6 April 2023

Change Documentation for

Testwell CTC++

Version 10.0.1

Bug Fixes

Support of IAR project names with blank spaces

ctclaunch with IAR specific build option `-iar` did not support quoted project names with spaces.

Multicondition instrumentation for user-defined types in C++

For a Boolean expression in an initialization like

```
ReturnCode retval3 = retval1 && retval2;
```

instrumentation in multicondition mode could lead to uncompilable code if `ReturnCode` provided a converting constructor for `bool`, but not for `int`.
The same happened for assignments and assignment operators available for `bool`, but not for `int`.

Sorting in HTML report

Due to a spelling error (**Sorting.js** vs. **sorting.js**), it was impossible on Linux to use the sorting function for coverage tables in the HTML report.

Missing const-counters in HTML report

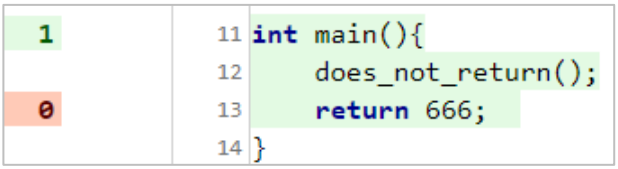
If a decision recognized as compile-time constant by **ctc** was not executed at all, the counter was missing in the HTML report. Now this counter with value 0 is shown in the first column.

ctcreport crash on Linux

An attempt to generate an empty HTML report - by excluding all source files, for example - led to a crash of **ctcreport** on Linux.

Falsely high statement and line coverage

- Unconditional jump statements like `return`, `goto`, `throw`, `break` and their associated lines were falsely reported as executed if a function call before does not return, despite their counter value of zero.



- Code after endless `while` and `for` loops was reported as executed.
- Code after `switch` - `case` blocks could be reported as executed.

Decision and all other higher coverage measures were not affected by these issues.

Missing Coverage

- Code before a range-based `for` loop could be falsely reported as not executed (statement coverage and line highlighting).
- After the definition of a lambda function, statement coverage could be reported too low.
- In a chain of labels and source code without counters, code after the second label could be misinterpreted as not executed (statement coverage and line highlighting).

Problems with command-line parameters

On Windows 7, **ctcreport** could not handle any command-line parameters and aborted with error "Loading the input files failed".

Size limit of option files

Option files used with **ctcreport** do not have a size limit anymore.

Inactive code and variant identification

To distinguish active from inactive code, **ctc** analyzes if preprocessed lines contain code during instrumentation. **ctcreport** uses this information from symbol files in combination with the original source code. Technical implementation issues are now fixed, leading to code misinterpreted as active or inactive or to multiplied representation of header files.

Abortions of **ctcreport**

ctcreport stopped with an error for some language elements represented in the symbol files:

- more than one `catch`-clauses belonging to one `try`-block,
- `__finally` and `__leave` from Microsoft's C language extension for exception handling,
- declarations in `while` and `for` loops,
- MS Visual C++ specific `for each` loops,
- C# `foreach` loops.