Joseph Les

CS 405 Professor Hodde

Static Code Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Issue Description | Risk | System | Description |
| Default member initializer for non-static data member is a C++11 extension | Not Risk | Visual Studio | This warning indicates that initializing a non-static member variable within the class definition is a feature introduced in C++11. It's a stylistic warning rather than a critical issue. |
| Expected ';' at end of declaration list | Risk | Visual Studio | This error is a syntax issue where a semicolon is missing at the end of a declaration list, causing compilation to fail. It must be corrected to build the project. |
| 'auto' type specifier is a C++11 extension | Not Risk | Visual Studio | This warning indicates that the auto keyword, which deduces the type of a variable automatically, is a feature introduced in C++11. It's a stylistic warning and not a critical issue. |
| Expected ';' at end of declaration | Risk | Visual Studio | A missing semicolon in the code leads to a syntax error, preventing the code from compiling. This error must be fixed for the program to run. |
| Result of comparison of constant 3 with expression of type 'bool' is always false | Risk | Visual Studio | The code is comparing a boolean expression with an integer, which is logically incorrect and may cause unintended behavior. |
| Include file: <cassert> not found | Not Risk | CppCheck | CppCheck indicates that the standard library header file <cassert> is not found. CppCheck does not require standard headers to function correctly, so this is more informational. |
| Include file: <iostream> not found | Not Risk | CppCheck | Similar to the previous entry, this indicates that the <iostream> header is not found. It's an informational message since CppCheck doesn't need these headers. |
| Include file: <numeric> not found | Not Risk | CppCheck | Indicates that <numeric> header is not found, similar to other missing include warnings. It's informational for CppCheck's analysis. |
| Include file: <set> not found | Not Risk | CppCheck | Indicates that the <set> header is not found. It's another informational message as CppCheck doesn't rely on standard library headers. |
| Include file: <vector> not found | Not Risk | CppCheck | Indicates that the <vector> header is not found. Like the other missing includes, it's informational. |
| Assert statement modifies 'z' | Risk | CppCheck | CppCheck found that an assert statement modifies a variable (z). Asserts are removed in release builds, so this can lead to unintended behavior if the code has side effects. |
| Assert statement calls a function which may have desired side effects: 'my\_function' | Risk | CppCheck | CppCheck warns that a function call with potential side effects is within an assert statement. Asserts are not included in release builds, potentially causing the function not to be called. |
| Assignment of function parameter has no effect outside the function | Risk | CppCheck | The code assigns a value to a function parameter, which does not affect the original argument. This is likely a bug since the intended effect may not occur. |
| Address of local auto-variable assigned to a function parameter | Risk | CppCheck | Assigning the address of a local variable to a function parameter can lead to undefined behavior if the variable is accessed after the function ends. |
| Comparison of a boolean expression with an integer other than 0 or 1 | Risk | CppCheck | Comparing a boolean expression with an integer other than 0 or 1 is logically incorrect and can cause unintended behavior. This mirrors the similar Visual Studio warning. |
| Non-boolean value returned from function returning bool | Not Risk | CppCheck | A function declared to return a boolean value is returning a non-boolean value. This is a style issue but can lead to confusion about the function's intended output. |
| Either the condition 'count==1000' is redundant or the array 'buf[10]' is accessed at index 1000, out of bounds | Risk | CppCheck | An out-of-bounds array access or a redundant condition. Accessing out-of-bounds memory is a severe issue that can lead to crashes or undefined behavior. |
| Member variable 'A::x' is not initialized in the copy constructor | Risk | CppCheck | A member variable is left uninitialized in a copy constructor. This can lead to unpredictable behavior when the class instance is copied. |
| Exception thrown in function declared not to throw exceptions | Risk | CppCheck | Throwing an exception in a noexcept function violates the function's contract, potentially causing unexpected termination. |
| Either the condition 'tok' is redundant or there is possible null pointer dereference | Risk | CppCheck | A potential null pointer dereference or a redundant check. Dereferencing null pointers can lead to crashes or undefined behavior. |
| The scope of the variable 'buf' can be reduced | Not Risk | CppCheck | Suggestion to reduce the scope of a variable, which is a stylistic improvement to make the code clearer and possibly more efficient. |
| Local variable 'x' shadows outer variable | Not Risk | CppCheck | A local variable is shadowing another variable with the same name, which can lead to confusion or errors if the wrong variable is accessed. |
| Using iterator to local container 'items' that may be invalid | Risk | CppCheck | Using an iterator to a local container after it may have been invalidated (e.g., by modifying the container) can lead to undefined behavior. |
| Class member 'A::x' is never used | Not Risk | CppCheck | A class member is declared but never used. This is a potential indication of dead code or a design issue. |
| Variable 'buf[count]' is assigned a value that is never used | Not Risk | CppCheck | The variable is assigned a value but not read or used, indicating potentially unnecessary code. |
| Variable 'tok' is assigned a value that is never used | Not Risk | CppCheck | A value assigned to a variable is never used. This might indicate dead code or a logic error where the value should be used. |
| Variable 'x' is assigned a value that is never used | Not Risk | CppCheck | Similar to other unread variables, this indicates that a value assigned to a variable is not used, which could suggest unnecessary code or a logic error. |
| Variable 'y' is assigned a value that is never used | Not Risk | CppCheck | This is another instance where a value assigned to a variable is not used, indicating potential code inefficiency or oversight. |
| The function 'DontThrow' is never used | Not Risk | CppCheck | A function is declared but never used, which could indicate unnecessary code or a need to remove the function. |
| The function 'do\_something\_useless' is never used | Not Risk | CppCheck | Similar to the above, this indicates that the function do\_something\_useless is declared but never called, suggesting potential dead code. |
| Active checkers: 166/802 | Not Risk | CppCheck | This is an informational message indicating the number of active checks performed by CppCheck. It does not indicate a specific issue in the code. |

A screenshot of a computer program

Description automatically generated