Dylan Coulter February 11, 2025

[Dylan.coulter@snhu.edu](mailto:Dylan.coulter@snhu.edu)

ID# 1896062

CS405

**Activity 5-3: Static Code Analysis**

**Static Code Analysis Summary:**

To begin this assignment, I created a new project in Visual Studio and pasted the provided code from the rubric into the program and ran it. Visual Studio recognized 2 warnings ‘C26495’ on line 42 and ‘C6386’ on line 66. Visual Studio also had 7 messages. After this I downloaded Cppcheck and followed the instructions within the code to generate a report and save the generated XML file. Cppcheck was a lot more efficient in recognizing issues. Not only did Cppcheck recognize the same warnings that Visual Studio did appearing on line 42 and 66 it also found 3 errors and an additional 5 warnings. Below you will see the breakdown of both programs results, screenshots, and the data from the XML file.

**Visual Studio Results**

**Errors: 0**

**Warnings: 2**

Line: 42

Code: C26495

Description: Variable ‘A::x’ is uninitialized. Always initialize a member variable (type.6).

Line: 66

Code: C6386

Description: Buffer overrun while writing to ‘buf’

**Messages: 7**

Line: 56

Code: VCR003

Description: Function ‘foo’ can be made static

Line: 62

Code: VCR003

Description: Function ‘work\_with\_arrays’ can be made static

Line: 69

Code: VCR003

Description: Function ‘do\_something\_useless’ can be made static

Line: 80

Code: VCR003

Description: Function ‘vector\_test’ can be made static

Line: 95

Code: VCR003

Description: Function ‘my\_function’ can be made static

Line: 106

Code: VCR003

Description: Function ‘foo’ can be made static

Line: 64

Code: int-uninit

Description: Local variable is not initialized

**Cppcheck Results**

**Errors: 3**

Line: 59

Id: autoVariables

CWE: 562

**RISK**

Description: Dangerous assignment - 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. So the pointer to a local variable is invalid after the function ends.

Fix: To make this issue less severe, we can change the way information is transferred within the function.

Line: 52

Id: throwInNoexceptFunction

CWE: 398

**RISK**

Description: Exception thrown in function declared not to throw exceptions.

Fix: To make this issue less severe, we can remove the exception throw statement.

Line: 87

Id: invalidContainer

CWE: 664

**RISK**

Description: Using iterator to local container 'items' that may be invalid.

Fix: To make this issue less severe, we must make sure the iterator continues to be valid.

**Warnings: 7**

Line: 127

Id: assignmentInAssert

CWE: 398

**RISK**

Description: Variable 'z' is modified inside assert statement. Assert statements are removed from release builds so the code inside assert statement is not executed. If the code is needed also in release builds, this is a bug.

Fix: To make this issue less severe, we can adjust the way assert check the code or move the assignment statement outside of the assert.

Line: 129

Id: assertWithSideEffect

CWE: 398

**RISK**

Description: Non-pure function: 'my\_function' is called inside assert statement. Assert statements are removed from release builds so the code inside assert statement is not executed. If the code is needed also in release builds, this is a bug.

Fix: To make this issue less severe, we can move the function call outside of the assert or adjust the way assert checks the code.

Line: 109

Id: uselessAssignmentPtrArg

CWE: 398

**NOT RISK**

Description: Assignment of function parameter has no effect outside the function. Did you forget dereferencing it?

Fix: This issue can be corrected by dereferencing it.

Line: 129

Id: compareBoolExpressionWithInt

**NOT RISK**

CWE: 398

Description: Comparison of a boolean expression with an integer other than 0 or 1.

Fix: To make this issue less severe, we can make changes to the return type from a bool to an int.

Line: 66

Id: arrayIndexOutOfBoundsCond

CWE: 788

**RISK**

Description: Either the condition 'count==1000' is redundant or the array 'buf[10]' is accessed at index 1000, which is out of bounds.

Fix: To make this issue less severe, we can get rid of the array access which is at index 1000, eliminating the condition ‘count == 1000’, or we can increase the array size.

**\*\*\*ALSO CAUGHT By VISUAL STUDIO\*\*\***

Line: 42

Id: uninitMemberVarPrivate

CWE: 398

**RISK**

Description: Member variable 'A::x' is not initialized in the copy constructor. Member variables of native types, pointers, or references are left uninitialized when the class is instantiated. That may cause bugs or undefined behavior.

Fix: To make this issue less severe, we can initialize the member variable A::x.

**\*\*\*ALSO CAUGHT By VISUAL STUDIO\*\*\***

Line: 109

Id: nullPointerRedundantCheck

CWE: 476

**RISK**

Description: Either the condition 'tok' is redundant or there is possible null pointer dereference: tok.

Fix: To make this issue less severe, we can change or get rid of the statement that is cause the null pointer dereference.

**Screenshot Visual Studio**

A computer screen with a black screen

AI-generated content may be incorrect.

**Screenshot Visual Studio Warnings and Errors list**

A screenshot of a computer

AI-generated content may be incorrect.

**Screenshot Cppcheck Results**

**A screenshot of a computer

AI-generated content may be incorrect.**

**XML File Data From Cppcheck**

<results version="2">

<cppcheck version="2.16.0"/>

<errors>

<error id="missingIncludeSystem" severity="information" msg="Include file: &lt;cassert&gt; not found. Please note: Cppcheck does not need standard library headers to get proper results." verbose="Include file: &lt;cassert&gt; not found. Please note: Cppcheck does not need standard library headers to get proper results.">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="4"/>

</error>

<error id="missingIncludeSystem" severity="information" msg="Include file: &lt;iostream&gt; not found. Please note: Cppcheck does not need standard library headers to get proper results." verbose="Include file: &lt;iostream&gt; not found. Please note: Cppcheck does not need standard library headers to get proper results.">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="5"/>

</error>

<error id="missingIncludeSystem" severity="information" msg="Include file: &lt;numeric&gt; not found. Please note: Cppcheck does not need standard library headers to get proper results." verbose="Include file: &lt;numeric&gt; not found. Please note: Cppcheck does not need standard library headers to get proper results.">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="6"/>

</error>

<error id="missingIncludeSystem" severity="information" msg="Include file: &lt;set&gt; not found. Please note: Cppcheck does not need standard library headers to get proper results." verbose="Include file: &lt;set&gt; not found. Please note: Cppcheck does not need standard library headers to get proper results.">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="7"/>

</error>

<error id="missingIncludeSystem" severity="information" msg="Include file: &lt;vector&gt; not found. Please note: Cppcheck does not need standard library headers to get proper results." verbose="Include file: &lt;vector&gt; not found. Please note: Cppcheck does not need standard library headers to get proper results.">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="8"/>

</error>

<error id="assignmentInAssert" severity="warning" msg="Assert statement modifies &#039;z&#039;." verbose="Variable &#039;z&#039; is modified inside assert statement. Assert statements are removed from release builds so the code inside assert statement is not executed. If the code is needed also in release builds, this is a bug." cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="127"/>

</error>

<error id="assertWithSideEffect" severity="warning" msg="Assert statement calls a function which may have desired side effects: &#039;my\_function&#039;." verbose="Non-pure function: &#039;my\_function&#039; is called inside assert statement. Assert statements are removed from release builds so the code inside assert statement is not executed. If the code is needed also in release builds, this is a bug." cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="129"/>

</error>

<error id="uselessAssignmentPtrArg" severity="warning" msg="Assignment of function parameter has no effect outside the function. Did you forget dereferencing it?" verbose="Assignment of function parameter has no effect outside the function. Did you forget dereferencing it?" cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="109"/>

</error>

<error id="autoVariables" severity="error" msg="Address of local auto-variable assigned to a function parameter." verbose="Dangerous assignment - 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. So the pointer to a local variable is invalid after the function ends." cwe="562" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="59"/>

</error>

<error id="compareBoolExpressionWithInt" severity="warning" msg="Comparison of a boolean expression with an integer other than 0 or 1." verbose="Comparison of a boolean expression with an integer other than 0 or 1." cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="129"/>

</error>

<error id="returnNonBoolInBooleanFunction" severity="style" msg="Non-boolean value returned from function returning bool" verbose="Non-boolean value returned from function returning bool" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="98"/>

</error>

<error id="arrayIndexOutOfBoundsCond" severity="warning" msg="Either the condition &#039;count==1000&#039; is redundant or the array &#039;buf[10]&#039; is accessed at index 1000, which is out of bounds." verbose="Either the condition &#039;count==1000&#039; is redundant or the array &#039;buf[10]&#039; is accessed at index 1000, which is out of bounds." cwe="788" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="66" info="Array index out of bounds"/>

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="65" info="Assuming that condition &#039;count==1000&#039; is not redundant"/>

</error>

<error id="uninitMemberVarPrivate" severity="warning" msg="Member variable &#039;A::x&#039; is not initialized in the copy constructor." verbose="Member variable &#039;A::x&#039; is not initialized in the copy constructor. Member variables of native types, pointers, or references are left uninitialized when the class is instantiated. That may cause bugs or undefined behavior." cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="42"/>

</error>

<error id="functionStatic" severity="performance" msg="Technically the member function &#039;MySpecialType::DontThrow&#039; can be static (but you may consider moving to unnamed namespace)." verbose="The member function &#039;MySpecialType::DontThrow&#039; can be made a static function. Making a function static can bring a performance benefit since no &#039;this&#039; 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 an unnamed namespace." inconclusive="true" cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="50"/>

</error>

<error id="functionStatic" severity="performance" msg="Technically the member function &#039;Token::next&#039; can be static (but you may consider moving to unnamed namespace)." verbose="The member function &#039;Token::next&#039; can be made a static function. Making a function static can bring a performance benefit since no &#039;this&#039; 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 an unnamed namespace." inconclusive="true" cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="103"/>

</error>

<error id="throwInNoexceptFunction" severity="error" msg="Exception thrown in function declared not to throw exceptions." verbose="Exception thrown in function declared not to throw exceptions." cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="52"/>

</error>

<error id="nullPointerRedundantCheck" severity="warning" msg="Either the condition &#039;tok&#039; is redundant or there is possible null pointer dereference: tok." verbose="Either the condition &#039;tok&#039; is redundant or there is possible null pointer dereference: tok." cwe="476" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="109" info="Null pointer dereference"/>

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="108" info="Assuming that condition &#039;tok&#039; is not redundant"/>

</error>

<error id="variableScope" severity="style" msg="The scope of the variable &#039;buf&#039; can be reduced." verbose="The scope of the variable &#039;buf&#039; can be reduced. Warning: Be careful when fixing this message, especially when there are inner loops. Here is an example where cppcheck will write that the scope for &#039;i&#039; can be reduced: void f(int x) { int i = 0; if (x) { // it&#039;s safe to move &#039;int i = 0;&#039; here for (int n = 0; n &lt; 10; ++n) { // it is possible but not safe to move &#039;int i = 0;&#039; here do\_something(&amp;i); } } } When you see this message it is always safe to reduce the variable scope 1 level." cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="64"/>

</error>

<error id="shadowVariable" severity="style" msg="Local variable &#039;x&#039; shadows outer variable" verbose="Local variable &#039;x&#039; shadows outer variable" cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="133" info="Shadow variable"/>

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="117" info="Shadowed declaration"/>

</error>

<error id="shadowVariable" severity="style" msg="Local variable &#039;y&#039; shadows outer variable" verbose="Local variable &#039;y&#039; shadows outer variable" cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="134" info="Shadow variable"/>

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="118" info="Shadowed declaration"/>

</error>

<error id="shadowVariable" severity="style" msg="Local variable &#039;z&#039; shadows outer variable" verbose="Local variable &#039;z&#039; shadows outer variable" cwe="398" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="135" info="Shadow variable"/>

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="119" info="Shadowed declaration"/>

</error>

<error id="invalidContainer" severity="error" msg="Using iterator to local container &#039;items&#039; that may be invalid." verbose="Using iterator to local container &#039;items&#039; that may be invalid." inconclusive="true" cwe="664" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="87" info=""/>

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="82" info="Variable created here."/>

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="89" info="After calling &#039;erase&#039;, iterators or references to the container&#039;s data may be invalid ."/>

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="87" info="Assuming condition is true."/>

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="88" info="Assuming condition is true."/>

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="87" info="Iterator to container is created here."/>

</error>

<error id="unusedStructMember" severity="style" msg="class member &#039;A::x&#039; is never used." verbose="class member &#039;A::x&#039; is never used." cwe="563" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="41"/>

</error>

<error id="unreadVariable" severity="style" msg="Variable &#039;buf[count]&#039; is assigned a value that is never used." verbose="Variable &#039;buf[count]&#039; is assigned a value that is never used." cwe="563" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="66"/>

</error>

<error id="unreadVariable" severity="style" msg="Variable &#039;tok&#039; is assigned a value that is never used." verbose="Variable &#039;tok&#039; is assigned a value that is never used." cwe="563" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="109"/>

</error>

<error id="unreadVariable" severity="style" msg="Variable &#039;x&#039; is assigned a value that is never used." verbose="Variable &#039;x&#039; is assigned a value that is never used." cwe="563" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="117"/>

</error>

<error id="unreadVariable" severity="style" msg="Variable &#039;y&#039; is assigned a value that is never used." verbose="Variable &#039;y&#039; is assigned a value that is never used." cwe="563" file0="CS405\_Questionable\_Code\_Analysis.cpp">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="118"/>

</error>

<error id="unusedFunction" severity="style" msg="The function &#039;is\_type&#039; is never used." verbose="The function &#039;is\_type&#039; is never used." cwe="561">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="31"/>

</error>

<error id="unusedFunction" severity="style" msg="The function &#039;DontThrow&#039; is never used." verbose="The function &#039;DontThrow&#039; is never used." cwe="561">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="50"/>

</error>

<error id="unusedFunction" severity="style" msg="The function &#039;do\_something\_useless&#039; is never used." verbose="The function &#039;do\_something\_useless&#039; is never used." cwe="561">

<location file="CS405\_Questionable\_Code\_Analysis.cpp" line="69"/>

</error>

</errors>

</results>