UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS110 - Program Design: Introduction

Practical 1 Specifications: Dynamic Jagged Array

Release Date: 31-07-2023 at 06:00
Due Date: 04-08-2023 at 23:59
Total Marks: 65

Contents

1 General Instructions

- ***Read the entire assignment thoroughly before you start coding.***
- This assignment should be completed individually; no group effort is allowed.
- **To prevent plagiarism, every submission will be inspected with the help of dedicated software.**
- Be ready to upload your assignment well before the deadline, as no extension will be granted.
- You may not import any of C++'s built-in data structures. Doing so will result in a mark of zero. You may only make use of 1-dimensional native arrays where applicable. If you require additional data structures, you will have to implement them yourself.
- If your code does not compile, you will be awarded a mark of zero. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- If your code experiences a runtime error, you will be awarded a mark of zero. Runtime errors are considered unsafe programming.
- Read the entire specification before you start coding.
- **Ensure your code compiles with C++98**

2 Plagiarism

The Department of Computer Science considers plagiarism a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to http://www.library.up.ac.za/plagiarism/index.htm (from the main page of the University of Pretoria site, follow the Library quick link, and then choose the Plagiarism option under the Services menu). **If you have any form of question regarding this, please ask one of the lecturers to avoid any misunderstanding.** Also note that the OOP principle of code reuse does not mean that you should copy and adapt code to suit your solution.

3 Outcomes

On completion of this practical, you will have gained experience with the following:

- Reading in data from textfiles and from the terminal.
- Creating dynamic jagged arrays.

4 Introduction

A dynamic array is an array whose size is not known beforehand. Thus, the size of the array is determined at runtime, and from this, the required memory is allocated. Since the size is not known beforehand and the memory is allocated by the program, it is also vitally important that the memory be deallocated at the end of the program runtime. If this does not happen, memory leaks might occur, which needs be avoided at all costs.

A jagged array is a multidimensional array in which the size of the elements are not consistent. For the purpose of this practical, you will be implementing a dynamic 2D array of dynamic integers. This implies that the size of each row (or the number of columns in each row) can vary from row to row.

5 Important Information and Hints

Since this is the first time a lot of you will be using FitchFork(FF) for marks, below is some important information and tips. Please note that we will be assuming you know all of this for all future practicals and assignments, and thus we won't be repeating this.

- FF related
  - You have 10 submission attempts. (For Assignments, you only have 5)
  - Only the best mark counts.
  - You can only see the feedback we provide, not the output of the program. If you need help understanding the feedback, please go to a prac session or create a ticket on Discord.
  - Make sure there are no folders inside your submission.
  - Make sure you don't include any libraries other than the ones we said are allowed.
  - Make sure you don't change the function names, return types or input parameters.
  - If one of your tasks crashes, you will receive 0 for that task.
  - Note we will use our own testing code during marking. Thus we can call the functions out of order, so make sure every function does exactly what the spec says it must do and nothing more or less.
- c++98
  - Your code MUST be able to compile with c++98.
  - You can do this by adding the following flag when compiling your code:

```
-std=c++98
```

  - If your code doesn't compile with c++98 you will receive 0.
  - Common things students use which are NOT c++98, and how to fix it.
    - stoi. Solution : Use stringstream
    - nullptr. Solution : Use NULL
    - fstream file(string). Solution call .c_str() on the string.
- how to read from text files
  - Use the following to create a file object

```
std::fstream file(fileName.c_str());
```

  - Use the following to store the next line in the file to a string called line. You can use this as the condition of a while loop, which will then stop when the end of the file is reached.

```
getline(file,line);
```

  - Remember to close the file once you are done with it

```
file.close();
```

- String manipulation
  - To convert from a string to an integer or vice versa, use the stringstream library.
  - string.find(char)
    - This function returns the position of the first occurrence of char inside string. If the char is not found inside string then "std::string::npos" is returned.
    - This example will print out "2"

```
std::string myString = "abcdef";
std::cout << myString.find('c');
```

    - This example prints out "True

```
std::string myString = "abcdef";
if(myString.find('z') == std::string::npos){
    std::cout << "True";
}
```

  - string.substr(start,howMany)
    - This function returns part of the string.
    - start is an integer indicating where the copying should start
    - howMany is an integer indicating how many characters should be copied
    - This example prints out "cde"

```
std::string myString = "abcdef";
std::cout << myString.substr(2,3);
```

  - string.erase(start,howMany)
    - This function deletes part of the string
    - start is an integer indicating where the deleting should start
    - howMany is an integer indicating how many characters should be deleted
    - This example prints out "abf"

```
std::string myString = "abcdef";
myString.erase(2,3);
std::cout << myString;
```

6 Tasks

6.1 main.cpp

You only need to upload the main.cpp. A skeleton has been provided for you, **do not change any of the given signatures**, as this will result in a mark of 0, if it cannot compile. Also, note that we will use our own main function to mark your code, so feel free to change the **contents** of the main function that is provided.

- Members:
  - myArr:int***
    - This is the jagged dynamic 2D array that consists of dynamic integers.
    - Make sure the array is perfectly sized, i.e. every row should have the **exact** amount of columns needed to store that row.
    - This array has numRows rows, and the size of every row is stored inside numCols.
    - If myArr is empty, it means it will be NULL.
  - numCols:int**
    - This is a 1D dynamic array consisting of dynamic integers with a size of numRows.
    - This array is used to store the size of each row, i.e. the number of columns in each row.
    - If myArr is empty, it means it will be NULL.
  - numRows
    - This integer is used to store the number of rows in myArr.
    - If myArr is empty, then its value will be 0.
- Functions:
  - clearArray(): void
    - This will deallocate the memory stored in myArr and numCols.
    - Print out the following on its own line, replacing X with numRows:

      ```
      Deleted array with X rows
      ```
      1

    - After this function has been called, myArr should be empty.
  - populateFromFile(fileName: std::string): void
    - If myArr is not empty, you need to first call clearArray() to ensure that no memory is leaked.
    - The array, numCols and numRows should then be populated from the text file which is passed in as a parameter.
    - Every line in the text file represents its own line. Note that lines may be empty. If this is the case, that specific row in the 2D array should have a size of 0.
    - On every line, the entries will be separated by a comma. You may assume that all of the lines are valid input. There is **no** comma at the end of each line.
    - Note that the number of lines in the textfile and the number of columns in each row is not explicitly stored inside the text file. It is up to your code to figure that out. *Hint: You are allowed to read through the text file multiple times.*
    - Print out the following on its own line, replacing X with numRows:

      ```
      Created array from textfile with X rows
      ```
      1

– populateFromTerminal(): void

* This is the same as populateFromFile(), but now the input comes from the terminal and should be read in using cin.
* The output will once again be split by newlines and commas, as mentioned with the previous function.
* The first line will be the number of rows for myArr. You may assume this will be larger than or equal to 0.
* You may also assume that all lines will be valid, with blank lines being considered **invalid** for this function (thus the part in the previous function with rows of size 0 is not applicable).
* Print out the following on its own line, replacing X with numRows:

```
Created array from terminal with X rows
```

* *Example:* If the following code is run:

```
int main() {
    std::cout << "Reading:\n";
    populateFromTerminal();
    std::cout << "Printing\n";
    printArr();
    printArrStructure();
    clearArray();
    return 0;
}
```

Then the terminal may look like this (note that the blue block was the keyboard input):

```
Reading:
3
1
1,2
1,2,3
Created array from terminal with 3 rows
Printing
1
1,2
1,2,3
3:[1,2,3]
Deleted array with 3 rows
```

Figure 1: Terminal Input Example

– printArr(): void

* If myArr is empty, then print the following on its own line:

```
Array is empty
```

* If myArr is not empty, then print the array as follows:
  - Every row should be on its own line.
  - Each column should be separated by a comma, and there must be **no** comma at the end of each line.
  - If there is an empty row, then print out an empty line.

- printArrStructure(): void
  - If myArr is empty, then print the following on its own line:

```
Array is empty
```

  - If myArr is not empty, then print the following on its own line:

```
X:[a,b,c,d,e,f]
```

    - Where X is the number of rows
    - a,..,f is the length of each row i.e. the number of columns in each row.
- main(): int
  - This is used to test your code.
  - An example has been provided with the correct output being saved inside 'output.txt'. Use this to make sure you have no formatting issues.
  - This function won't be marked. Thus, you can change it and expand on testing. This main is trivial, don't expect to get any marks just because you get this function's output. You will need to test your code and ensure everything works according to the specification.

7 Memory Management

As memory management is a core part of COS110 and c++, each task on FitchFork will allocate approximately 10% of the marks to memory management. The following command is used:

```
valgrind --leak-check=full ./main
```

Please ensure, at all times, that your code *correctly* de-allocate *all* the memory that was allocated.

8 Upload Checklist

- main.cpp

9 Allowed Libraries

- fstream
- iostream
- sstream

10 Submission

You need to submit your source files, only the .cpp files, on the FitchFork website (https://ff.cs.up.ac.za/). All methods need to be implemented (or at least stubbed) before submission. Place the above-mentioned files in a zip named uXXXXXXXX.zip where XXXXXXXX is your student number. There is no need to include any other files or .h files in your submission. Your code must be able to be compiled with the C++98 standard

For this practical, you will have 10 upload opportunities and your best mark will be your final mark. Upload your archive to the appropriate slot on the FitchFork website well before the deadline. **No late submissions will be accepted!**