialized to 100;
- define an integer **pointer** variable ptr, and make it point to age



- display the value of age, both via age (direct access), and **via pointer ptr**(indirect access).
- **use ptr** to change the value of age to 14;
- confirm by displaying the value of age, both via age and via its pointer **ptr**



- define another pointer variable ptr2, and make it point to age2
- assign triple of age's value to age2 via **pointer ptr and ptr2** (i.e., without referring to age and age2). age2 is 42 now.
- display the value of age2, both via age2, and via its pointer **ptr2**
- now let ptr2 point to age(too) by getting the address of age from pointer variable ptr (i.,e., without using &age)



- confirm by displaying the value of ptr2's pointee via ptr2
- display value of age, both from age, and via ptr and ptr2.
- use ptr2 to decrease the value of age by 1. age is 13 now.
- display value of age, both from age, and via ptr and ptr2.
- finally, display the address of age, using printf("%p %p %p\n",&age,ptr,ptr2);
- Notice that here we print ~~*~~ptr and ~~*~~ptr2 directly. This displays the content of the pointer variables, which is the address of age (in Hex).

Sample Inputs/Outputs:

```
age: 10 10
age: 14 14
age: 42 42
ptr2's current pointee: 14
age: 14 14 14
age: 13 13 13
ptr2's current pointee: 13
0061FF14 0061FF14 0061FF14
```

You will get different numbers here but they should be identical to each other. This is the memory address of variable `age`, in Hex.

LAB 4: 2D array, Structures, Pointer basics

Released: Oct. 21 Due: Oct. 29(Tuesday) 11:59 PM Total mark: 100 pts

Pointers and passing address of scalar variables

Problem C1:

Subject

Experiencing "modifying scalar arguments by passing addresses/pointers".

Specification

Write an ANSI-C program that reads three integers line by line and modify the input values.

Implementation Download file **lab4C1.c** to start off.

- The program reads user inputs from stdin line by line. Each line of input contains 3 integers separated by blanks. A line that has the first number being -1 indicates the end of input.
- Store the 3 input integers into variable a, b and c;
- Function **swapIncres()** is called in main() with an aim to change the values of a, b and c in such a way that, after function swapIncres returns, b's value is doubled, a stores c's original value incremented by 100, and c stores the original value of a. As an example, suppose a is 1, b is 2 and c is 3, then after function returns, a has value 103, b has value 4 and c has value 1.
  - Compile and run the program and observe unsurprisingly that the values of a, b and c are not changed at all (why?).

```
 Enter three values seperated by a blank:  1 2 3
Original inputs:   a:1     b:2     c:3
Rearranged inputs: a:1     b:2     c:3

 Enter three values seperated by a blank: 4 5 6
Original inputs:   a:4     b:5     c:6
Rearranged inputs: a:4     b:5     c:6

 Enter three values seperated by a blank: -1 2 3
```

- Modify the program so that it works correctly, as shown in the sample inputs/outputs below. **You should only modify function swapIncres and the statement in main that calls this function. No global variables should be used.**

Sample Inputs/Outputs:

```
 Enter three values seperated by a blank: 4 8 9
Original inputs:   a:4     b:8     c:9
Rearranged inputs: a:109   b:16    c:4

 Enter three values seperated by a blank: 5 12 7
Original inputs:   a:5     b:12    c:7
Rearranged inputs: a:107   b:24    c:5

 Enter three values seperated by a blank: 12 20 3
Original inputs:   a:12    b:20    c:3
Rearranged inputs: a:103   b:40    c:12

 Enter three values seperated by a blank: -1 2 3
```

% **cat inputA.txt**
3 5 6
2 67 -1
-12 45 66
66 55 1404
22 3 412
-2 44 6
-1 55 605

% a.exe < inputA.txt
Enter three values separated by a blank:
Original inputs: a:3 b:5 c:6
Rearranged inputs: a:106 b:10 c:3

Enter three values separated by a blank:
Original inputs: a:2 b:67 c:-1
Rearranged inputs: a:99 b:134 c:2

Enter three values separated by a blank:
Original inputs: a:-12 b:45 c:66
Rearranged inputs: a:166 b:90 c:-12

Enter three values separated by a blank:
Original inputs: a:66 b:55 c:1404
Rearranged inputs: a:1504 b:110 c:66

Enter three values separated by a blank:
Original inputs: a:22 b:3 c:412
Rearranged inputs: a:512 b:6 c:22

Enter three values separated by a blank:
Original inputs: a:-2 b:44 c:6
Rearranged inputs: a:106 b:88 c:-2

Enter three values separated by a blank:

No submission for Problem C1

Problem C2: (20 pt)

Modify program **lab4C1.c,** by defining a new function **void swap(int *, int *)** which swaps the values of a and c. This function should be called in function **void swapIncres(int*, int*, int*).** Specifically, swapIncres()only increases the value of parameters, and delegates the swapping task to swap().

LAB 4: 2D array, Structures, Pointer basics

Released: Oct. 21 Due: Oct. 29(Tuesday) 11:59 PM Total mark: 100 pts

You should use the same code of main, and the parameter list of swapIncres that you did in problem C1.

Again, no global variables should be used. Name the new program **lab4C2.c**

Sample Inputs/Outputs: Same as Problem C1

```
 Enter three values seperated by a blank: 4 8 9
Original inputs:   a:4     b:8     c:9
Rearranged inputs: a:109   b:16    c:4

 Enter three values seperated by a blank: 5 12 7
Original inputs:   a:5     b:12    c:7
Rearranged inputs: a:107   b:24    c:5

 Enter three values seperated by a blank: 12 20 3
Original inputs:   a:12    b:20    c:3
Rearranged inputs: a:103   b:40    c:12

 Enter three values seperated by a blank: -1 2 3
```

Problem C3: (20pt)

Modify the above program, by changing the prototype of function swap to be **void swap(int **, int **)** which swaps the values of a and c. This function should be called in function **void swapIncres(int*, int*, int*)** .Specifically, swapIncres()only increases the value of parameters, and delegates the swapping task to swap(). **Hint:** You would be calling swap(&x, &z) in swapIncres() function. And will be using double pointers in swap() for all pointer variables e.g int temp = **x;

You should use the same code as main, and the parameter list of swapIncres that you did in problem C2. Again, no global variables should be used. Name the new program **lab4C3.c**

Sample Inputs/Outputs: Same as above.

```
 Enter three values seperated by a blank: 4 8 9
Original inputs:   a:4     b:8     c:9
Rearranged inputs: a:109   b:16    c:4

 Enter three values seperated by a blank: 5 12 7
Original inputs:   a:5     b:12    c:7
Rearranged inputs: a:107   b:24    c:5

 Enter three values seperated by a blank: 12 20 3
Original inputs:   a:12    b:20    c:3
Rearranged inputs: a:103   b:40    c:12

 Enter three values seperated by a blank: -1 2 3
```

LAB 4: 2D array, Structures, Pointer basics

Released: Oct. 21 Due: Oct. 29(Tuesday) 11:59 PM Total mark: 100 pts

---

SUBMISSION INSTRUCTIONS: In summary, for this lab you should submit the following FIVE files in a ZIPPED folder titled Lab04:
Lab4A.c, lab4B.c, lab4C.c ,lab4C2.c, lab4C3.c

Common Notes
All submitted files should contain the following header:

```
/**************************************
* Fall24 – Lab04 *
*  Author: Last name, first name *
*  EECS/Prism username: Your prism login username *
*  Yorku Student #: Your student number *
*  Email: Your email address *
**************************************/
```

Other common notes:

- **Make sure your program compiles in the lab environment. A program that does not compile, or crashes with "segmentation fault" in the lab environment will get 0.**
- **Note that labs are individual work. You can discuss with others but should not copy code from others, or from the web. Doing so is considered a violation of academic honesty.**
- **Note that if you have taken this course before, you should do the lab again. Submitting the previous term's file – even it is yours -- is considered self- plagiarism and will receive 0.**
- **All submissions need to be done in eClass in a zipped folder.**
  - **Also note that you can submit the multiple zip folders, the latest one will overwrite the old one.**
  - **Please submit the right folder. Wrong/incorrect submissions will result in zero marks.**

****