UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS110: Practical 3

Deadline: Friday 26 August 2022, 11:59

Objectives

The aim of this practical is to learn how to implement operator overloading.

Instructions

Complete the task below. Certain classes have been provided for you in the *files* zip archive of the practical. You have also been given a main file which will test some code functionality, but it is by no means intended to provide extensive test coverage. You are encouraged to edit this file and test your code more thoroughly. Remember to test boundary cases. Upload **only** the given source files with your changes in a zip archive before the deadline. Please comment your name **and** student number in at the top of each file.

Task 1: Copy constructors and aggregation [20]

The task will involve the classes Chessboard and Chesspiece, located in the given files `chessboard.h and chesspiece.cpp`

For this task you will implement the functions provided in the `chessboard.h and chesspiece.h` files inside the `chessboard.cpp and chesspiece.cpp` files.

Implement the following methods according to the given specification:

Chessboard

`Chessboard(Chessboard &obj)`

The copy constructor for Chessboard, should make a copy of all member variables.

Note: make sure to make use of deep copying rather than shallow copy

`Chessboard(int in_rows, int in_columns)`

Should set all member variables to the correct parameters, and correctly initialize all elements in board to null.

`~Chessboard()`

Correctly deallocate all memory assigned to the board member variable.

`void addChesspiece(Chesspiece* piece, int row, int column)`

Takes a Chesspiece pointer, and sets it to the correct row and column in the board element.

`void removeChesspiece(int row, int column)`

Deletes the specified Chesspiece from the board member variable, and reassigns it it's null value.

`Chesspiece& at(int row, int column) const`

Returns an address to a Chesspiece object at the index (row,column) in the board member variable, that may not be modified, but may be accessed.

Chesspiece

`Chesspiece()`

Should simply set the name to " ", and increment the numberOfPieces member variable.

`Chesspiece(Chesspiece &obj)`

The copy constructor for Chesspiece, should make a copy of all member variables and increment the numberOfPieces.

`Chesspiece(string in_name)`

Should simply set the name to the parameter, and increment the numberOfPieces member variable.

`~Chesspiece()`

Decrement the numberOfPieces variable.

`string getName() const`

Simply return the name of the Chesspiece object.

`int getNumberOfPieces() const`

Simply return the number of pieces currently in existence.

Task 2: Operator overloading and friends [25]

Task 2 expands on Task 1, thus one should only continue once 1 is complete. The a **t function** and **copy constructor** are not required to be working in order to continue, but it is advised that they are:

Note: some members were made private within the the **Chesspiece** class, thus you must figure out a way to make those member functions available to the **Chessboard** class without moving them from private. Remember friends are your friends.

Functions

The following functions are required to be implemented:

Chessboard

`void displayBoard()`

"as to the terminal" mam WHAT

This should display the chessboard as to the terminal. The characters displayed should be the following:

- R - rook
- K - knight
- B - bishop
- Q - queen
- X - king
- P - pawn
- * - for any empty space

If the Chesspiece has type set to true, the character should be uppercase. If the Chesspiece has it's type set to false, the character should be lower case. Each character should be followed by a space aside from the very last in a column, and the display should end with a new line. e.g.

R K B * * * * R
P P * * * X * *
* * * Q * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
p p p * * * * *
r k * x * Q * *

`Chessboard operator =(const Chessboard& other)`

The assignment operator should create a deep copy of the right hand side operand, and assign all it's member variables to the left hand side operand.

`Chessboard operator+=(const Chessboard& rhs)`

If there exists an element in the right hand side operand board, where there is an empty slot in the left hand side operand board (Thus that position is null). A copy of that Chesspiece in the right hand side operand should be assigned to that position in the left hand operand.

Chesspiece

`bool getType() const`

Should return the new type member variable in Chesspiece.

`Chesspiece(string in_name, bool in_type)`

The new constructor now also sets the type member variable.

`Chesspiece()`

The default constructor now sets type to true by default.

Submission

You need to submit your source files on the Assignment website `https://ff.cs.up.ac.za/`. All tasks need to be implemented (or at least stubbed) before submission. Place **chesspiece.cpp, chessboard.cpp and makefile** file in a zip or tar/gzip archive (you need to compress your tar archive) named uXXXXXXXX.zip or uXXXXXXXX.tar.gz where XXXXXXXX is your student number. You have 5 days to complete this practical, regardless of which practical session you attend. Upload your archive to the *Practical 3* slot on the Assignment website. Submit your work before the deadline. No late submissions will be accepted.