Programming Assignment 1

Submission Date : 21.10.2022
Due Date : 04.11.2022 (23:59)
Programming Languages: C++
Subject : Multi-dimensional array, matrix
Advisors : R.A. Bahar GEZİCİ

1 Introduction

In this experiment, you are expected to implement an application which is the logic core simulator of a simple hidden treasure game. Main focus point of this experiment is to get you familiar with arrays, matrices, dynamic memory allocation and read-write files. Implementing the program without using the benefits of dynamic memory allocation will be penalized.

2 Background Information

2.1 C++ language

C++ was developed by Bjarne Stroustrup, as an extension to the C language. It gives programmers a high level of control over system resources and memory.

C++ is one of the world's most popular programming languages. It can be found in today's operating systems, Graphical User Interfaces, and embedded systems. It is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.

2.2 Arrays

An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier. It can be used to store the collection of primitive data types such as int, float, double, char, etc of any particular type. To add to it, an array in C++ can store derived data types such as structures, pointers etc.

C++ allows programmers to create statically or dynamically allocated arrays. A statically allocated array variable is a constant pointer that points to a fixed size array in automatic storage (i.e. the call stack). In contrast, pointer variables can be used to point to arrays allocated in dynamic memory (known as the heap or free store) that can be assigned different addresses (of other arrays in dynamic memory).

Static

```
int array[5]
```

Dynamic

```
int *p;
P=new int[5]
```

One more important point that you need to remember when we have allocated the memory in heap and after some time during the execution of the program if that memory is no more required then we must delete the memory. If we don't delete the unused memory then it causes a memory leak problem.

2.3 Multi-dimensional arrays

A multi-dimensional array can be termed as an array of arrays that stores homogeneous data in tabular form ($row \times column$) which is also known as matrix. We can create arrays with any number of dimensions. However, the complexity also increases as the number of dimensions increases. The most used multidimensional array is the Two-Dimensional Array.

2.4 Dynamic memory allocation

The task of fulfilling an allocation request consists of locating a block of unused memory of sufficient size. Memory requests are satisfied by allocating portions from a large pool of memory called the heap or free store. At any given time, some parts of the heap are in use, while some are "free" (unused) and thus available for future allocations. There may be cases where the memory needs of a program can only be determined during runtime. For example, when the memory needed depends on user input. On these cases, programs need to dynamically allocate memory, for which the C++ language integrates the operators **new** and **delete**.

2.5 Online resources

- C++ Tutorial
- C++ Wikipedia

3 Experiment

In this experiment, it is aimed to find the hidden treasure within a treasure map designed as a matrix. For this purpose, the map and key data must be read from the file. Afterwards, the key must be moved on the map depending on the rules. The result of the multiplication of the map matrix and the key matrix will give an idea of the place of the treasure.

You should use matrix structure to implement all these operations in the C++ programming language. To store the matrices, you must use dynamic memory allocation.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	3	1	5	8	6	9	4	7	5	1	2	9	2	8	3	6	0
0	1	5	8	3	2	8	4	6	9	3	7	4	6	8	9	3	0
0	1	6	4	9	3	5	8	9	2	8	1	8	9	7	3	1	0
0	5	7	5	8	1	3	4	8	5	1	5	8	9	3	5	2	0
0	9	4	7	5	1	2	9	4	7	5	1	2	9	2	8	3	0
0	8	4	6	9	3	7	8	4	6	9	3	7	4	6	8	9	0
0	5	8	9	2	6	1	5	8	9	2	6	1	8	9	7	3	0
0	3	8	1	3	4	8	5	1	5	3	8	1	3	4	8	5	0
0	7	5	1	2	9	4	7	5	1	7	5	1	2	9	4	7	0
0	6	9	3	7	8	4	6	9	3	6	9	3	7	8	4	6	0
0	9	2	6	1	5	8	9	2	3	4	8	5	1	5	8	9	0
0	8	4	6	9	3	7	3	7	2	9	4	7	5	1	2	9	0
0	5	8	9	2	6	1	6	1	7	8	4	6	9	3	7	4	0
0	3	8	1	3	4	8	2	9	1	5	8	9	2	6	1	8	0
0	7	5	1	2	9	4	8	1	3	4	8	8	4	6	8	4	0
0	8	4	6	9	3	7	5	1	2	9	4	5	8	9	5	8	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1: An Example for Treasure Map Matrix

The treasure map is designed as a matrix as shown in Figure 1. The size of the map matrix is not fixed but must be determined according to the input file that will be given at runtime. Each element of the map matrix is a number. Map matrix may include negative or more than one digit numbers. The elements of the matrix placed in the edges are zeros. That show you the boundary of the map.

0	-1	0
-1	20	-1
0	-1	0

Figure 2: An Example for Key Matrix

As shown in Figure 2, the key is designed as a square matrix with degree n that is an odd value (e.g. 3,5,7). n is a variable that will be determined at runtime. By using the key matrix, the hidden treasure can be found in the map.

In order to find treasure, we slide the key matrix on the map. The part of the map matrix that is covered by the key specify a sub-matrix on the map. By multiplying the key matrix by the obtained sub-matrix, we find the direction of the next sliding. The direction is computed by taking mod five of multiplication result (result % 5). The result of the mod operation is interpreted as follows:

(Result%5)	0	1	2	3	4
Action	Found treasure	Go up	Go down	Go right	Go left

The search operation must be started from the top left corner of the map matrix. The details of the first step are shown in Figure 3. In order to get a full score from this assignment, you have to design search function recursively.

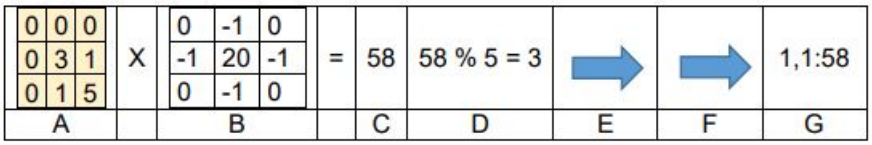

Figure 3: Start search

A: Sub-matrix
B: Key Matrix
C: Multiplication of matrices A and B (Dot product)
(Summation of all product of each corresponding entries).
D: Remaining part of the mod (5,C)
E: Estimated direction
F: Final direction
G: Output: write center cell address of sub-matrix (with the order *Row, Column*) and C

P.S. If the C is negative, then add 5 in negative cases.

In case of key matrix is placed on the boundary of map (e.g. most right or most left) where there is no way to slide through the determined direction, you must move in the opposite direction as shown in Figure 4. Actually, the estimated direction may be changed by the mentioned exception.

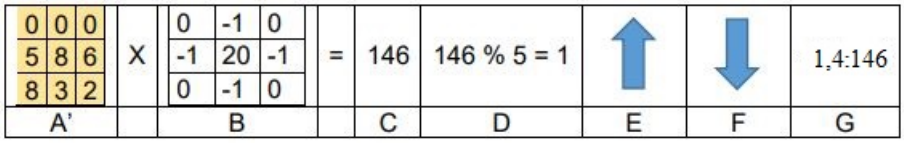

Figure 4: Handling the exception for the estimated direction

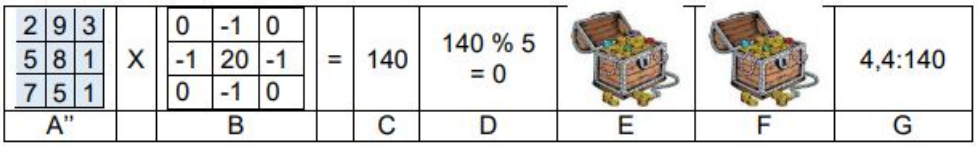

Figure 5: Treasure Found!

If the mod result is zero, the treasure is found as shown in Figure 5. The place of the treasure is the midpoint of the sub-matrix. Your program must print out the midpoints of the sub-matrix in each step.

P.S. There will not be a map exception with no treasure.

3.1 Execution

The name of the compiled executable program should be "findtreasure". Your program should read input/output file names from the command line, so it will be executed as follows:

findtreasure [size of map matrix] [size of key matrix] [input file name] [output file name] e.g. ./findtreasure 18x18 3 mapmatrix.txt keymatrix.txt output.txt

You can see sample input and output in piazza page. The program must run on DEV (dev.cs.hacettepe.edu.tr) UNIX machines. So make sure that it compiles and runs on dev server. If we are unable to compile or run, the project risks getting zero point. It is recommended that you test the program using the same mechanism on the sample files (provided) and your own inputs. You must compare your own output and sample output. If your output is different from the sample, the project risks getting zero point, too.

3.2 Input/Output file format

In the map matrix file, the numbers are separated by spaces and newlines. You do not need to check the input for errors. An example map matrix file can be given as follows:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 1 5 8 6 9 4 7 5 1 2 9 2 8 3 6 0
0 1 5 8 3 2 8 4 6 9 3 7 4 6 8 9 3 0
0 1 6 2 9 3 5 8 9 2 6 1 8 9 7 3 1 0
0 5 7 5 8 1 3 4 8 5 1 5 8 9 3 5 2 0
0 9 4 7 5 1 2 9 4 7 5 1 2 9 2 8 3 0
0 8 4 6 9 3 7 8 4 6 9 3 7 4 6 8 9 0
0 5 8 9 2 6 1 5 8 9 2 6 1 8 9 7 3 0
0 3 8 1 3 4 8 5 1 5 3 8 1 3 4 8 5 0
0 7 5 1 2 9 4 7 5 1 7 5 1 2 9 4 7 0
0 6 9 3 7 8 4 6 9 3 6 9 3 7 8 4 6 0
0 9 2 6 1 5 8 9 2 3 4 8 5 1 5 8 9 0
0 8 4 6 9 3 7 3 7 2 9 4 7 5 1 2 9 0
0 5 8 9 2 6 1 6 1 7 8 4 6 9 3 7 4 0
0 3 8 1 3 4 8 2 9 1 5 8 9 2 6 1 8 0
0 7 5 1 2 9 4 8 1 3 4 8 8 4 6 8 4 0
0 8 4 6 9 3 7 5 1 2 9 4 5 8 9 5 8 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 6: Sample map matrix file

Your program should write outputs to the output file, so that each line in the output file will contain the midpoint of the sub-matrix and also the result of multiplication. If the above input file is given, the output file should be as below:

```
1,1:58
1,4:146
4,4:140
```

Figure 7: Sample output file

3.3 Design Expectations

You must use dynamic memory allocation for your solution. Writing spaghetti, or static arrays could render your entire assignment "unacceptable" and make it subject to grade loss. Your source code must have "Dynamic memory allocation, multidimensional array and comment lines". You can achieve 5 points for each item, total 15 points.

3.4 Required Files and Submit Format

Do not submit any file via e-mail. You should upload your files via "Online Experiment Submission System" which is at http://submit.cs.hacettepe.edu.tr.
You should create and submit a ZIP archive in the following structure for evaluation. An invalid structured archive will cause you partial or full score loss. You are required to submit a Makefile, which will be used to compile your program to generate the findtreasure executable.

- student id.zip < src (*.cpp, *.h, Makefile) >

4 Grading Policy

Task	**Point**
Compiled	10
Design	15
Output	75
Total	100

P.S. The Compiled part represents the note that will break if there is a compilation error and your code works after the small fix.

5 Notes

- The assignment must be original, individual work. Downloaded or modified source codes will be considered as cheating. Also the students who share their works will be punished in the same way.
- We will be using the Measure of Software Similarity (MOSS) to identify cases of possible plagiarism. Our department takes the act of plagiarism very seriously. Those caught plagiarizing (both originators and copiers) will be sanctioned.
- You can ask your questions through course's piazza group and you are supposed to be aware of everything discussed in the piazza group. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms, source codes and reports .
- Questions asked over the Piazza will be answered within working hours on weekdays. On weekends, we will try to answer as much as possible, but in case of possible unavailability, please do not insist on getting an answer.

- Ignore the cases which are not stated in this assignment and do not ask questions on Piazza for such extreme cases.
- Don't forget to write comments of your codes when necessary.
- Do not submit your project without first compiling it on dev machine. Make sure that the source code you submitted is compiled with gcc compiler under Linux OS. Applications that produce compilation or runtime errors cannot be graded.
- Save all work until the assignment is graded.
- Do not miss the deadline. Submission will be end at 04/11/2022 23:59, the system will be open until 23:59:59. The problem about submission after 23:59 will not be considered.
- Your grades will be announced within 15 days at the latest. Objections about your grades can be made within specified days given by TAs. Objections made outside of the specified days will not be accepted.