William Dragstrem

Professor Josiah Greenwell

Project Check-In 1

Shapes Project V1.1.0

Project Link:

https://github.com/WMDragstrem/CEN3078-Computer-Security-Project/tree/V-1.1.0

Github Information:

The original project can be found in the “main” branch. The most recent version, as of this document, is in the V-1.1.0 branch. All updates listed occurred in ShapesProject.cpp.

Updates:

1.) *Program now checks to ensure input for coordinates and dimensions is an integer, and outputs an error message if it is not.*

Previously, inputting a non-integer caused the program to go into an infinite output loop.
This issue was resolved with help from a post on https://cplusplus.com/forum/beginner/123439/ to modify the getPoint and getDimension functions.
The given code was modified to accept all integers rather than only whole numbers. It can likely be optimized but for all intents and purposes it is fully functional.

2.) *Program now limits the total number of saved shapes to 256 so as to preserve memory.*

This was done by incrementing an int variable, numLoops each time a new shape is saved to shapesVec. Before the program asks for input in the main menu, it checks that the current

numLoops value is less than the const int MAX_SHAPES (which is currently is assigned 256). If it is less than 256, it functions as previously. If it is >= 256, the program checks if your input is asking to create another shape by checking if it is inside of a vector (shapeInputs) that contains all possible shape inputs (‘l’, ‘r’, or ‘c’). If it isn’t, the program runs as normal (meaning your input was either requesting to print, exit, or was invalid, in which case the default unknown input error message will display). If the input is within the vector, it displays an error message and requests your input once more.

Should you wish to modify the cap, only the const MAX_SHAPES needs to be modified at the top of the program.

3.) *Github is now implemented as part of Backup Policy and access control.*

Future updates are uploaded to separate branches (as is this current update).

Future Updates:

In order of Priority

1.) Write terms of use in the online library. Moderate difficulty
2.) Create a log of all known issues and changes to code with a standard protocol for all logs. Moderate difficulty.
3.) Adding enums so input can be better future-proofed should more options be added. Low difficulty.
4.) Research Dependency injection and how to prevent it with the existing libraries and headers in the project. High difficulty.