Table of Contents

[1 Introduction 2](#_Toc339983857)

[2 Pre-requisites 2](#_Toc339983858)

[3 Walk-Through 2](#_Toc339983859)

[3.1 Visual Studio 2010 Setup 2](#_Toc339983860)

[3.1.1 Recommended Project Layout 2](#_Toc339983861)

[3.1.2 Walk-though 3](#_Toc339983862)

[3.2 Project Setup 9](#_Toc339983863)

[3.2.1 CppUTest Path 9](#_Toc339983864)

[3.2.2 Compiler Options 10](#_Toc339983865)

[3.2.3 Linker Options 14](#_Toc339983866)

[3.2.4 Miscellaneous 17](#_Toc339983867)

[3.3 Writing Your First Test 19](#_Toc339983868)

[4 Summary 22](#_Toc339983869)

# Introduction

Setting up CppUTest and getting your first unit test underway is easy and straight forward. And it should be. However, the steps involved may not be that intuitive. This document therefore provides a walkthrough for those who are experiencing difficulties in getting their first CppUTest unit test under their belt.

Although this walkthrough uses the Visual Studio 2010 IDE, the knowledge presented in setting up the compiler and linker options are generic operations that must be performed regardless of the chosen IDE. Users who are unfamiliar with setting up compiler and linker options may use Section 3.2 as a generic guide to understanding what needs to be configured.

# Pre-requisites

This walk-through assumes that you already have CppUTest & Visual Studio 2010 installed and configured correctly on your local machine that runs Windows 7.

# Walk-Through

## Visual Studio 2010 Setup

### Recommended Project Layout

The following section guides the user in setting up a recommended project layout. It consists of a Visual Studio solution that contains the following 3 projects:

* CppUTest Project

This is the CppUTest project that is included in the CppUTest repository (ie. CPPUTEST\_HOME\(CppUTest.vcxproj). Including this project in your solution, simplifies management of your CppUTest dependency, since updating your CppUTest repository, automatically updates the CppUTest project files which results in up-to-date CppUTest libraries that your project uses.

* Unit Test Project

This is the project that contains all your unit tests. By encapsulating your unit tests in the same project environment facilitates writing, maintaining and running your unit tests that much easier, especially if you are doing Test Driven Development (TDD).

* Your Project

This project contains your project source files (i.e. your production code).

### Walk-though

This section details the steps to set up the recommended project layout.

1. Create an empty solution
2. Open Visual Studio 2010 and navigate to the menu File -> New -> Project.



1. Select Other Project Types -> Visual Studio Solutions -> Blank Solution.



1. Enter a solution name, GameOfLife in this case. Click OK.
2. Add CppUTest to the Solution
3. Navigate to the menu File -> Add -> Existing Project.



1. Browse to the CppUTest project file (i.e. CPPUTEST\_HOME\CppUTest.vcxproj), where CPPUTEST\_HOME is the path to your local CppUTest repository. Click Open.
2. Verify that CppUTest has been added correctly as shown by the screenshot below.



1. Create & Add a Unit Test Project
2. Navigate to File -> New -> Project.



1. Select Visual C++ -> Empty Project.
2. Enter an appropriate name (i.e. UnitTest).
3. Select “Add to solution” and click OK.



1. Verify that UnitTest has been added correctly as shown by the screenshot below.



1. Create and Add Your Project
2. Right-click the solution file in the Solution Explorer.
3. Navigate to Add -> New Project.



1. Select Visual C++ -> Empty Project.
2. Enter a your project name (i.e. GameOfLife).



1. Verify that the project has been added correctly as shown by the screenshot below.



1. Create a Custom Property Sheet
2. Enable the Property Manager view selecting View -> Other Windows -> Property Manager. *(Note: If you are using VS2010 Express, you will first need to enable Expert Settings by selecting Tools -> Setting -> Expert Settings.)*



1. Navigate to the Property Manager tab



1. Right-click the UnitTest project and select “Add New Project Property Sheet…”



1. Enter an appropriate name and path.



1. Verify that the property sheet has been added correctly as shown in the screenshot below.



1. The property sheet will be configured in the next section.

## Project Setup

In this section we will be setting up our project environment, compiler and linker options according to the instructions presented in the “Getting Started” section from <http://cpputest.org/node/20>. We will be using the property sheets created in the preceding section for this.

### CppUTest Path

Before customizing the property sheets, it is required to first a system environment for the CppUTest path. Go to Control Panel -> System and Security -> System -> Advanced system settings -> Environment Variables… -> System variables -> New…

Create the system environment variable CPPUTEST\_HOME and set the value as the path to your CppUTest repository (i.e. C:\CppUTest) as shown in the screen capture below.



*(Note: Please restart VS2010 if it is currently open as the newly created environment variable will only be updated in VS2010 upon a restart.)*

### Compiler Options

The compiler options will be configured via the property sheet.

1. Open CppUTest\_VS2010 property sheet
2. Open (double click) the CppUTest\_VS2010 property sheet from the Property Manager window. 
3. User Macros -> Add Macro
4. Create user macro CPPUTEST\_HOME
5. Create the macro by entering the parameters as displayed in the screenshot below: 
6. Check the checkbox “Set this macro as an environment variable in the build environment”
7. Click OK.
8. Create user macro CPPUTEST\_INCLUDE\_PATHS
9. Go to User Macros -> Add Macro and enter the following values (separated by semicolons):

* $(CPPUTEST\_HOME)\include;
* $(CPPUTEST\_HOME)\include\CppUTestExt\CppUTestGTest;
* $(CPPUTEST\_HOME)\include\CppUTestExt\CppUTestGMock;
* $(CPPUTEST\_HOME)\include\Platforms\VisualCpp;



1. Check the checkbox “Set this macro as an environment variable in the build environment”
2. Click OK.
3. Create user macro CPPUTEST\_FORCED\_INCLUDES
4. Go to User Macros -> Add Macro and enter the following values (separated by semicolons):

* $(CPPUTEST\_HOME)\include\Platforms\VisualCpp\Platform.h
* $(CPPUTEST\_HOME)\include\CppUTest\MemoryLeakDetectorMallocMacros.h

1. Check the checkbox “Set this macro as an environment variable in the build environment”
2. Click OK.
3. Configure the Compiler Options
4. C/C++ -> Additional Include Directories -> Edit…
5. Enter the user macro $(CPPUTEST\_INCLUDE\_PATHS) 
6. C/C++ -> Advanced -> Forced Include File -> Edit…



1. Enter the user macro $(CPPUTEST\_FORCED\_INCLUDES)



### Linker Options

1. Create user macro CPPUTEST\_LIB\_PATHS
2. Go to User Macros -> Add Macro and enter the following the library directory:

* $(CPPUTEST\_HOME)\lib;



1. Check the checkbox “Set this macro as an environment variable in the build environment”
2. Click OK.
3. Create user macro CPPUTEST\_LIB\_DEPENDENCIES
4. Go to User Macros -> Add Macro and enter the following the library:

* CppUTest.lib



1. Check the checkbox “Set this macro as an environment variable in the build environment”
2. Click OK.
3. Configure the Linker Options
4. Linker -> General -> Additional Library Directories -> Edit…



1. Enter the user macro, $CPPUTEST\_LIB\_PATHS



1. Linker -> Input -> Additional Dependencies -> Edit…



1. Enter the user macro, CPPUTEST\_LIB\_DEPENDENCIES.



1. Click OK.

### Miscellaneous

1. Configure project dependencies
2. In the Solution Explorer, right-click the solution -> Properties



1. Project Dependencies -> Project -> UnitTest.
2. Check CppUTest.
3. Check GameOfLife



1. Setup Post Build Event

Set up a post build event to run the unit tests automatically after each build.

1. Navigate to the CppUTest\_VS2010 property sheet.
2. Build Events -> Post-Build Event -> Command Line -> Edit…



1. Enter the macro, $(TargetPath).



1. Click OK.

## Writing Your First Test

1. Create a test main
2. In Solution Explorer, navigate to UnitTest project -> right-click Source Files -> Add -> New Item…



1. Navigate to Code -> C++ File (.cpp) -> Enter name -> Add -> New Item…
2. Typically, the main is in an AllTests.cpp file. Name your file accordingly.
3. Click Add.



1. Open AllTests.cpp and enter the following:

#include "CppUTest/CommandLineTestRunner.h"

IMPORT\_TEST\_GROUP(FirstTestGroup);

int main(int ac, char\*\* av)

{

return CommandLineTestRunner::RunAllTests(ac, av);

}



1. Create the test
2. Create and add a new C++ source file as per 1(a).
3. Name the file, MyFirstTest.
4. Click Add.



1. Open MyFirstTest.cpp and enter the following source code:

#include "CppUTest/TestHarness.h"

TEST\_GROUP(FirstTestGroup)

{

};

TEST(FirstTestGroup, FirstTest)

{

FAIL("Fail me!");

}



1. Build and run your test
2. Press F7 to build the solution. Your unit test should run automatically.
3. Congratulations, your first test should fail.

# Summary

This document was created to help users setup their VS2010 project and get their first CppUTest unit test up and running. Though the information presented here is specific to VS2010, the information presented in Section 3.2 should be portable to other IDEs since setting up the compiler and linker options are essential steps in any IDE.

The property sheet created in Section 3.2 has been added to the scripts\VS2010Templates in the CppUTest distribution. Use it to save some typing and setup time. Just remember to set the CPPUTEST\_HOME system environment variable accordingly.

Please feel free to update, modify and improve this document as necessary.