Hacettepe University Computer Science Department

BBM 203 – Software Laboratory Programming Assignment 1

Name: Mert

Surname: Çökelek

ID: 21727082

Subject: Treasure Hunt

Date: 01.11.2018

In this assignment we are expected to write a C program that makes a treasure hunting with multiplying matrices.

In this assignment, our programming skills about "C pointers, multidimensional arrays, dynamic memory allocation, recursion and file i/o " were analyzed.

Problem Definition : Our code is expected to get the user input from commandline and output a file for the result.

During execution, our program must create 2d arrays according to the inputs, dynamically. The arrays contain Map matrix and Key matrix, respectively. After that, the Key matrix should be iterated over the Map matrix and each time, a matrix multiplication must be done and the current position and the result must be printed out the output file. This operation should be handled recursively. According to the result of that calculation, the next direction of move of iteration is specified. We take the mod5 of the result and we keep iterating and calculating until we get a result with mod5 = 0. This assures that our prorgam has found the hidden treasure. When we find the hidden treasure, program execution is completed succesfully.

The inputs are:

1- Size of Map Matrix - ("Row x Column" format).

2- Degree of Key Matrix (integer)

3- Txt file for the Map Matrix

4- Txt file for the Key Matrix respectively.

The output is:

The 5th argument of the commandline. A .txt extension file.

- First commandline argument allows us to dynamically create a 2 dimensional array with a specific size for holding the Map Matrix.
- Second argument allows us to dynamically create a 2 dimensional array with specific size for holding the Key Matrix.
- Third argument allows us to initialize our 2d Map array with values.
- Fourth argument allows us to initialize our 2d Key array with values.

- The last commandline argument is for printing the outputs in it.

Methods and Solution:

My approach to this assignment is:

1. Find the best way to work on the inputs and output.
   1.1. Get the inputs from commandline as expected,
   1.2. Store them in variables to use in the following time.
2. Determine how to read the input files.
   2.1. For this purpose, I have written 2 functions named “split” and “readMatrix”. (These functions will be mentioned more detailed in the next part.)
3. Determine what other functions are needed.
   3.1. For matrix multiplication, a matrixMul function defined.
   3.2. For recursive treasure hunting (the most important one), treasureHunt function.

Methods (Detailed):

1. **int* split (char* line, char* tok, int amount):**

Split function takes 3 parameters, and returns an integer array for the values.

1- The string part to be splitted

2- The token (splitting factor)

3- The size of the returning array.

In split function, a char pointer is iterated over the first parameter in a while loop, and until reaching the end of line, every pair of integers are taken without the token and stored in an array.

Lastly, the splitted array is returned.

2. int readMatrix(FILE* mapFile, int row, int col):**

This function takes 3 parameters for the file to be read and the size values for the returning array.

In a for loop, the 2d array for the matrices is created dynamically.

In a while loop, fgets function is used and every line in the file is splitted by the split function and stored in the array created in the previous loop.

Lastly, the created 2d array is returned.

3. int matrixMul(int map, int r, int c, int** key, int n):**

This function takes 5 parameters for the map matrix, row and column numbers of map, key matrix and degree of the key, respectively.

A result variable is created for returning.

In 2 for loops, key matrix is iterated over the submatrix of the map, and every pair of matching coordinates, the result variable is changed.

Lastly, the result is returned.

4. int treasureHunt(int map, int** key, int x, int y, int n, FILE* output, int mapLength, int mapWidth) :**

This function takes 8 parameters for map and key matrices, the starting coordinates of the calculation, degree of the key, output file, row and column numbers of the map, respectively.

This is a recursive function, first creates a result variable which is equal to the result of the matrixMul function for the first parameters. Then checks if the result mod5 is equal to 0, this is the Base Case.

In the recursive case, we are considering the results except for the ones whose mod5 is not equal to 0. There are 5 different cases here,

1. Result is negative: In this case, we add to the mod5 of it.
2. Result is 1: Slide the key upwards. If there is a boundary, downwards.
3. Result is 2: Slide the key downwards. If there is a boundary, upwards.
4. Result is 3: Slide the key to right side. If there is a boundary, left side.
5. Result is 4: Slide the key to left side. If there is a boundary, right side.

And after each calculation, print out the current coordinates and the result of the multiplication to the file.

<u>In my main function:</u>

1. I got the inputs from commandline and stored them in variables.

   The variables are:

   1- sizeOfMap: is an array of two integers, corresponding to the row and column amount of the map matrix, respectively. (first argument splitted by “x”)

   2- row: first element of the sizeOfMap

   3- col: second element of the sizeOfMap

   4- degree: 2nd argument, degree of the key matrix

   5- FILE* mapMatrix: 3rd argument is taken as the map.txt file.

   6- FILE* keyMatrix : 4th argument is taken as the key.txt file.

   7- FILE* output: 5th and the last argument corresponding the output file.

   8- map: the returned pointer of the readMatrix function for the mapMatrix file.

   9- key: the returned pointer of the readMatrix function for the keyMatrix file.

2. I only called the treasureHunt function with the parameters defined above.
3. After the function call, I freed all the memory allocated during this program.
4. Finally returned 0 for a clean exit.