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 3 Specifications

Release Date: 12-08-2024 at 06:00

Due Date: 16-08-2024 at 23:59

Total Marks: 190

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.**
- If your code does not compile, you will be awarded a mark of 0. The output of your program will be primarily considered for marks, although internal structure may also be tested (eg. the presence/absence of certain functions or classes).
- Failure of your program to successfully exit will result in a mark of 0.
- Note that plagiarism is considered a very serious offence. Plagiarism will not be tolerated, and disciplinary action will be taken against offending students. Please refer to the University of Pretoria's plagiarism page at `https://portal.cs.up.ac.za/files/departmental-guide/`.
- Unless otherwise stated, the usage of c++11 or additional libraries outside of those indicated in the assignment, will **not** be allowed. Some of the appropriate files that you have submit will be overwritten during marking to ensure compliance to these requirements. **Please ensure you use c++98**
- All functions should be implemented in the corresponding `cpp` file. No inline implementation in the header file apart from the provided functions.
- The usage of ChatGPT and other AI-Related software is strictly forbidden and will be considered as plagiarism.
- Note, UML notation for variable and function signatures are used in this assignment.
- Please note that there is a late deadline which is 1 hour after the initial deadline, but you will lose 20% of your mark.

2 Overview

Copy constructors and assignment operators are special types of functions in C++ which allow programmers to make copies of existing objects. It is important that these functions are properly implemented as the copy constructor is used when parameters are passed by value. The assignment operator is our first example of operator overloading in C++ and is used to create copies of objects. Destructors are also extremely important to implement properly, as these functions ensure that memory is not leaked during the execution of your program.

3 Your Task:

You are required to implement the following class diagram illustrated in Figure 1. Pay close attention to the function signatures as the `h` files will be overwritten, thus failure to comply with the UML, will result in a mark of 0.

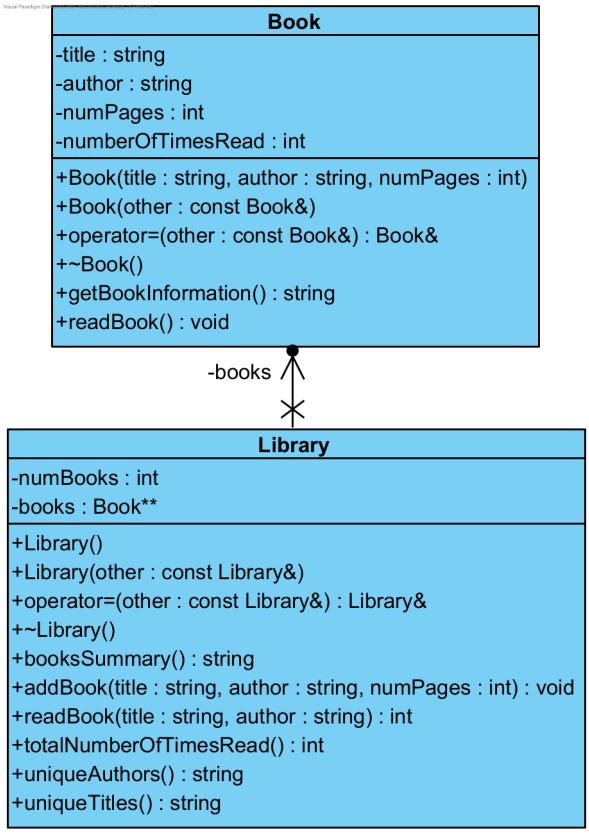

Figure 1: Class diagrams

Note, the importance of the arrows between the classes are not important for COS 110 and will be expanded on in COS 214.

3.1 Book

This is a class which will represent books. Books have a title, author, number of pages, and number of times read.

3.1.1 Members

- -title: string
  - This is the title of the book.
  - You may assume that all titles used contain only letters from the English alphabet and spaces. Thus, no punctuation or other symbols will be used. (You do not need to check for this; you may assume that it is always true).

- -author: string
  - This is the author of the book.
  - You may assume that all authors used contain only letters from the English alphabet and spaces. Thus, no punctuation or other symbols will be used. (You do not need to check for this; you may assume that it is always true).
- -numPages: int
  - This is the number of pages in the book.
- -numberOfTimesRead: int
  - This is the number of times this book has been read.

3.1.2 Functions

- +Book(title: string, author: string, numPages: int)
  - This is a parameterised constructor.
  - Initialize the member variables to the passed-in parameters.
  - Initialize the *numberOfTimesRead* variable to 0.
- +Book(other: const Book&)
  - This is the copy constructor.
  - Since we now have two copies of the book, we want them to be seen as seperate books. Thus the *numberOfTimesRead* variable should **NOT** be copied, it should be set to 0, since the copied book has not been read yet.
  - *Note: Since this copy constructor does not copy all variables, this means that when passing books by value, or when trying to make deep copies, proper copies will not be made. This is the intended behaviour and you should work around this for the rest of this practical.*
- +operator=(other: const Book&): Book&
  - This is the assignment operator.
  - Since we are assigning two books equal to each other, we want to make a proper copy. Thus the *numberOfTimesRead* should also be copied over.
- +~Book()
  - This is the destructor. All dynamic memory should be freed.

- +getBookInformation(): string
  - This should return a string representation of the book.
  - The format is as follows:
    * Start with the *title*.
    * Add a colon.
    * Add the *author*.
    * Inside parentheses, put the *numPages*.
    * Add the *numberOfTimesRead*.
    * Note that you should not add any spaces or newlines.
  - Since all of the member variables are private, if you want to access them in other classes, you will need to call this function and do string manipulation on the result to extract the needed variables.
- +readBook(): void
  - This should increase the *numberOfTimesRead*.

3.2 Library

This is a class which will represent a library. A library is a collection of books, and as such will have a dynamically sized array of books.

3.2.1 Members

- -books: Book**
  - This is a dynamic 1D array of dynamic Books.
  - This array should be properly sized, such that there are no NULL values inside the array.
  - The array should always be of size *numBooks*.
- -numBooks: int
  - This is the number of books in the library.

3.2.2 Functions

- +Library()
  - This is the library constructor. It should intialise the member variables to show that there are 0 books in the library.
- +Library(other: const Library&)
  - This is the copy constructor.
  - This should make a copy of all the books. Since we are copying the books, we want *numberOfTimesRead* of the books to be 0. *Hint: If you call the correct function this will be done automatically.*
- +operator=(other: const Library&) Library&
  - This is the assignment operator.
  - This should assign all the books of this library to the books in the other library. Thus, we also want to copy the *numberOfTimesRead*. *Hint: If you call the correct function this will be done automatically.*
- +~Library()
  - This is the destructor. All dynamic memory should be freed.
- +booksSummary(): string
  - This should return a summary of all of the books in the library.
  - The summary of each book should be the `getBookInformation()` result for each book.
  - The summaries should have a newline between each book. The result should not end with a newline.
- +addBook(title: string, author: string, numPages: int): void
  - This should create a book from the passed-in parameters and add it to the end of the books array.
  - The books array should be properly resized, such that the book can be added at the end.
  - Duplicates *are* allowed.
- +readBook(title: string, author: string): int
  - This should try and read the book with the corresponding title and author.
  - If multiple books have the same title and author, then only read the first book.
  - If the book was found, then read the book, and return the number of pages in the book.
  - If the book was not found, then return -1.

- +totalNumberOfTimesRead(): int
  - This should return the sum of all the times each book in the library has been read.
- +uniqueAuthors(): string
  - This should return a string containing all the authors in the library. Note that each author should appear only once.
  - The authors name should be inside square brackets. You should add all of this together. Thus, there are no spaces or newlines between different authors.
  - Example: If we have four books with authors
    "Sean Macmillan","Daniel van Zyl","Sean Macmillan","Sean Macmillans" then this should return

```
[Sean Macmillan][Daniel van Zyl][Sean Macmillans]
```

- +uniqueTitles(): string
  - This should return a string containing all the titles in the library. Note that each title should appear only once.
  - The title should be inside square brackets. You should add all of this together. Thus, there are no spaces or newlines between different titles.
  - Example: If we have three books with titles
    "Title 1","Title 12","Title 1" then this should return

```
[Title 1][Title 12]
```

4 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-allocates *all* the memory that was allocated.

5 Testing

As testing is a vital skill that all software developers need to know and be able to perform daily, 10% of the assignment marks will be allocated to your testing skills. To do this, you will need to submit a testing main (inside the `main.cpp` file) that will be used to test an Instructor Provided solution. You may add any helper functions to the main.cpp file to aid your testing. In order to determine the coverage of your testing the gcov [1] tool, specifically the following version *gcov (Debian 8.3.0-6) 8.3.0*, will be used. The following set of commands will be used to run gcov:

```
g++ --coverage *.cpp -o main
./main
gcov -f -m -r -j ${files}
```

This will generate output which we will use to determine your testing coverage. The following coverage ratio will be used:

$$\frac{\text{number of lines executed}}{\text{number of source code lines}}$$

We will scale this ration based on class size.

The mark you will receive for the testing coverage task is determined using Table 1:

Coverage ratio range	% of testing mark
0%-5%	0%
5%-20%	20%
20%-40%	40%
40%-60%	60%
60%-80%	80%
80%-100%	100%

Table 1: Mark assignment for testing

Note the top boundary for the Coverage ratio range is not inclusive except for 100%. Also, note that only the functions stipulated in this specification will be considered to determine your testing mark. Remember that your main will be testing the Instructor Provided code and as such, it can

[1] For more information on gcov please see `https://gcc.gnu.org/onlinedocs/gcc/Gcov.html`

only be assumed that the functions outlined in this specification are defined and implemented.

As you will be receiving marks for your testing main, we will also be doing plagiarism checks on your testing main.

6 Implementation Details

- You must implement the functions in the header files exactly as stipulated in this specification. Failure to do so will result in compilation errors on FitchFork.
- You may only use **c++98**.
- You may only utilize the specified libraries. Failure to do so will result in compilation errors on FitchFork.
- Do not include using `namespace std` in any of the files.
- You may only use the following libraries:
  - sstream
  - string

7 Upload Checklist

The following c++ files should be in a zip archive named uXXXXXXXX.zip where XXXXXXXX is your student number:

- `Book.cpp`
- `Library.cpp`
- `main.cpp`
- Any textfiles used by your `main.cpp`
- `testingFramework.h` and `testingFramework.cpp` if you used these files.

The files should be in the root directory of your zip file. In other words, when you open your zip file you should immediately see your files. They should not be inside another folder.

8 Submission

You need to submit your source files on the FitchFork website (`https://ff.cs.up.ac.za/`). All methods need to be implemented (or at least stubbed) before submission. Your code should be able to be compiled with the following command:

```
g++ *.cpp -o main
```

and run with the following command:

```
./main
```

Remember your `h` file will be overwritten, so ensure you do not alter the provided `h` files.
You have 10 submissions and your best mark will be your final mark. Upload your archive to the Practical 3 slot on the FitchFork website. Submit your work before the deadline. **No late submissions will be accepted!**

9 Accessibility

Figure 1 shows the UML of the following classes:

- Book
- Library

The member functions and variables for each class is discussed in Section 4. The following relationships are shown in Figure 1:

- Library has a relation with Book due to the *books* variable.