16.216: ECE Application Programming

Fall 2011

Programming Assignment #8: 2D Arrays and Strings

Due **Friday, December 9**, 11:59:59 PM

1. Introduction

This assignment gives you some basic experience dealing with two-dimensional arrays and strings. You will implement a program that uses different commands to manipulate a two-dimensional array of user-specified size and print the results to the screen.

2. Deliverables

Submit your source file using our course site in Blackboard; you should be able to access the site through https://continuinged.uml.edu/login/login.cfm.

Ensure your source file name is ***prog8_2Darrays.c***. <u>You should submit only the .c file.</u> Failure to meet this specification will reduce your grade, as described in the program grading guidelines.

3. Specifications

Input: Your program should first prompt the user to input the dimensions of a two-dimensional array, which should be initialized to 0. The maximum array size is 20x20, so the prompt should be repeated as long as the user enters invalid dimensions.

Your program should then repeatedly prompt the user to enter one of the following commands:

- `set`: Set array element `[i][j]` to a particular value
- `add`: Add a value to the current value in array element `[i][j]`
- `subtract`: Subtract a value from the current value in array element `[i][j]`
    - After any of the above three commands, your program should prompt the user to enter a row number, column number, and value.
    - If the row number and column number are valid (i.e., in the array), perform the specified operation with the value entered.
    - If either the row or column (or both) is invalid, print an error message and repeat the prompt.
- `clear`: Reset all values in the array to 0
- `exit`: Exit the program.

Output: Any time a command changes the array, all array values should be printed. You should print the array in such a way that all values line up if possible. Assume each array value will use no more than 5 characters.

See Section 5: Test Cases for sample inputs and outputs.

Error checking: Your program should print an error and repeat a prompt in the following cases:

- If the user initially enters invalid array dimensions.
- If the user enters an invalid row or column number for any command.
- If the user enters an invalid command at any point.

4. Formulating a solution

Initializing arrays: As mentioned briefly in class, providing just one initial value for an array will cause all other array values to be set to zero. You can therefore initialize an entire array to 0 simply by writing a statement similar to:

```
arr[3][4] = {0};
```

Array dimensions: Note that, in order to ensure all possible arrays work, your array should be declared with the maximum possible dimensions. You can then keep separate variables to track the actual number of rows and columns in your array—these values will be set by your initial user input, and will be used as the array boundaries for the rest of the program.

Reading strings: This point corrects an error made in class—if you use a character array to store string input, you only need to pass the array name to `scanf()` when reading the string. Do not use the address operator &:

```
char command[10];
scanf("%s", command);          // RIGHT
scanf("%s", &command);         // WRONG–don't use &
```

Functions: Most commands require at least one of these two actions:

- Prompt the user to enter a row number, column number, and input value, read those inputs, and print an error (and repeat the prompt) if the row or column number is invalid.
- Print the entire two-dimensional array.

I therefore suggest writing the following functions for use in your main program:

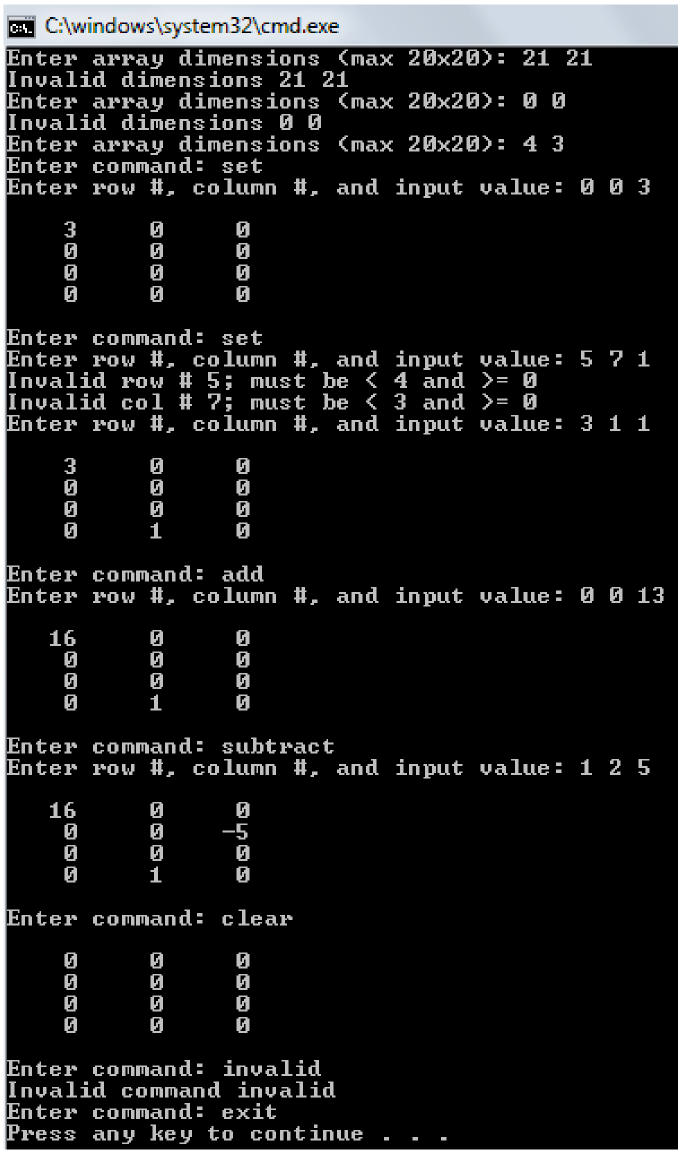
-
  ```
  void readVals(int *rowNum, int *colNum, int *val,
                int maxR, int maxC);
  ```
    - Reads row number, column number, and input value, which have been passed by address. If the row or column number falls outside the array, print an error, repeat the prompt, and re-read the input.

Functions (cont.)

- `void printMatrix(int m[MAX_ROWS][MAX_COLS], int r,`
  `                  int c);`
    - Print the entire matrix, using appropriate formatting.
    - Note that this function definition assumes you have defined symbolic constants for the maximum number of rows and columns.

5. Test Cases

Your output should match these test cases exactly for the given input values. I will use these test cases in grading of your lab, but will also generate additional cases that will not be publicly available. Note that these test cases may not cover all possible program outcomes. You should create your own tests to help debug your code and ensure proper operation for all possible inputs.


C:\windows\system32\cmd.exe

```
Enter array dimensions (max 20x20): 21 21
Invalid dimensions 21 21
Enter array dimensions (max 20x20): 0 0
Invalid dimensions 0 0
Enter array dimensions (max 20x20): 4 3
Enter command: set
Enter row #, column #, and input value: 0 0 3

    3      0      0
    0      0      0
    0      0      0
    0      0      0

Enter command: set
Enter row #, column #, and input value: 5 7 1
Invalid row # 5; must be < 4 and >= 0
Invalid col # 7; must be < 3 and >= 0
Enter row #, column #, and input value: 3 1 1

    3      0      0
    0      0      0
    0      0      0
    0      1      0

Enter command: add
Enter row #, column #, and input value: 0 0 13

   16      0      0
    0      0      0
    0      0      0
    0      1      0

Enter command: subtract
Enter row #, column #, and input value: 1 2 5

   16      0      0
    0      0     -5
    0      0      0
    0      1      0

Enter command: clear

    0      0      0
    0      0      0
    0      0      0
    0      0      0

Enter command: invalid
Invalid command invalid
Enter command: exit
Press any key to continue . . .
```