**BCS 230 Lab – Classes - VideoGame**

***Overview***

Write a VideoGame class and use it in a program. The class definition should be in a .h file. The function definitions for the class should be in a .cpp file. There should be files named VideoGame.h and a VideoGame.cpp. All member variables should be private and all member functions should be public.

***Part 1***

Create a new solution and add the following files to it:

* VideoGame.h – Add to the Headers folder.
* VideoGame.cpp – Add to the Source folder.
* Main.cpp – Add to the Source folder. Will contain the main function.

***Part 2***

Add the following enumerations to VideoGame.h.

enum RatingType { EVERYONE, EVERYONE10PLUS, TEEN, MATURE };

enum SystemType { XBOXONE, PLAYSTATION4, PC};

***Part 3***

Define a VideoGame class (.h file). The list below shows the member variables and their types. All member variables should be decleared as private.

* Title - std::string
* Cost - double
* Rating - RatingType
* System - SystemType

***Part 4***

Write get/set functions for all member variables. These function definitions should be defined in the VideoGame.cpp file.

***Part 5***

Write a default constructor. The default constructor should initialize all member variables to some reasonable default values. For example, a good default value for cost would be 0. This function should be defined in the VideoGame.cpp.

***Part 6***

Write a constructor that takes four parameters. Each parameter should correspond to a member variable. It should initialize the member variables with data from the passed in parameters. This function should be defined in VideoGame.cpp.

***Part 7***

Write a destructor for VideoGame. The destructor is used for clean-up code. The prototype of the destructor is:

~VideoGame();

***Part 8***

Create instances of VideoGame in main. Make sure to call all functions.

***Part 9***

void WriteToScreen()

Write a function named WriteToScreen that will print member variable data to the screen. Just write the data. Do not write any descriptive text. This function should be defined in VideoGame.cpp. Add a call to this function in main.

***Part 10***

void WriteToFile(std::string filename)

Write a function named WriteToFile that will print member variable data to a file. This function should open an output file to the passed in parameter. Just write all member variable data to the output file (similary to WriteToScreen except you are writing to a file). This function should be defined in VideoGame.cpp. Add a call to this function in main.

***Part 11***

void ReadFromScreen()

Write a function named ReadFromScreen that will read member variable data from the screen. Just read in the data. Do not write any descriptive text. This function should be defined in VideoGame.cpp. Add a call to this function in main.

***Part 12***

void ReadFromFile(std::string filename)

Write a function named ReadFromScreen that will read member variable data from the screen. Just read in the data. Do not write any descriptive text. This function should be defined in VideoGame.cpp. Add a call to this function in main.

***Part 13***

Write a copy constructor. The purpose of the copy constructor is to initialize all member variables with data from another instance of a class. Here is the function prototype for the VideoGame copy constructor:

VideoGame(const VideoGame& other);

The code inside the copy constructor should get values from the parameters and copy them into the corresponding member variables of the current instance. For example:

title = other.GetTitle(); // Copies the title from other to the current instance

***Part 14***

Call the VideoGame copy constructor in main. To call the copy constructor you will need to pass in another instance of VideoGame at the declaration. For example,

VideoGame v1;

// Call function to put data into v1

VideoGame v2(v1); // Calls copy constructor