Tampere (Finland) / Offenburg (Germany), 7 October 2009

Please be informed that a new **CTC++ version 6.5.5** has been released.

This version is primarily a bug fix version. But there are also some enhancements. See the v6.5.5 version details below.

The new version is available on all supported host platforms.

```
Version 6.5.5  (5 October 2009)
-------------------------------
```

This revision 6.5.5 of CTC++ has the following version numbers in its components:

```
  Preprocessor                 6.5.5 (was 6.5.4; seen by -h option)
  Run-time libraries           6.5.5 (was 6.5.4; seen by 'ident'
                                     command applied on the library
                                     in some environments)
  Postprocessor                6.5.5 (was 6.5.4; seen by -h option
                                     and in the listings)
  Header file ctc.h            6.5.5 (was 6.5.4; seen in the ctc.h comments)
  Configuration file ctc.ini   6.5.5 (was 6.5.4; seen in the ctc.ini header)
  CTC++ to HTML Converter      2.5   (unchanged; seen by -h option)
  CTC++ to Excel Converter     1.1   (unchanged; seen by -h option)
  CTC++ Merger utility         1.0   (unchanged; seen by -H option
                                     and in the merged listings)
  ctc2dat receiver utility     2.0   (unchanged; seen by -h option)
```

and the following version numbers in its Windows platform specific components:

```
 CTC++ IDE Integration       3.2   (unchanged, except some enhancements
                                    in the installation script; seen by
                                    clicking the Tw-icon in the dialog
                                    program and selecting "About...")

  Visual Studio 5/6 Integration
                               2.2   (unchanged, except a minor enhancement
                                     in the installation script; seen by
                                     clicking the TW-icon in the dialog
                                     program and selecting "About CTCui...")

  CTC++ Wrapper for Windows    2.5   (was 2.4; seen by -h option)
```

and the following version numbers in its Unix platform (Linux, Solaris, HPUX) specific components:

```
  CTC++ Wrapper for Unix     1.3   (unchanged; seen by -h option)
```

The corrections and enhancements in this version are the following:

In the CTC++ preprocessor (ctc):

- Bug fix: The instrumented file did not compile, if a 'case' or 'default' statement did not appear immediately inside a compound statement forming the switch body. Such "abnormal" 'case' and 'default' statements are no longer instrumented, and a warning message is given. E.g., switch (0) case 0: { ... }

- Bug fix: In certain cases concerning template instantiations or specializations, ctc produced from a correctly written "< ::" the incorrect "<::". This happened if "< ::" was in parentheses between the angle brackets, e.g., T1<sizeof(T2< ::NS::T3>)>. Such cases occurred especially in the Boost C++ libraries.

- Bug fix: In certain cases concerning template instantiations or specializations, ctc "forgot" a space between 'typename' and the following identifier. This happened between the angle brackets, if there was #line (or #pragma) after 'typename', e.g.,
```
   T1<typename
   #line 123
   T2>         // inainstrumented code 'typenameT2'
```
  Such cases occurred especially in the Boost C++ libraries.
- Bug fix: The instrumented file did not compile, if a template instantiation or specialization, e.g., T<sizeof("VeryLongString")>, was over 4096 characters. The limit is now 15000 characters, and an error message is given, if this limit is exceeded.

- Bug fix: If a member function specialization contained quotation marks and CTC++'s function call trace feature was used, the instrumented file did not compile. Further, if the specialization contained newlines, also the generated symbolfile was corrupted.
  For example,
```
   void T<sizeof("ABC")>::memb() { }
   void T<sizeof("ABCD\n\
EFGH\n")>::memb2() { }
```

- Bug fix: It could in certain cases happen that the instrumented file did not compile, if the following kind of GCC extension, a union (or struct or class) inside an expression, was encountered: ...(union { ... })...

- Problem fix: Some C preprocessors (notably Visual C++) may produce from ...?...:M something like ...?...:::NS::T, if the macro M expands to ::NS::T. ctc parsed ':::' according to C++ rules to '::' and ':', but this is non-compilable (as is ':::'). Now this specific case is "corrected" to ':' and '::'.

verifysoft TECHNOLOGY

- Bug fix: The instrumented file did not compile, if none of the functions in a file was instrumented (e.g., #pragma CTC SKIP was used), but nevertheless there was #pragma CTC APPEND in some function. (The same applied to other CTC++ instrumentation pragmas and to the configuration parameter EMBED_FUNCTION_NAME.)

- Change: A ternary expression (?:) is no longer instrumented, if it is inside a static definition, e.g., 'static Type object = M ? 0 : 1;' Previously, such expressions were instrumented in C++ but not in C. Instrumenting ones turns the initialization dynamic (if 'M' was initially a constant expression), which was a problem in some environments.

- Enhancement: The ARM RVCT compiler (v2.2) allows certain function qualifiers after the parameter list, e.g., int f() __softfp {...}. Such functions were not recognized by ctc. Now the following qualifiers __irq, __pure, __softfp, __swi, __swi_indirect, __swi_indirect_r7, and __values_in_regs are allowed, and these functions get recognized and instrumented.

- Documentation fix (ref. v6.5.2 level version.txt): ctc's support to allow ">>" as two closing angle brackets is limited to nested instantiations, e.g., T<T2<T3>> which is taken equivalent to T<T2<T3> >. But the following default template argument use is not allowed by ctc: template<class T1, class T2=A<int>> class X;

- Change: For future CTC++ needs, it is now ensured that when two or more source files are instrumented in a row, they are not given the same timestamps. There will be a difference of at least one second. This is not, however, warranted in parallel builds.

In the CTC++ run-time library:

- New: Added certain sanity checks against corrupted data:
  -- when instrumented source file registers itself to the CTC++ run-time, certain control information must not have nonsense values
  -- when coverage data is written from memory to a datafile, the control information must still be healthy
  -- when a datafile is read (for cumulating the coverage data), the data structures in the datafile must be healthy

  If these checks do not pass, the test run is aborted with an error message.

In the CTC++ postprocessor (ctcpost):

- Enhancement: It is now allowed to select with the -f option which source file(s) the listing options -l or -L will show. Examples:

```
   ctcpost -l -f "*\thefile.cpp" MON.dat
   ctcpost -L -f "*\dir1\*" -f "*\dir2\*" *.sym
```

- Change: The -l and -L listing options now display a description line of all encountered files. Previously, for example in a reinstrumentation situation, descriptions of old/discarded files were not displayed.

- Change: To reduce information bloat, removed some CTCPost notice messages, whose information value is not very significant, e.g., coverage data for a file is summed up from two datafiles. Now only such notice messages are given, which express some "abnormal" input from a symbolfile or datafile, e.g., some coverage data for a file is discarded.

- New: Added certain sanity checks against corrupted data when reading a datafile.

In the CTC++ Wrapper for Windows (ctcwrap):

- Enhancement: Added integration support (-hard option) for cases where the build system invokes the compiler with (absolute) path. For advanced users only! Read more from %ctchome%\Doc\ctcwrap-hard.txt.

In CTC++/Linux, /Solaris, /HPUX delivery packages:

- Change: New model how the installation is suggested to be done. The installation makefile is adjusted correspondingly.

Visual Studio .NET 2003/2005/2008 IDE integration:

- Enhancement: Installation procedure improved. Support for Visual Studio Express edition added. Read more from %ctchome%\vs_integ\version.txt.

Visual Studio 5/6 IDE integration:

- Enhancement: Minor improvement in the installation script ds_integ.bat. Read more from %ctchome%\devstud\version.txt.

General:

- CTC++ User's Guide upgraded to v6.5.5 level.

Version 6.5.4 (26 February 2009)

Information available from http://www.verifysoft.com/ctcpp654.pdf

Verifysoft Technology GmbH – Technologiepark – In der Spöck 10 – D-77656 Offenburg
Geschäftsführer : Klaus Lambertz - Handelsregister Freiburg HRB 472242
Bank : Sparkasse Offenburg/Ortenau BLZ 664 500 50, Konto : 568 719
IBAN : DE30 6645 0050 0000 5687 19   SWIFT-BIC : SOLADES1OFG
Tel +49 781 6392-027 Fax +49 781 6392-029 Email: info@verifysoft.com Internet: http://www.verifysoft.com