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

Practical 1 Specifications:
Dynamic Arrays

Release Date: 02-08-2022 at 06:00

Due Date: 05-08-2022 at 23:59

Total Marks: 35

Contents

1 General instructions

- This practical should be completed individually, no group effort is allowed.
- You may not include/import any other modules than what were already included in the given main.cpp file
- 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.
- You are not allowed to plagiarise.
- You will be afforded 10 upload opportunities.

2 Plagiarism

The Department of Computer Science considers plagiarism as 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 `https://www.up.ac.za/students/article/2745913/what-is-plagiarism`. **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 re-use does not mean that you should copy and adapt code to suit your solution.

3 Introduction

3.1 Scenario

For this practical you will be assisting a building manager with checking the offices they are currently renting out. The building manager has a list of floors and the people renting on those floors called building.data. The manager needs to check specific floors and would like you to provide them with a program that will read this list and give them the name of all the peoples' offices that they need to check in on.
Unfortunately when the building was built the elevator was placed on the opposite side of the building thus you will also need to print the names for the manager in **reverse order** since they are coming from the other side of the floor now.

3.2 Valgrind

As a software developer it is your job to ensure you write code that is both efficient and memory safe. Thus for this practical your submission will be tested for memory leaks. An easy way to check if your program has memory leaks is with a tool called Valgrind. For Linux users you would need to run the following command to install valgrind:
sudo apt-get install valgrind
Then you need to edit your makefile run command to run with valgrind instead of the plain run. To do this change:

```
./main
```

to

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

If you have a memory leak, valgrind will print a leak summary that will look something like:

```
LEAK SUMMARY:
definitely lost: 48 bytes in 1 blocks
indirectly lost: 0 bytes in 0 blocks
possibly lost: 0 bytes in 0 blocks
still reachable: 0 bytes in 0 blocks
suppressed: 0 bytes in 0 blocks
```

If you do have memory leaks you are not deleting memory that you have allocated.

3.3 Mark Distribution

Activity	Mark
Task 1	20
Valgrind	15
Total	35

4 Your Tasks

A dynamic array is a basic data structure that is widely used. Your task is to implement such a data structure in C++98. You are required to read the given text file and create dynamic arrays according the data within the text file. For this practical you will be given:

- main.cpp
- building.data

You are required to write all your code inside of the given **main.cpp** file. Inside your main file you are given a simple string to integer function that you may use inside your program. You are also allowed to add any other helper functions.

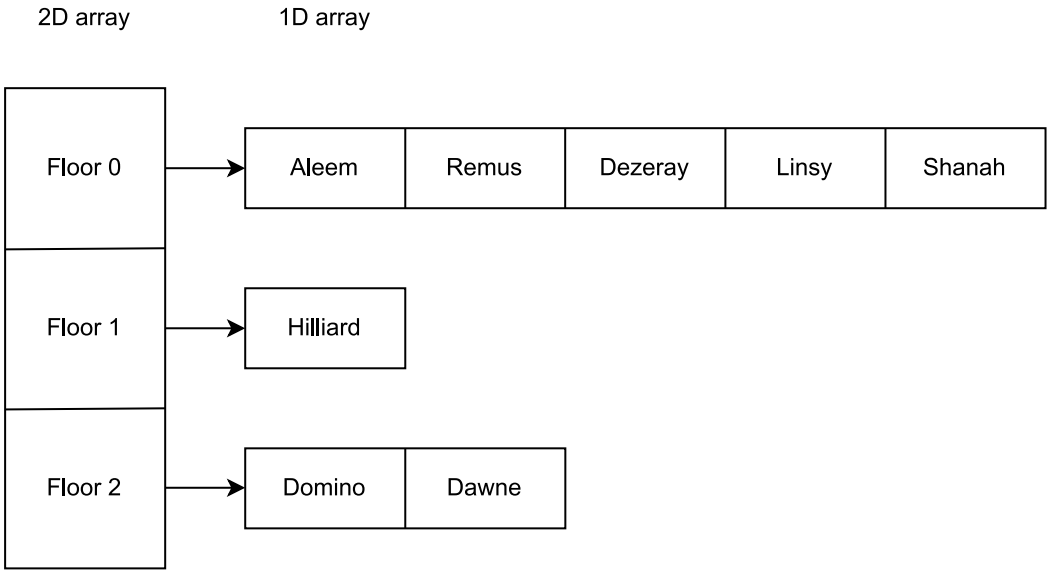
At the start of your program you will need to open and read the values inside the **buidling.data** file. The file is made up of an unknown number of lines.
The format of building.data is as follows:

```
1,0,2                                                    1
Floor ID:0                                               2
Aleem,Remus,Dezeray,Donisha,Linsy,Shanah                 3
Floor ID:2                                               4
Domino,Dawne                                             5
Floor ID:1                                               6
Hilliard                                                 7
```

Every floor exists from id 0 to the size of the building but not every floor will be visited.
Line 1 indicates what floors(using the floor id as reference) **need to be visited** and in what **order** they need to be visited. Where the left-most id is first and the right-most index is last. Thus in the above example floor 1 needs to be visited first then floor 0 (or ground floor) then lastly the 2nd floor. Then it will list all the floors and their information in a random order.
Each Floor ID indicates the start of a new floor and it's information. The line underneath the Floor ID is a list of everyone currently renting on that floor.
You may assume that every floor will have at least 1 office and that every floor that needs to be visited will be present in the list.

4.1 Task 1

For this task you will need to save the floors that need to visit. Then you will need to read and store all the floors from the building text file inside a dynamically allocated 2D string array. Using the id of the floor as the index for the 2D array, and at each index you would store all the people who have offices on that floor inside a 1D string array.
Example:



Floor id 0 would be at index 0 and it points to a string array of all the people who have offices on that floor.
You will then need to iterate through the floors that needs to be visited and only print the names of the people **in reverse** with each floor separated by a newline and each name separated by a comma. Ensure that you print **according to the order listed** and that you only print the names and that they are in **reverse order**.
Example
Given building.data as:

```
1,0,2
Floor ID:0
Aleem,Remus,Dezeray,Donisha,Linsy,Shanah
Floor ID:2
Domino,Dawne
Floor ID:1
Hilliard
```

The output of your code would be

```
Hilliard
Shanah,Linsy,Donisha,Dezeray,Remus,Aleem
Dawne,Domino
```

In the above example the order that we need to visit the floors are: 1, 0, 2. So we first print the names from floor 1 in reverse. Then floor 0 and lastly floor 2.

4.2 Task 2

For this task you are required to ensure your program is memory safe.
You must ensure that you are not allocating more memory than necessary and that you are freeing the memory that you have allocated during runtime.
This will be tested using Valgrind. If your program uses to much memory or causes memory leaks you will not get full marks for this task.

5 Submission

You need to submit your source files on the Fitch Fork website (https://ff.cs.up.ac.za/). Only the following files should be in a zip archive named uXXXXXXXX.zip where XXXXXXXX is your student number:

- main.cpp
- makefile
- building.data

Ensure that your archive is correct and that your files are not within another folder within the archive.
Your code should be able to be compiled/executed with the following commands:

- make
- make run
- make clean
- valgrind –leak-check=full ./main

For this practical you are not allowed to use any other includes than the ones listed below.

- iostream
- fstream
- string
- sstream
- iomanip

You have 10 submissions and your best mark will be your final mark. Upload your archive to the Practical 1 slot on the Fitch Fork website. **No late submissions will be accepted!**