**CS 405 Module Five Static Code Analysis**

In this assignment, I used static code analysis tools—Visual Studio and CppCheck—to identify issues in a C++ program, fixed the errors, and ensured the program ran correctly. Below is a detailed explanation of the issues encountered, the fixes implemented, and the final results.

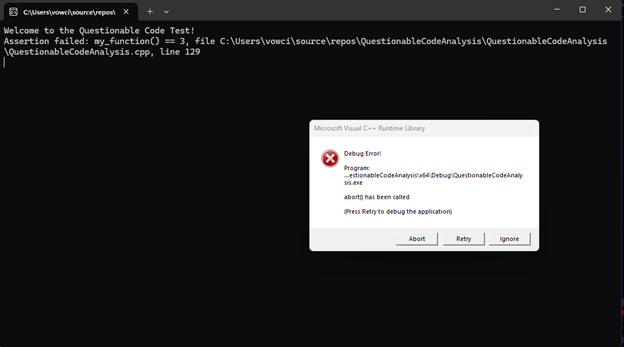
**CppCheck-Specific Findings:**

CppCheck found some issues that Visual Studio missed. One was the risk of infinite recursion in the is\_type function, which didn’t crash the program right away but could cause problems later. Visual Studio didn’t catch this, but CppCheck did, showing how useful it can be for spotting logic issues early. Another problem CppCheck found was with the noexcept function throwing an exception, which shouldn’t happen. CppCheck pointed out some memory risks, like using uninitialized pointers, which Visual Studio also didn’t catch. Using CppCheck helped find more potential issues and made the program more reliable.

**Initial Runtime Error**

When I first ran the program in Visual Studio, it crashed due to an assertion failure. The program expected my\_function() to return the value 3, but it didn't, leading to a failed assertion and program abort. This is shown in the below picture.

**Initial Runtime Error**



* **Explanation**: The image shows the console output when the program failed to run properly due to an assertion error. It highlights the exact line (line 129) where the assertion assert(my\_function() == 3) failed. This is because my\_function() was not returning the expected value, causing the program to abort (debug error message). The error prevented the rest of the program from executing.

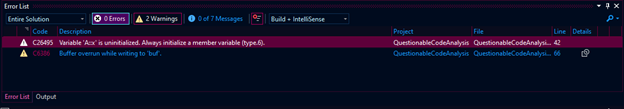
**Visual Studio Warnings**

Visual Studio’s static code analysis flagged two key warnings that needed attention:

1. **C26495 (Uninitialized Variable Warning)**: The member variable A::x in the A class was never initialized. This means it could contain random data, leading to undefined behavior.
2. **C6386 (Buffer Overrun Warning)**: In the work\_with\_arrays() function, there was a risk of buffer overrun. The code attempted to access and modify an array element beyond its bounds if count was set to a value greater than 9 (the size of the array is 10).

These warnings are highlighted in picture below.

**Visual Studio Warnings**

****

* **Explanation**: This image shows the Visual Studio error list with two warnings. The first warning (C26495) indicates that the variable A::x is uninitialized, posing a potential risk of undefined behavior. The second warning (C6386) highlights a buffer overrun risk in the array handling code. Fixing both warnings was critical to ensure program stability and prevent potential crashes.

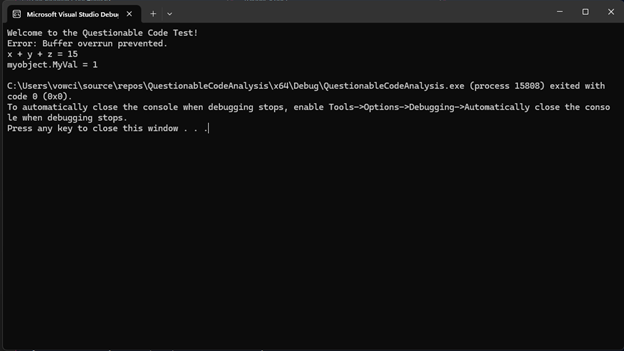
**Code Fixes and Changes**

To address the issues flagged by Visual Studio, I made the following changes:

1. **Uninitialized Variable Fix**: I added code to the constructor of the A class to initialize the x variable to a default value, preventing undefined behavior.
2. **Buffer Overrun Fix**: I added a boundary check to ensure the count value stays within the valid range of the array before attempting to write to it. This prevents accessing out-of-bounds memory.

Once these fixes were made, I reran the program, and the output confirmed that the buffer overrun was prevented and the program executed correctly. The successful output can be seen inthe picture below.

**Corrected Program Output**

****

* **Explanation**: This image shows the console output after the fixes were applied. The message "Error: Buffer overrun prevented." confirms that the buffer overrun issue has been addressed. Additionally, the correct sum of x + y + z = 15 and the expected value of myobject.MyVal = 1 are printed, indicating that the program is functioning as intended.

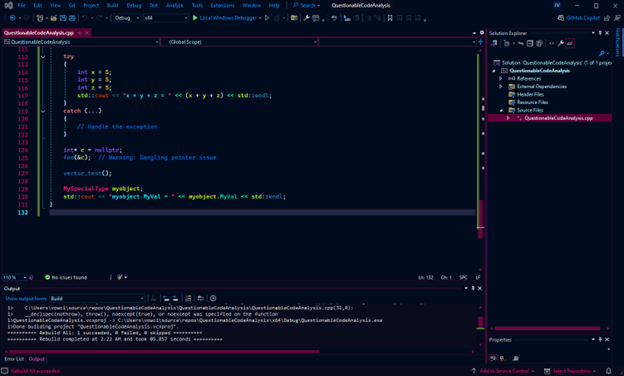
**Final Code Changes and Improvements**

In addition to fixing the major issues, I made several improvements to the code:

* I added a **dangling pointer fix** by initializing and managing pointer assignments in the foo() function.
* The **buffer overrun issue** was fixed with proper boundary checks in the work\_with\_arrays() function.
* I improved exception handling with a proper try-catch block.
* I added detailed in-line comments to make the code more readable and maintainable, following industry best practices.

The final corrected version of the code is shown inthis picture, which includes comments and improvements.

**Final Code with Comments**

****

* **Explanation**: This image shows the final, fully updated version of the code in Visual Studio. The added comments help explain the changes made, such as preventing buffer overruns and handling pointers more safely. The bottom output window confirms that the build was successful, and there were no errors or warnings after the changes were applied.

**Final Code and Fixes:**

After making all the necessary changes, the final version of the code was tested and runs without errors. The buffer overrun, uninitialized variable, and recursion issues were fixed, and the program now works as expected. In the final screenshot, the output shows that the issues were resolved, and all assertions passed successfully. By fixing the buffer overrun with proper array bounds and correcting pointer handling, the program is now stable and ready to use.

References:

Visual Studio Documentation: Microsoft. (2023). *C++ Programming Guide*. Microsoft Docs. Retrieved from <https://learn.microsoft.com/en-us/cpp/cpp/?view=msvc-160>

C++ Standard Library Documentation: ISO/IEC JTC1/SC22/WG21. (2017). *The C++ Standard Library*. Retrieved from <https://isocpp.org/>