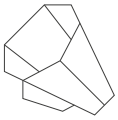


Installation of CppUTest

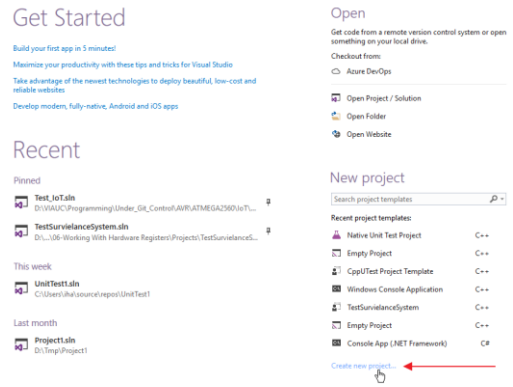
This section will describe how to install CppUTest libraries and corresponding include files in Visual Studio 2017.

All you need is in the 7z file: CppUTest.7z

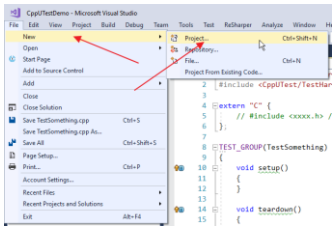
First you must extract the files from the 7z file. In this example it is extracted to *D:\Tmp\CppUTest.*

Create a new Empty Project

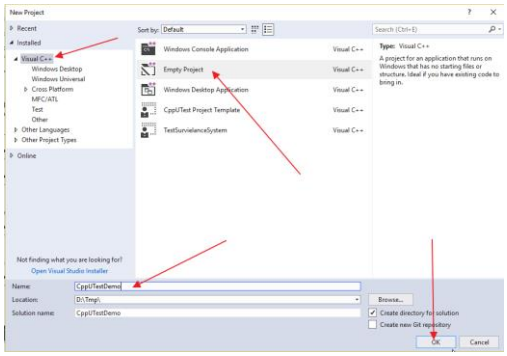
When Visual Studio starts up, then create a new project from the *Getting Started* page:



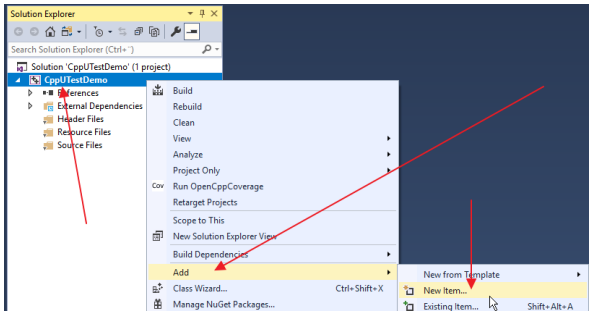
Or use the *File* menu to create a new project:

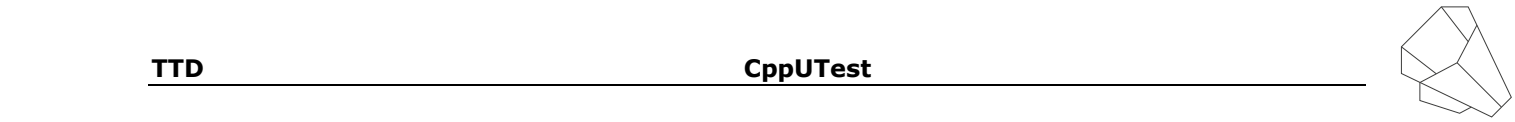


Create a new Visual C++ Empty Project as shown here

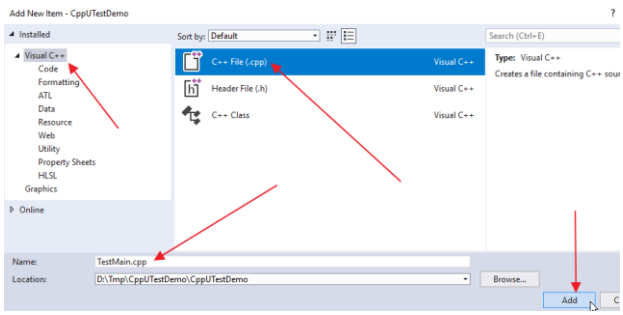


Add the *Main* file for the test project. Right click on the *Project->Add->New Item…*





Give it the name *TestMain.cpp*

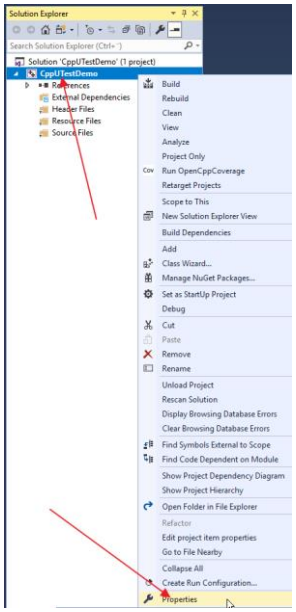


And edit the file to contain the following code:

```
#include "CppUTest/CommandLineTestRunner.h"

int main(int ac, char** av)
{
    return CommandLineTestRunner::RunAllTests(ac, av);
}
```

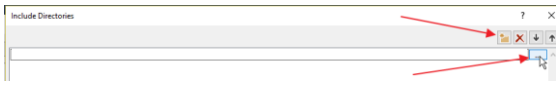
Setting the project up to use CppUTest

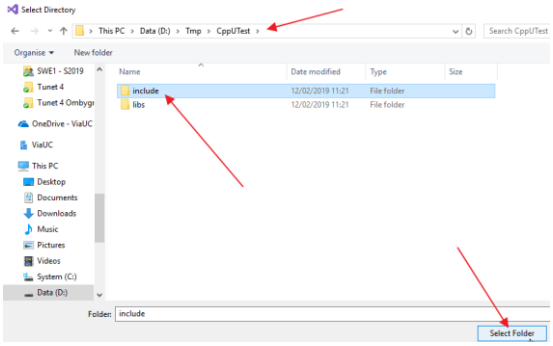
Right click on the project and click on *Properties*:



Then go into *VC++ Directories-> Include Directories* and click the little dropdown arrow and *<Edit…>*

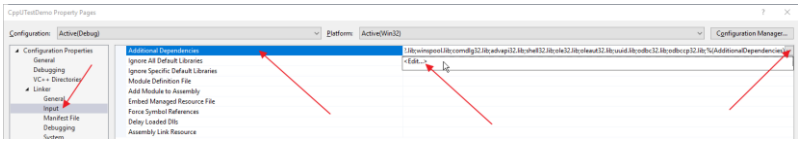
Then browse to the include directory in the extracted files:



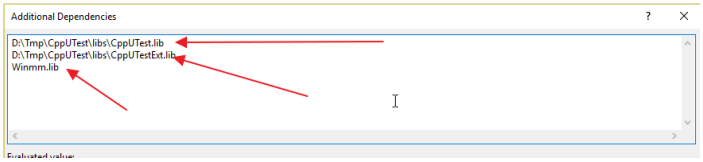


After that we need to setup three dependencies for the linker to the two CppUTest libraries *CppUTest.lib* and *CppUTestExt.lib*.

In the Property Page click on *Linker->Input->Additional Dependencies* and the little dropdown arrow and *<Edit…>*



For some reason, it is not possible to browse to the library files; thus, use Windows file browser to find the location where you have extracted the libraries and copy the path from there.

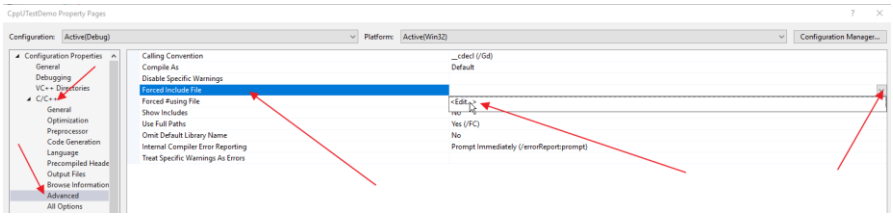


Additionally we need *Winmm.lib* from the Windows libraries. This library contains some time functions needed by *CppUTest*.

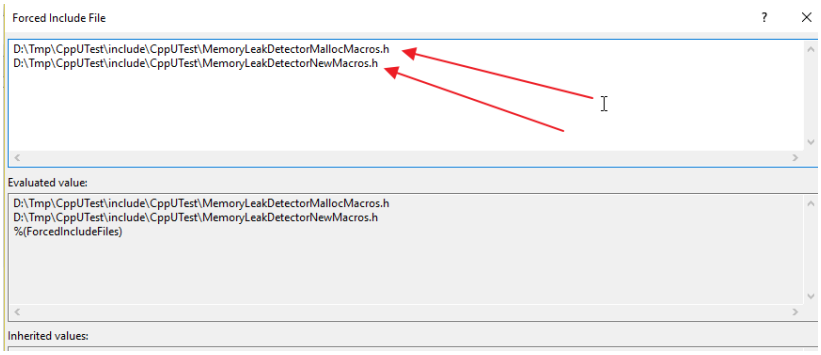
To enable memory leak detection two header files containing some macros must be forced into our compilation.

Note: For some reason this can first be done when at least one source file exist in the project.

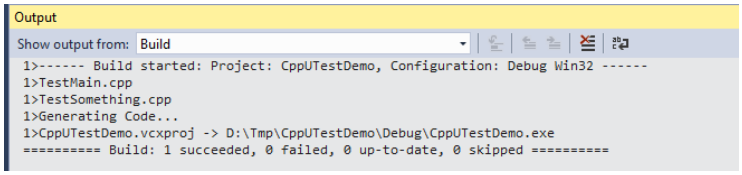
In the Property Page click on *C/C++->Advanced->Forced Include File* and the little dropdown arrow and *<Edit…>*



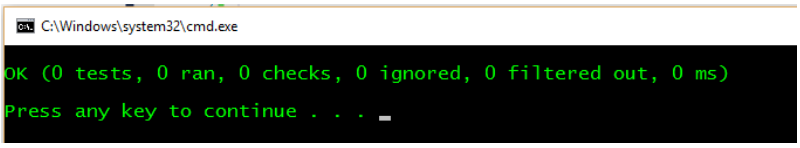
Then insert the two include files *CppUTest/MemoryLeakDetectorMallocMacros.h* and *CppUtest/MemoryLeakDetectorNewMacros.h*



Then everything is setup to start testing. Build the test project to see that you have no errors



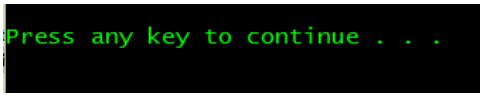
Try to execute it with *CTRL+F5* and the result should be:



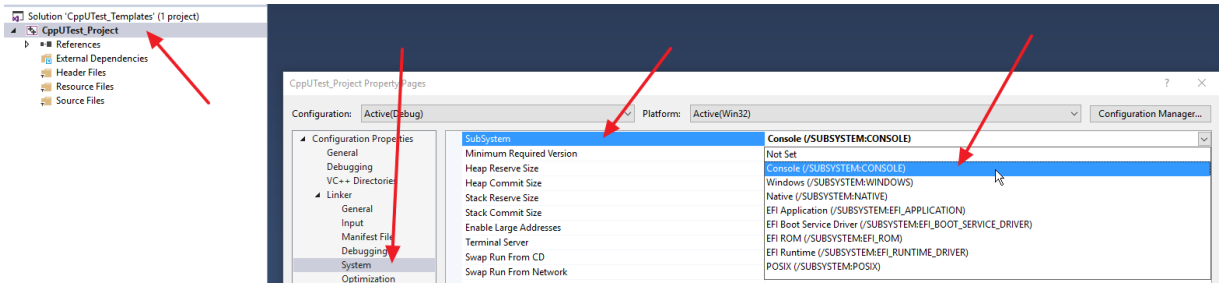
Hint: At this stage: Save the test project as a project template, then it can be reused to create new test projects in the future!!

Keep Console Window Open after Run

To keep the console window open after a run and let it wait for a key press

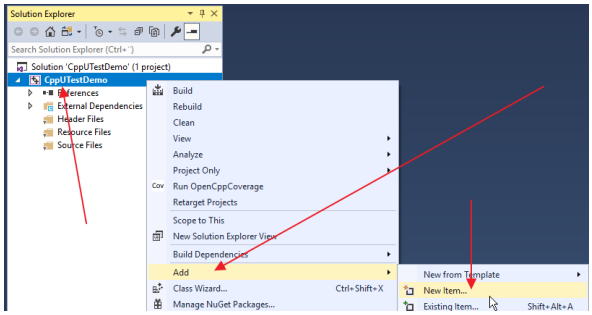


Right click on the project and select *Properties*, under *Linker*, *System*, *SubSystem* Select *Console (/SUBSYSTEM:CONSOLE)* in the dropdown menu

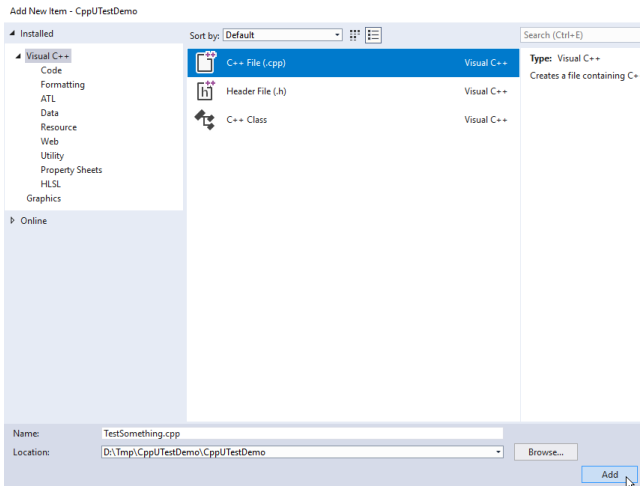


Example of Use

Add a TestUnit (.cpp file) to the project:



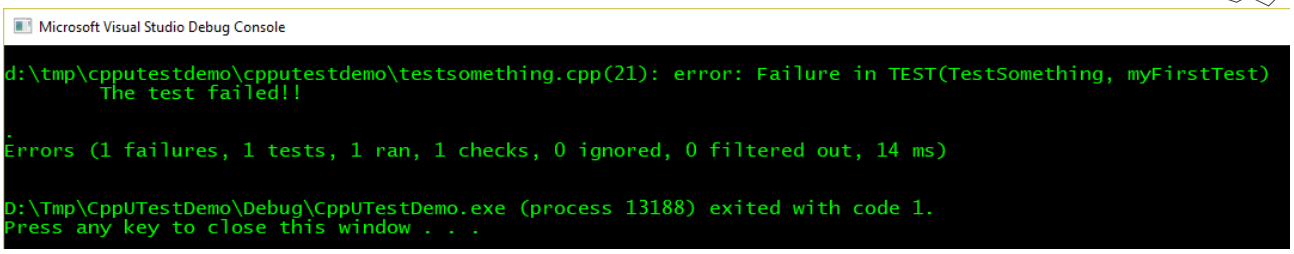
Give it a name, here *TestSomething.cpp*



Add the following to the file

```
#include <CppUTest/TestHarness.h>
#include <CppUTest/TestHarness_c.h>

extern "C" {
	// #include <xxxx.h> // Header files from production code is included here
};

TEST_GROUP(TestSomething)
{
	void setup()
	{
	}

	void teardown()
	{
	}
};

TEST(TestSomething, myFirstTest)
{
	FAIL("The test failed!!");
}
```

Try to execute the program (*Ctrl+F5*) and see the test fails!

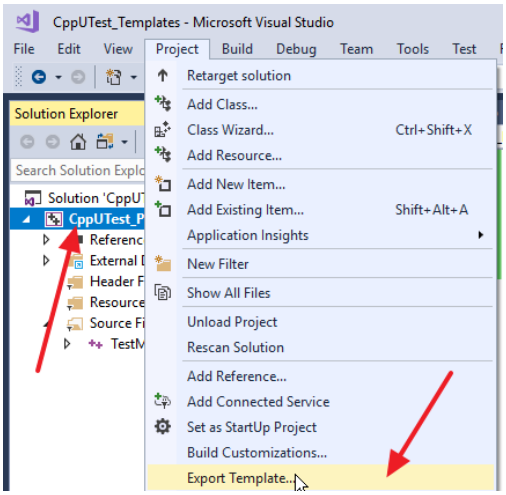


For the use of CppUTest see: http://cpputest.github.io/index.html

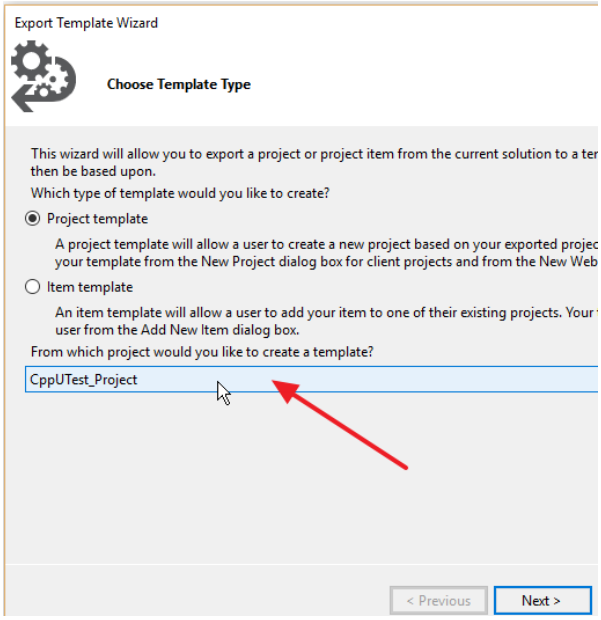
Save a project as a Project Template

All the setting made in the test project can easily be reused if a Project Template is exported of this project.

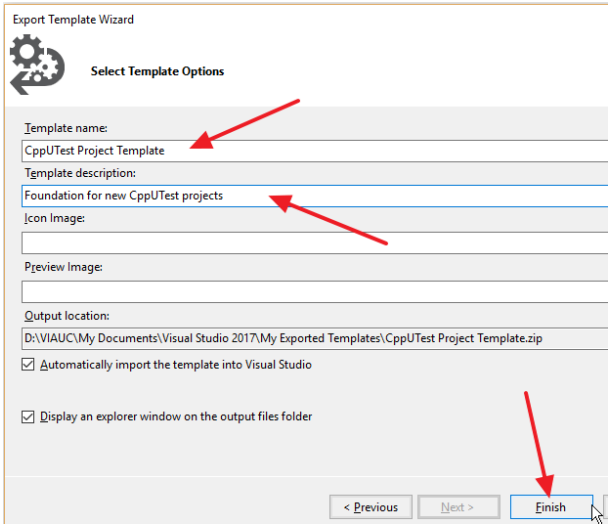
Select *Project* in menu and *Export Template...*



Create the template from the *CppUTest_Project*



Give the template a meaningful name and description

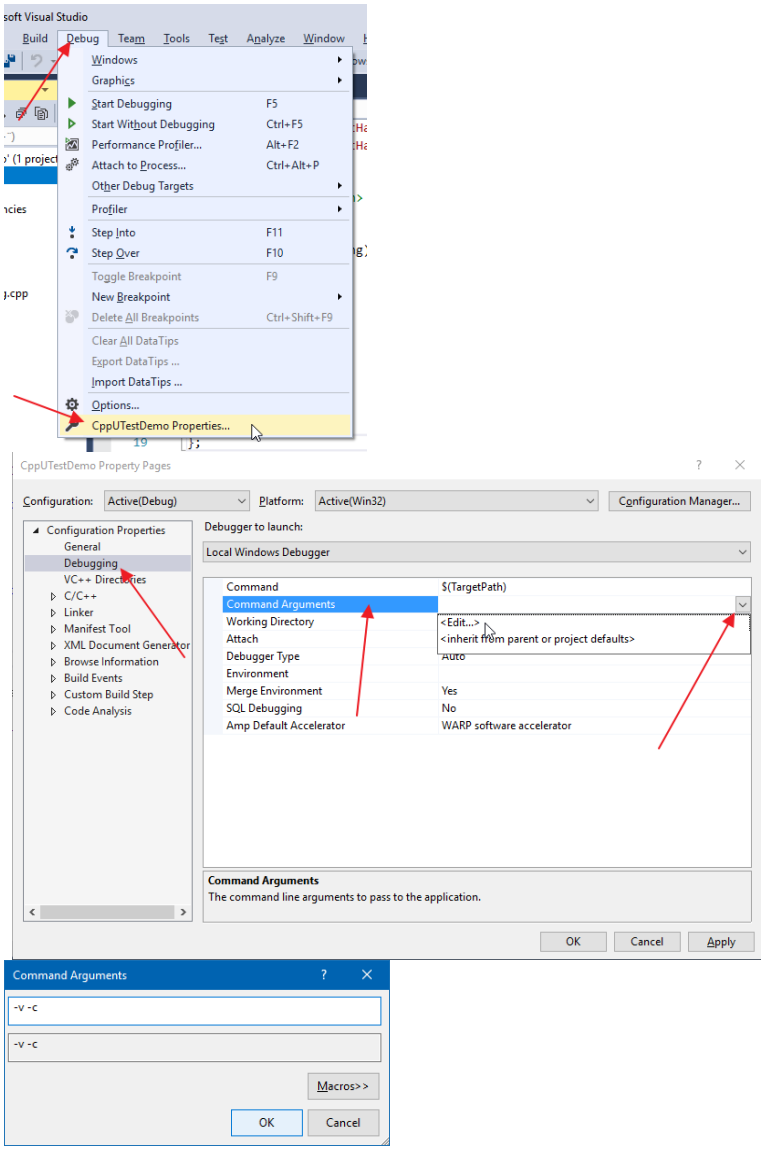


The test project template is now available when you are going to implement another test project.

Enable colour coding and verbose output

A couple of command line switches can be given as argument to the test runner to display each test name as it runs and make the test results clearer. This show passed tests in green text and failed in red.

The setting cannot be included in the template, so it must be configured for each project. Luckily it is easy ☺



Try to run the test again Ctrl+F5 and observe the difference.