For this assignment, I needed to build a stack-based calculator using a monolithic object. In computer science, an object is something that contains functions. To the outside, you can not see what is going on inside the object; all you can see is the inputs going in and the outputs that come out. Imagine an object as a big building with no windows and just two doors. One door is an entrance, and one door is an exit. At the entrance, pieces of plastic are collected, and at the exit, toys are distributed. Inside the building, the plastic is molding into the toys, the toys are tested, etc. This represents the functions going on inside the program. The only difference is that integers (numbers) and characters (operations) are going in, and calculations are coming out…and instead of being happy to see toys, you are banging your head against the wall trying to get your program to run. In the program, the object is the calculator; you feed it integers and characters, and it gives you back numbers after going through functions inside the calculator.

First, we used a struct to house numbers that will be evaluated and characters for operations. This allows us to differentiate between numbers that can be evaluated and operations. Then we use a complex container that can hold both numbers and operations. Then we assign characters for each operation, so that the calculator can actually do something.

But we need to be able to load the stack; we can do this using a constructor or a function. For this program, a function was better because with a function, you can get data at any point. After housing the numbers and characters, assigning the characters with operations, and loading the stack, we need a way to evaluate a full stack. You can do this multiple ways, I'm pretty sure, we did this using vectors.