Release Notes for Testwell CTC++ v9.0

On November 16th, 2018, a new version of Testwell CTC++ has been released.

The most important enhancements are:

- Support of Objective-C,
- Recognition of compile-time constant expressions,
- Fingerprinting for source code files.

Furthermore, this version includes a fully renewed installer for Windows and all bug fixes since version 8.2.2.

All changes in detail

This is an excerpt from `version.txt` located in the CTC folder of an installation.

General Enhancement in the Testwell CTC++ toolchain:

The way how different versions of instrumented source files are identified is fundamentally changed: Instead of the instrumentation timestamp, a fingerprint is used. In consequence, counter data can be reused more often. See User Guide for details.

Please note that with this change in version 9.0, it is not possible to use symbolfiles or datafiles from older versions.

In the CTC++ preprocessor (ctc):

Enhancement: Objective-C code is now instrumented by ctc. To identify source files with Objective-C code, there is a new parameter for the corresponding name extensions:

```
EXT_OBJC = m, mm, M
```

This setting is delivered in ctc.ini for compilers supporting Objective-C.

Enhancement: The recognition of compile-time constant expressions, e.g.

```
if ( 0 )
```

is highly improved. More complex expressions like

```
if ( 0 + ( (int) 42))
```

are recognized. Consequently, one evaluation (true OR false) during test is sufficient for full structural coverage.

Enhancement: Variable templates are now supported (C++14).

Enhancement: For compilers already supporting functionality of C++20 draft, support is added for constrained function templates. The constraints are not executed and thus not instrumented.

Change: For Java, methods inside enums are now instrumented.

Bug fix: With SKIP_PARAMETER_LIST, some specific functions can be defined whose parameters shall not be instrumented. This setting is not longer ignored when instrumentation mode is multicondition and ternary-? expressions appear in a parameter.

Bug fix: In a switch-case structure, the attributes [[clang::fallthrough]] and [[fallthrough]] do not longer prevent instrumentation.

Bug fix: "Constexpr if"- statements (C++17), for example if constexpr(1<2) are now supported. Previously they caused a syntax error.

Bug fix: In some situations, variables in brackets like

```
        (my_variable)
```

were misinterpreted by ctc as casting. This is now fixed.

Enhancement: GNU C extension allows to use labels as values with the unary && operator, like in

```
        label:
        void *ptr = &&label;
```

This does not longer cause a syntax error.

Enhancement: For the dialect Dynamic-C, costate blocks are now instrumented.

Bug fix: Universal character names like \u00c0 are now supported.

Enhancement: When a certain compiler is configured for ctc, i.e. for the parameter COMMAND in ctc.ini like

```
        COMMAND = ccarm
```

the version with '-orig' attached is implicitely taken into account. Hence it is no longer necessary to list it explicitely like

```
        COMMAND = ccarm, ccarm-orig
```

The '-orig' versions are needed for the ctcwrap-hard setup, for example. This behaviour already existed on Windows and is new on Unix/ Linux.

Bug fix: Functions with the attribute

```
        __attribute((naked))
```

are not longer instrumented, as this resulted in uncompilable code.

Bug fix: An EOF character (255) as character literal does not longer break instrumentation.

Bug fix: When an element of an enumeration inside a typedef was used in an expression, a syntax error could occur. This is fixed.

Enhancement: Added instrumentation support for statement lambdas in Java and C#.

Enhancement: The lock file handling is improved to avoid timeouts and deadlocks.

In the CTC++ postprocessor (ctcpost):

Enhancement: For lamdas, language dependent descriptions are used in the reports:

C++	lambda-[]()
Java	lambda-()->
C#	lambda-()=>
Obj-C	lambda-^()

Change: On Windows platform, the runtime Dll files are renamed to

- ctcwin32.dll
- ctcwin64.dll

with common ordinals unchanged.

Bug fix: In some special cases, a header file included in several source code files was not correctly identified by ctcpost when the extracted report for this file was created. This is fixed.

In the CTC++ XML Merger utility:

Bug fix: The handling of many ternary conditional operators at a single source line is improved. Previously it allocated too much memory.

In the CTC++ HTML converter (ctc2html):

Enhancement: In source code, also CR line breaks are accepted.

Enhancement: When the ecjctc-package is used for Java/ C# development, the generation of the HTML report works now directly, without taking back the instrumentation manually.

Previous version of Testwell CTC++

The release note for v8.2.2 can be found here: https://www.verifysoft.com/ctcpp822.pdf