project[4]: Functions

Due Tuesday, 10/27/2015, 11:59:59pm

Project Goals

The goals of this project are to explore the use of functions.

Project Description

This project consists of two programs:

Program 1 (Counting digits):

Write a program that asks the user to enter an NxM-dimensional array containing only the digits between 0 and 9 and counts the number of times each one of the 10 digits appears in the array. This program should:

- prompt a user to enter the sizes (rows and columns) of the array, and read in the dimensions
- prompt the user to enter the array row-by-row and read in each row and verify that the user has entered values between 0 and 9. If values outside of this range are entered, the program should prompt the user to enter the row again
- display the total number of times each digit appears in the array

Here is an example of exactly how the program should behave (items underlined are entered by the user):

```
This program counts occurrences of digits 0 through 9 in an NxM array
Enter the size of your array: 2 6
Enter row 0: 0 1 2 3 4 5
Enter row 1: 0 1 6 72 81 9
Values outside of range.
```

```
Enter row 1: 0 1 6 7 8 9

Total count for each digit:
Digit 0 occurs 2 times
Digit 1 occurs 2 times
Digit 2 occurs 1 time
Digit 3 occurs 1 time
Digit 4 occurs 1 time
Digit 5 occurs 1 time
Digit 6 occurs 1 time
Digit 7 occurs 1 time
Digit 8 occurs 1 time
Digit 9 occurs 1 time
```

Requirements: your program should use the following user defined functions:

- a function `read_row` to read in each row (including the prompt and retrieving the values). This may need to take an array (or part of the array) as a parameter.
- a function `check_input` to verify that the user has not entered values outside the range 0 through 9. The function returns `true` if the range is ok and `false` otherwise.
- a function `compute_row_count` to count the number of time each digit occurs in an individual row. This may also need to take an array as a parameter, in order to store the count values.
- a function `print_total_count` to print out the total count for each digit. This may also need to take an array as a parameter. Note the difference between printing "1 time" and "2 times".

Code for this program should be in a file named `count_digits.c`

Program 2 (Array Formatting):

Write a program that takes in an NxM dimensional array of integers and prints it out formatted according to some user defined specifications. The user can define the following parameters:

- the number of empty (new) lines between rows
- the number of characters that each array element can span
- alignment: left or right justification
- what character should be placed in between the elements of each row

Your program should:

- prompt a user to enter the sizes (rows and columns) of the array, and read in the dimensions
- prompt the user to enter the array row-by-row and read in each row. Reuse your `read_row` function from Program 1.
- ask for the formatting specifications and check that they can be obeyed
- print the formatted array

Requirements: your program should use the following user defined functions:

- the function `read_row` from above
- a function `format_row` that formats an individual row based on specifications (these should be given as parameters to the function)
- a function `print_array` that uses `format_row` to print the entire array
- a function `check_formatting` that checks the feasibility of the formatting specifications: for example, if the user allows only 3 characters for each array element, but the array contains a number that is 4 digits long, the function should return false. This should cause the program to ask again for new formatting specifications.

Here is an example of how the program should behave (items underlined are entered by the user):

```
This program prints a formatted NxM array

Enter the size of your array: 2 6
Enter row 0: 0 1 2 3 4 5234
Enter row 1: 0 1 6 7 8 9
Enter the number of new lines between rows: 2
Enter the span of each array element: 3
The array contains elements longer than 3 digits long.
Enter the span of each array element: 6
Enter the type of justification (l - left, r - right): r
Enter the character to be placed between elements: #




Your formatted array is:
```

```
_ _ _ _ _0#_ _ _ _ _ 1#_ _ _ _ _ 2#_ _ _ _ _ 3#_ _ _ _ _ 4#_ _ 5234

_ _ _ _ _0#_ _ _ _ _ 1#_ _ _ _ _ 6#_ _ _ _ _ 7#_ _ _ _ _ 8#_ _ _ _ _ 9
```

Note: above the '_' represent blank spaces.

Code for this program should be in a file named `array_formatting.c`

Program 3 (Challenge):

`TBD`