**Objective**

Practice using pointers with and without needing to use any dynamic memory.

**Main.cpp**

Implement the logic for the following two functions. **You will not be using dynamic memory for any of these functions (no new or delete)**. The first function should return the sum of the datas pointed at by the two int pointers. The second function should not return anything but instead store the result of the sum of a and b into the data of the sum pointer.

**C++**

|  |
| --- |
| int AddNumbersReturnInt(int\* a, int\* b); |
| void AddNumbersReturnVoid(int a, int b, int\* sum); |
|  |

This function should swap the data pointed at by both pointers. Example: a’s data is 5 and b’s data is 10: the function will make a’s data 10 and b’s data 5.

**C++**

|  |
| --- |
| void SwapValues(int\* a, int\* b); |

**TriangleStack.h/.cpp and TriangleHeap.h/.cpp**

Create two separate triangle classes: one where the members are stored on the stack (float) and one where the members are stored on the heap (dynamic float\*). Each triangle should have:  
  
*-private mBase and mHeight members  
  
-two public Set methods to allow main to change each of these members’ values. These should NOT receive float\* for the heap class, just float parameters for the data to set your datas to.  
  
-public method GetArea() that returns the area. The formula for calculating the area is base \* height / 2. Does NOT return a float\* for the heap class, still just a float data result.*

Remember, the dynamic Triangle must fully implement the rule of 3.

**Main.cpp (again)**

Declare two TriangleStack variables and set their members to whatever two different values you want. Then declare a vector<TriangleStack> and add the two triangles to it using its .push\_back() method. Finally, loop through the vector and print each triangle’s area to the screen. Repeat this process again but use TriangleHeap instead.