

Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2020

Homework 4 – Area Filling using Stacks

Due: 23/03/2020, Monday, 21:00

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism will not be tolerated!

Introduction

In this homework, you are asked to implement an *area filling* program which makes use of *stack* data structure. Area filling is the process of filling a connected and closed area in a map with a given value (color or a filling character). The map is a 2D array with solid boundary and some occupied cells within itself. A connected and closed area in this map contains consecutive empty cells surrounded by the boundary and/or occupied cells. Such an area can be of any shape (including amorphous shapes). In the program that you will write, the user will enter a coordinate of an empty cell of an area in the map. After that, your program should fill all the empty cells in the corresponding area with a filling character. The details of the methods, algorithms and the data structure will be given in the subsequent section of this homework specification.

General idea of area filling

In this section, we will explain the general idea of area filling via some examples. An example of a map which has 11 rows and 9 columns can be seen in Figure 1 below. In this homework, we assume that the map is of rectangular shape and there is a boundary around it.

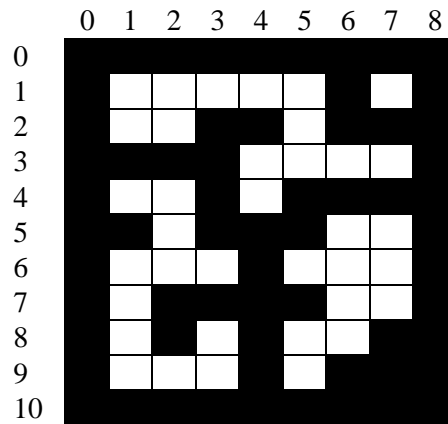


Fig 1. An example map

A map can have any number of connected and closed areas. In the map of Figure 1, there are four areas of different shapes and different numbers of cells. In order to fill an area, the user will enter a starting coordinate of an empty cell in that area. For example, let us assume that (2, 5) is given as the starting point. If the starting point is empty, your program must fill the connected and closed area which contains this point. The point (2, 5) is empty in the given example. For the sake of simplicity of explanation, let us fill this area with blue (in the actual program you will use character filling). The resulting filled map can be seen in Figure 2.

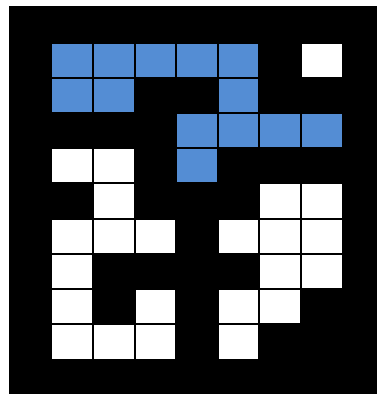


Fig 2. An example result

Instead of (2, 5), if the user chooses to start from (7, 6), the result must be like in Figure 3.

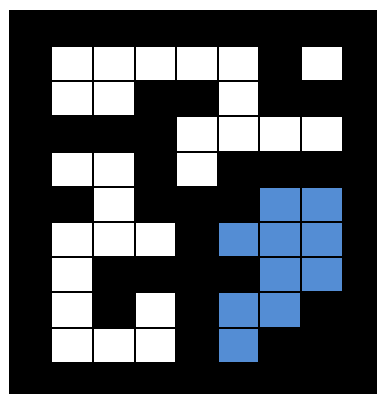


Fig 3. An example result

Here please remark that the program that you will implement will use ASCII characters for borders, occupied cells and filling. The details of the data structure and input/output will be given in the next sections.

Input and output

Firstly, the program asks the number of rows and columns of the map (first number of rows and then number of columns). Here, you need to check if these values are positive integers that are greater than or equal to 3; if not, your program should display an error message and then ask for a new input until values that are greater than or equal to 3 have been entered. Then, your program asks the name of the file which includes the map with the given number of rows and columns. Coordinates begin with zero for both rows and columns. You can assume that the file has data for a correct rectangular map in terms of row and column counts entered by the user. You can also assume that the map has a boundary around it. However, the boundary is also counted in the numbers of rows and columns. You do not need to make an input check for the file content, but you should check whether the file exists; if it does not, then your program should ask for a new file name until file is opened successfully.

After reading map from the file, in order to fill an area in this map, a starting point is entered from the keyboard. This input consists of two values: row and column coordinates (you have to read first row and then column). These values must be within the range of the matrix. If not, your program should repeatedly ask for them until valid values are entered. Moreover, the starting point must be empty. If the entered starting point is occupied, then your program should not perform filling and terminate after giving a message. The connected and closed area is filled with a character taken from the user as input from the keyboard. The filling character should not be 'X' or 'x'. If an invalid filling character is entered, it must be repeatedly entered until user enters a valid one. See sample runs for example cases.

We know the limitations of the console and printable characters. Therefore, in the text file, non-empty cells in the map are represented with 'X'. On the other hand, empty cells are blank character, ' '. Since it is not possible to fill a text file with colors, we fill it with a character taken from the user as input. The text file version of the example given in Figure 1 can be seen below in Figure 4.

```
X X X X X X X X X
X           X  X
X       X X   X X X
X X X X      X
X       X   X X X X
X X   X X X   X
X       X       X
X   X X X X   X
X   X   X   X X
X       X   X X X
X X X X X X X X X
```

Fig 4. An example text file

If we fill the area with character 'O' and the starting point is (2, 5), the result must be like in Figure 5.

```
X X X X X X X X X
X O O O O X  X
X O O X X O X X
X X X X O O O X
X       X O X X X
X X   X X X   X
X       X       X
X   X X X X   X
X   X   X   X X
X       X   X X X
X X X X X X X X X
```

Fig 5. An example result

If we fill the area with character “s”, and the starting point is (7, 6), the result must be like in Figure 6.

```

X X X X X X X X X
X           X   X
X       X X   X X X
X X X X           X
X       X   X X X X
X X   X X X s s X
X       X s s s X
X   X X X X s s X
X   X   X s s X X
X       X s X X X
X X X X X X X X X

```

Fig 6 An example result

Data structures used

The map is actually a character matrix. In your program, you will create a **dynamic char matrix**, without using `vector` class, to store the characters in the map.

Another data structure that you will use is a *stack*. As will be discussed in the next section, in order to keep track of visited cells while scanning the area, you must use a stack data structure. The data that you will keep in each element of this stack are *row* and *column* coordinates of a cell in the map. We are expecting you to implement a dynamic stack class for this application and use it in the main program. The dynamic stack class that you will implement must contain only *push*, *pop*, and *isEmpty* member functions other than the standard ones (constructor, destructor, deep copy constructor and assignment operator). In other words, you are not allowed to add extra public member functions that violate the spirit behind stack (i.e. a member functions such as to display the content, check the top element without popping, insert in the middle, etc. are NOT allowed). If some helper functions are needed (e.g. `createClone`), define them as private and use only in the member function implementations. In the program part, that you will fill the area of the map, your programmer's interface will only be classical stack operations (constructor, copy constructor, = operator, destructor, push, pop, and `isEmpty`). Actually, you may not want to implement deep copy constructor and assignment operator if you do not need (this is up to you). However, you must write a constructor and a destructor for your dynamic stack class. During the implementation of class member functions, you can (actually have to) use linked lists.

You are not allowed to use any other container in this homework.

Algorithmic details and hints

Since it may not be so clear to you how to scan an area of an amorphous shape, we give some algorithmic hints in this section. The main idea of the algorithm that you will use is to keep track of all visited cells (by pushing them onto the stack) and backtrack to the previously visited cell (by popping from the stack) when you are stuck.

You have to start from the starting coordinate and fill it with the filling character. After that you have to move towards an empty neighbor in one of four directions (north, east, south, west). Crosswise neighbors are not taken into consideration. If you can move in one of those directions, first remember the current cell (using stack), and then move. When you move to this neighbor, fill it and do the same until you are stuck (i.e. no empty neighbors). If you are stuck, you have to go back (using stack) and try to find an available cell with an empty

neighbor. When you backtrack to such a cell, continue as above. If you follow this algorithm properly, when you fill all the cells in an area, the stack must be empty.

At the end, your program must display the resulting filled map to the screen.

There also exist recursive solutions to the area filling problem. However, we **strongly recommend** you **not** to use recursion for this problem. If you use recursion for area filling problem, at each function call you can make as much as four recursive calls (one for each direction). For some big (actually not so big) maps, these recursive calls cause to use up all available memory and your program crashes. If you write your code recursively and if your recursive program does not work with our grading cases, we do **not** take any responsibility.

Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate. Since using classes is mandated in this homework, a proper object-oriented design and implementation will also be considered in grading.

Since you will use dynamic memory allocation in this homework, it is very crucial to properly manage the allocated area and return the deleted parts to the heap whenever appropriate. Inefficient use of memory may reduce your grade.

When we grade your homework we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases**. Of course, your program should work in *Debug* mode as well.

You are allowed to use sample codes shared with the class by the instructor and TAs. However, you cannot start with an existing .cpp or .h file directly and update it; you have start with an empty file. Only the necessary parts of the shared code files can be used and these parts must be clearly marked in your homework by putting comments like the following. Even if you take a piece of code and update it slightly, you have to put a similar marking (by adding "and updated" to the comments below.

```
/* Begin: code taken from ptrfunc.cpp */
```

```
...
```

```
/* End: code taken from ptrfunc.cpp */
```

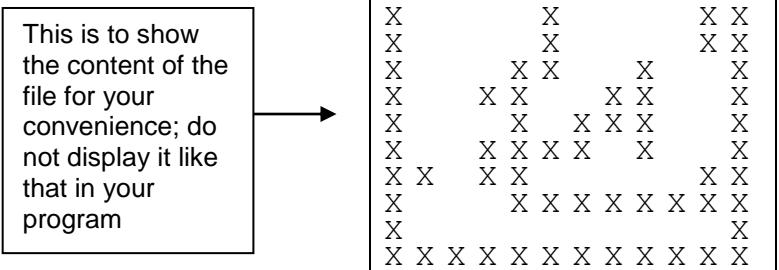
Sample runs

Some sample runs are given below, but these are not comprehensive, therefore you have to consider **all possible cases** to get full mark. Please do not make any assumptions on the names of files.

Sample run 1:

File: 11x12.txt

This is to show
the content of the
file for your
convenience; do
not display it like
that in your
program



```
X X X X X X X X X X X X
X           X           X X
X           X           X X
X           X X         X X
X           X X         X X
X           X X X X     X X
X           X X X X     X X
X X        X X         X X
X           X X X X X X X X
X           X X X X X X X
X           X X X X X X
X X X X X X X X X X X X
```

```
Enter the number of rows: -1000
-1000 is not valid!
Enter the number of rows: 0
0 is not valid!
Enter the number of rows: 2
2 is not valid!
Enter the number of rows: ASD
ASD is not valid!
Enter the number of rows: 11
Enter the number of columns: -1000
-1000 is not valid!
Enter the number of columns: 0
0 is not valid!
Enter the number of columns: 2
2 is not valid!
Enter the number of columns: ASD
ASD is not valid!
Enter the number of columns: 12
Please enter file name: 11x12
Cannot open a file named 11x12
Please enter file name: 11x12.txt
Enter the starting point: ASD ASD
Invalid coordinate!
Enter the starting point: 1 ASD
Invalid coordinate!
Enter the starting point: ASD 1
Invalid coordinate!
Enter the starting point: -1 10
Invalid coordinate!
Enter the starting point: -1 ASD
Invalid coordinate!
Enter the starting point: 11 12
Invalid coordinate!
Enter the starting point: 10 11
Starting point is already occupied.
Terminating...
```

Sample run 2:
File: 11x12.txt

This is to show
the content of the
file for your
convenience; do
not display it like
that in your
program

```

X X X X X X X X X X X X
X           X           X X
X           X           X X
X           X X           X
X           X X           X X
X           X   X X X X   X
X           X X X X   X   X
X X       X X           X X
X           X X X X X X X X
X           X X X X X X X
X X X X X X X X X X X X

```

```

Enter the number of rows: 11
Enter the number of columns: 12
Please enter file name: 11x12.txt
Enter the starting point: 1 1
Enter the filling char: x
Filling char is not valid!
Enter the filling char: X
Filling char is not valid!
Enter the filling char: O

```

```

XXXXXXXXXXXXX
XOOOOX      XX
XOOOOX      XX
XOOOXX  X  X
XOOXX   XX  X
XOOOX  XXX  X
XOOXXXXX X  X
XXOXX      XX
XOOOXXXXXXXXX
XOOOOOOOOOOOX
XXXXXXXXXXXXX

```


File: 34x51.txt

File: 34x51.txt

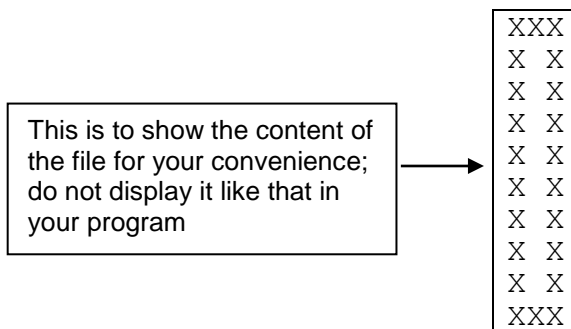
This is to show the content of the file for your convenience; do not display it like that in your program

[illegible]

```
Enter the number of rows: 34
Enter the number of columns: 51
Please enter file name: 34x51.txt
Enter the starting point: 16 25
Enter the filling char: Q
```

[illegible]

Sample run 7:
File: 3x10.txt



```
Enter the number of rows: 10
Enter the number of columns: 3
Please enter file name: 10x3.txt
Enter the starting point: 5 1
Enter the filling char: \
```

```
XXX
X\X
X\X
X\X
X\X
X\X
X\X
X\X
X\X
X\X
XXX
```

What and where to submit (PLEASE READ, IMPORTANT)

You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course).

Submissions guidelines are below. Some parts of the grading process might be automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Name your cpp file that contains your main program using the following convention:

“SUCourseUserName_YourLastname_YourName_HWnumber.cpp”

Your SUCourse user name is your SUNet user name which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For

example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroglu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw4.cpp

In some homework assignments, you may need to have more than one .cpp or .h files to submit. In this case, add informative phrases after the hw number. However, do not add any other character or phrase to the file names. Sometimes, you may want to use some user defined libraries (such as strutils of Tapestry); in such cases, you have to provide the necessary .cpp and .h files of them as well. If you use standard C++ libraries, you do not need to provide extra files for them.

These source files are the ones that you are going to submit as your homework. However, even if you have a single file to submit, you have to compress it using ZIP format. To do so, first create a folder that follows the abovementioned naming convention ("SUCourseUserName_YourLastname_YourName_HWnumber"). Then, copy your source file(s) there. And finally compress this folder using WINZIP or WINRAR programs (or another mechanism). Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains all of the files that belong to the latest version of your homework.

You will receive zero if your compressed zip file does not expand or it does not contain the correct files. The naming convention of the zip file is the same. The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example, zubzipler_Zipleroglu_Zubeyir_hw4.zip is a valid name, but

Hw4_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!

Albert Levi, Vedat Peran