Joseph Silva Jr.

SNHU

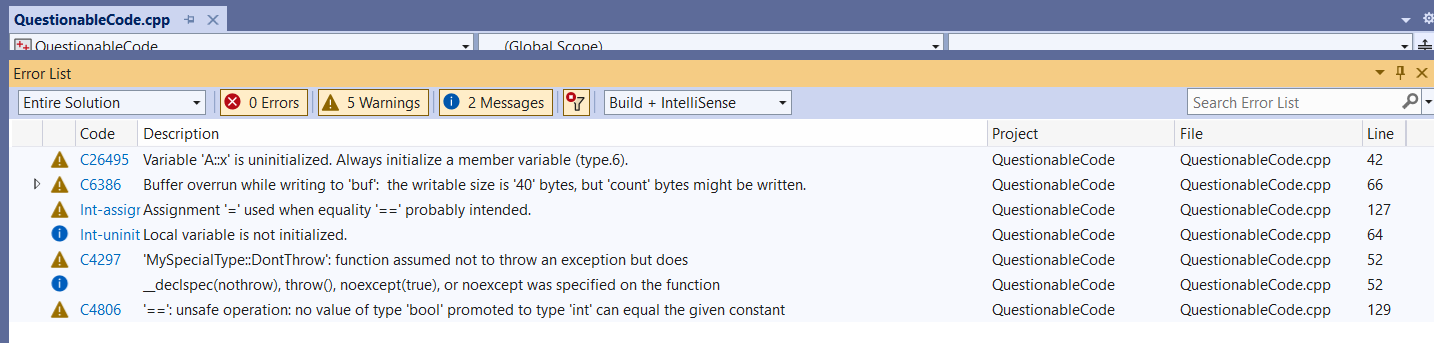
CS 405: 5-3 Activity: Static Code Analysis

11/27/2021

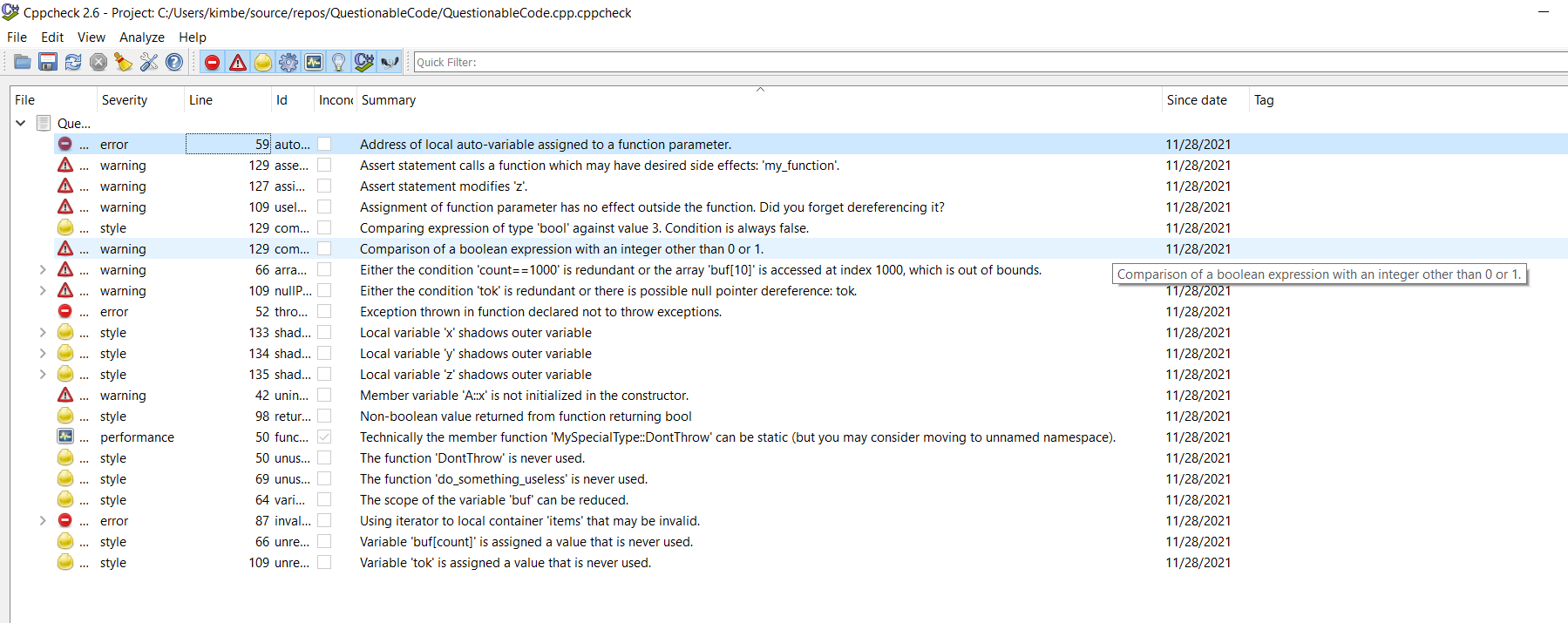
**Original Code Debug Results – Visual Studios**

Graphical user interface, application, Teams

Description automatically generated



**Original Code Debug Results – Cpp check**



**Visual Studios vs cppchecker:**

**Cppchecker:**

1. Line 59: Address of local auto-variable assigned to a function parameter. – Risk. Error severity due to code having some bad behavior when it is executed. The function parameter is assigned the address of a local auto-variable. Local auto-variables are reserved from the stack which is freed when the function ends. The pointer to a local variable is invalid after the function ends.
2. Line 133: local variable ‘x’ shadows outer variable. – Risk. Style severity, pointing out possible mistakes and suggests more defensive program. Name hiding or Shadowing.
3. Line 134: local variable ‘y’ shadows outer variable. – Risk. Style severity, pointing out possible mistakes and suggests more defensive program. Name hiding or Shadowing.
4. Line 135: local variable ‘z’ shadows outer variable. – Risk. Style severity, pointing out possible mistakes and suggests more defensive program. Name hiding or Shadowing.
5. Line 98: Non-boolean value returned from function returning bool. – No Risk. Style severity, pointing out possible mistakes and suggests more defensive program. return a; non-boolean value and a = 1+2;
6. Line 50: Technically the member function ‘MySpecialType::DontThrow’ can be static (but you made consider moving to unnamed space). – No Risk. Performance severity, common knowledge fixes for mistakes. The member function 'MySpecialType::DontThrow' can be made a static function. Making a function static can bring a performance benefit since no 'this' instance is passed to the function. This change should not cause compiler errors, but it does not necessarily make sense conceptually. Think about your design and the task of the function first - is it a function that must not access members of class instances? And maybe it is more appropriate to move this function to a unnamed namespace.
7. Line 50: The function ‘DontThrow’ is never used. – No Risk. Style severity.
8. Line 69: The function ‘do\_something\_useless’ is never used. – No Risk. Style severity, pointing out possible mistakes and suggests more defensive program. Unused Function.
9. Line 87: Using iterator to local container ‘items’ that may be invalid. – Risk. Error severity due to code having some bad behavior when it is executed. Invalid container.
10. Line 66: Variable ‘buff[count]’ is assigned a value that is never used. – No Risk. Style severity, pointing out possible mistakes and suggests more defensive program. Unread variable.
11. Line 109: Variable ‘tok’ is assigned to a value that is never used. – No Risk. Style severity, pointing out possible mistakes and suggests more defensive program. The variable’s value is assigned, but never used and it makes it a dead store
12. Line 109: Assignment of function parameter has no effect outside the function. Did you forget deference it? – No Risk. Warning severity, during execution undefined behavior.
13. Line 109: Either the condition ‘ tok’ is redundant or there is possible null pointer dereference: tok. – No Risk. Warning severity, during execution undefined behavior.

All the warning and messages in visual studios were also discovered in cppchecker.