BBM203: Pointer Exercises

Fall 2024

Instructor: Anar Mammadov

Submission Deadline: Whenever you want!

Introduction

In this non-graded assignment, you will work with pointers and dynamic memory allocation in C++. The goal of this assignment is to help you understand how pointers are used to manipulate data and manage memory. You will be required to implement a series of functions, each designed to reinforce key concepts related to pointers, arrays, and dynamic memory management.

Each function should be implemented in a separate `.cpp` file. Once you've completed the functions, you will need to test them thoroughly to ensure they work as expected, especially for edge cases like empty arrays or swapping the same row in a 2D array. Finally, make sure your implementation is memory-efficient and handles dynamic memory allocation and deallocation correctly.

Task 1: Swap Values

Function Signature:

```
void swapValues(int* a, int* b);
```

Problem Description:
You are tasked with writing a function that swaps the values of two integers using pointers.

Instructions:
- You will be given two integer pointers, and your task is to swap the values stored at the memory addresses these pointers point to. - Ensure that the function modifies the values in place and doesn't return anything.

Things to Consider:
- How can you temporarily store one value to perform the swap? - How does dereferencing a pointer work to access the value it points to?

Task 2: Reverse Array

Function Signature:

```
void reverseArray(int* arr, int size);
```

Problem Description:
This function should reverse the contents of an array in place. The array will be provided as a pointer, along with its size.

Instructions:
- You must reverse the array in place. That is, the array itself should be modified, and no new array should be created. - The function takes an integer pointer (array) and its size.

Things to Consider:
- How can you use two pointers or indexes (one at the start and one at the end) to swap elements until the middle of the array is reached? - What happens if the array is empty or has only one element?

Task 3: Find Min/Max in Array

Function Signature:

```
void findMinMax(int* arr, int size, int* min, int* max);
```

Problem Description:
This function will find the minimum and maximum values in an array. The results should be returned via the ‘min‘ and ‘max‘ pointers.

Instructions:
- The function will receive an array, its size, and two pointers (‘min‘ and ‘max‘) where the minimum and maximum values should be stored. - You need to iterate through the array and update the values of ‘*min‘ and ‘*max‘ based on the array elements.

Things to Consider:
- What is the initial value of ‘*min‘ and ‘*max‘ when you start scanning the array? - How do you update the values at the ‘min‘ and ‘max‘ pointers as you iterate through the array?

Task 4: Dynamic 2D Array Creation/Deletion

Function Signatures:

```
int** create2DArray(int rows, int cols);
void delete2DArray(int** arr, int rows);
```

Problem Description:
The first function, ‘create2DArray‘, should dynamically allocate memory for a 2D array (a matrix) based on the provided number of rows and columns. The second function, ‘delete2DArray‘, should properly deallocate the memory for the 2D array.

Instructions:
- ‘create2DArray‘: This function takes the number of rows and columns as input, allocates memory for a 2D array, and returns a pointer to the newly created array. Fill the array as arr[i][j] = i + j. E.g. an array with 2 rows and 3 columns would look like this:
0 1 2
1 2 3
- ‘delete2DArray‘: This function takes the array and the number of rows as input and frees the memory allocated for the 2D array.

Things to Consider:
- How do you allocate memory dynamically for both the rows and columns of the array? - What happens if you forget to free the memory allocated for each row and the overall array?

Task 5: Swap Rows in a 2D Array

Function Signature:

```
void swapRows(int** arr, int row1, int row2, int cols);
```

Problem Description:
This function swaps two rows in a dynamically allocated 2D array. You are provided the array, the indices of the rows to be swapped, and the number of columns in each row.

Instructions:
- You will receive a pointer to the array, two row indices (‘row1‘ and ‘row2‘), and the number of columns in the array. - The task is to swap the contents of ‘row1‘ and ‘row2‘.

Things to Consider:
- How can you swap the values of two rows efficiently without creating new arrays? - What happens if ‘row1‘ and ‘row2‘ are the same?

How to Build and Run Your Program

To compile and run your program, follow these instructions:

Step 1: Build the Program

You will use the `g++` compiler to compile all of your `.cpp` files into a single executable. Open a terminal in the directory containing your `.cpp` files and run the following command:
$ g++ main.cpp swapValues.cpp reverseArray.cpp minMaxArray.cpp dynamic2DArray.cpp swapRows.cpp -o test_program

Step 2: Run the Program

After successfully building the program, you can run the executable by typing the following command in the terminal:

$ ./test_program

This will execute your program, and it will test all the functions you have implemented (e.g., `swapValues`, `reverseArray`, `minMaxArray`, etc.).

Step 3: Example Output

If your functions are correctly implemented, you should see an output similar to the following:

```
Reversed array: 5 4 3 2 1
Min: 1, Max: 5
Dynamic 2D array:
0 1 2 3
1 2 3 4
2 3 4 5
2D array before swapping rows:
0 1 2 3
1 2 3 4
2 3 4 5
2D array after swapping row 0 and row 2:
2 3 4 5
1 2 3 4
0 1 2 3
swapValues result: a = 20, b = 10
```

Submission Instructions

- Implement each function in a separate `.cpp` file.
- Submit your code as `.cpp` files through the TurboGrader.
- Ensure that your code passes all the provided test cases. Before submitting your work, ensure that your directory is organized as shown below:

  ```
  b<studentID>.zip/

   swapValues.cpp
   swapValues.h
   reverseArray.cpp
   reverseArray.h
   minMaxArray.cpp
   minMaxArray.h
   dynamic2DArray.cpp
   dynamic2DArray.h
   swapRows.cpp
   swapRows.h
  ```

- You MUST use the starter code. All classes should be placed directly in your zip archive.

- This file hierarchy must be zipped before submitted (not .rar, only .zip files are supported).
- Do not submit any object or executable files. Only header and C++ files are allowed.

Grading Criteria

Keep in mind that this assignment does not have any influence in your grade, and is only to help you to understand the topic thoroughly.

- **Correctness (90%)**: Does the code correctly implement the required functionality?
- **Memory Management (10%)**: Does the code handle dynamic memory allocation and deallocation correctly?