**C++ Tutorial - Google Test (gtest)**

目录

[Google Unit Test (GTest) 1](#_Toc428968340)

[Step 1. Download Google test (gtest) 3](#_Toc428968341)

[Step 2. Compile gtest into a static library 4](#_Toc428968342)

[Step 3. Create a unit test project 7](#_Toc428968343)

[Step 4. Create a Test Case 9](#_Toc428968344)

# Google Unit Test (GTest)

The Framework of Google C++ Testing is based on xUnit architecture. It is a cross platform system that provides automatic test discovery. In other words, we don't have to enumerate all of the test in our test suite manually. It supports a rich set of assertions such as fatal assertions (**ASSERT\_**), non-fatal assertions (**EXPECT\_**), and death test which checks that a program terminates expectedly.

Here is the [Primer](https://code.google.com/p/googletest/wiki/Primer).

GTest also provides various options for running tests and offers textual and XML report. It also supports a mock object testing framework (Google Mock).

Initially, we have a project calculating a cubic:

**// simplemath.h**

**#include <cmath>**

**double cubic(double d)**

**{**

**return pow(d,3);**

**}**

**// SimpleMath.cpp : Defines the entry point for the console application.**

**#include "simplemath.h"**

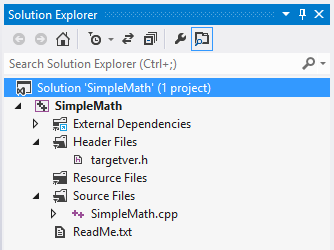
**int main()**

**{**

**cubic(10);**

**return 0;**

**}**

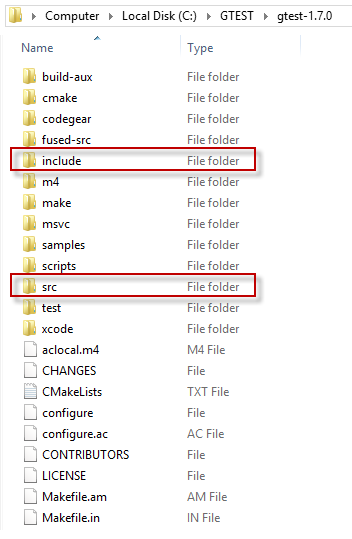
In the following example, we used Visual Studio 2012 with 4 steps:

1. Download Google test
2. Compile gtest into a static library
3. Create a unit test project
4. Make a test case

# Step 1. Download Google test (gtest)

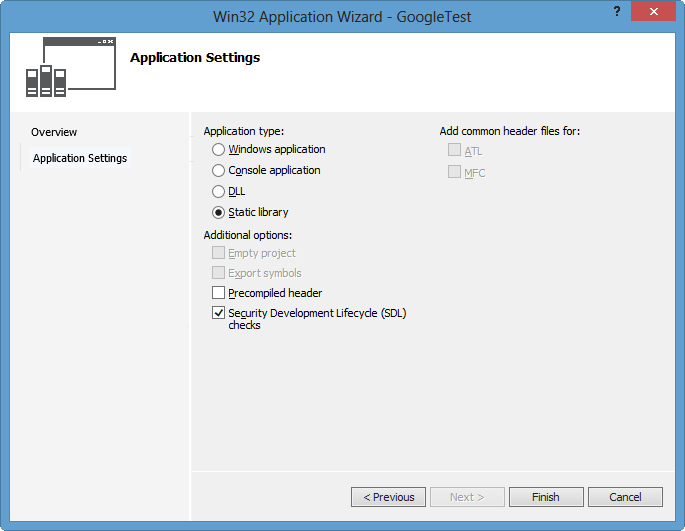
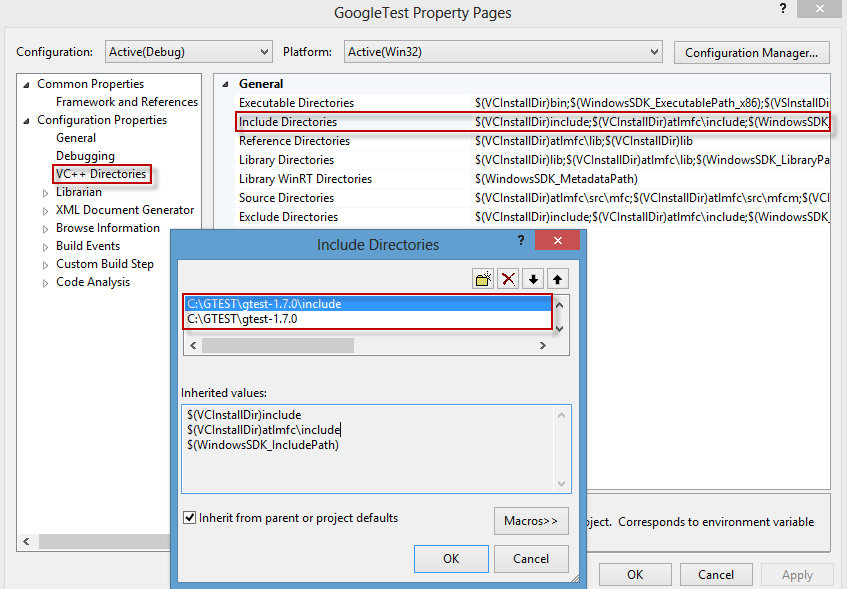
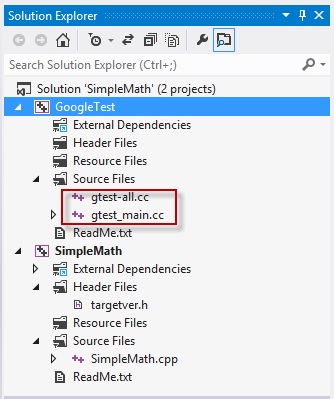
Download the **gtest-1.7.0-rc1.zip** from [Google C++ Unit Test](http://code.google.com/p/googletest/) or from [gtest-1.7.0-rc1.zip](http://www.bogotobogo.com/cplusplus/files/cpptest/gtest-1.7.0-rc1.zip), then extracts it.

Let's look at the **C:\GTEST\gtest-1.7.0** directory to see what files are there.

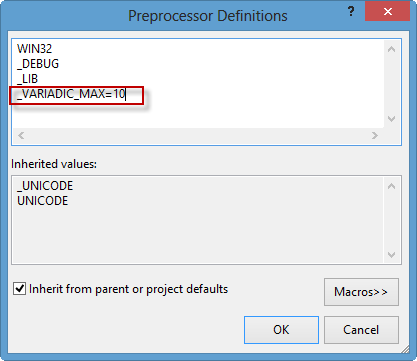
 

The **src** folder has all the gtest source files and later we need to add the **include** directory to the include path.

# Step 2. Compile gtest into a static library

1. Create a new static library project with a name **GoogleTest**.  
   Add->New Project->Win32 Project->Static Library without precompiled header.   
     
   
2. Right click on our new project, **GoogleTest**.   
   On the Properties Pages, add include path:  
   **C:\GTEST\gtest-1.7.0** and **C:\GTEST\gtest-1.7.0\include**.   
     
   
3. Add source files by Add->Existing Item...   
   **C:\GTEST\gtest-1.7.0\src\gtest\_all.cc**   
   and **C:\GTEST\gtest-1.7.0\src\gtest\_main.cc**.   
     
   
4. Build **GoogleTest** into static library.  
   In the build process, we may have some errors related to class template:

VC++ 2012 does not (and will never) support variadic templates; consequently, its standard library implementation attempts to fake them using preprocessor-generated overloads and specializations. The number of faux variadic template parameters defaults to 5 - the problem is that gtest is trying to instantiate std::tuple<> with as many as 10 template arguments. - [Google Test in Visual Studio 2012](http://stackoverflow.com/questions/12558327/google-test-in-visual-studio-2012).

So, we need to set **\_VARIADIC\_MAX=10** for Preprocessor Definitions under C/C++.   
  


Now, build it again:

**1>------ Rebuild All started: Project: GoogleTest, Configuration: Debug Win32 ------**

**1> gtest\_main.cc**

**1> gtest-all.cc**

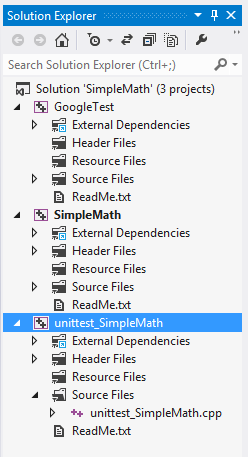
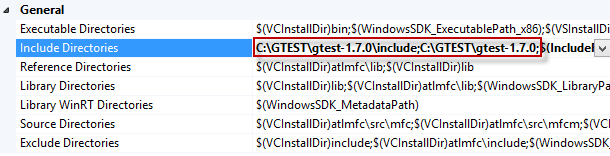
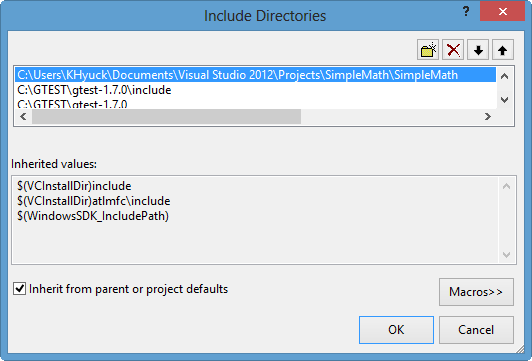
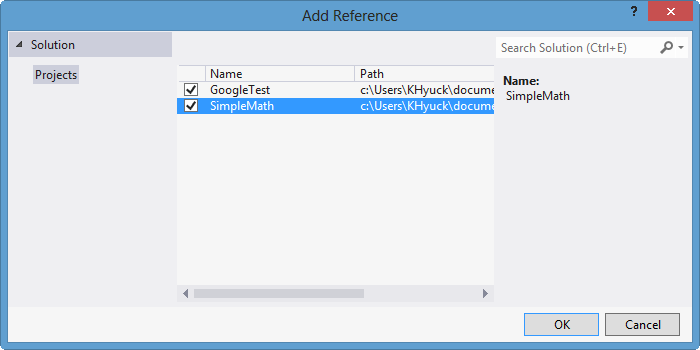
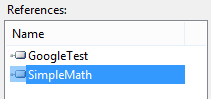
**1> Generating Code...**

**1> GoogleTest.vcxproj -> c:\users\khyuck\documents\visual studio 2012\Projects\SimpleMath\Debug\GoogleTest.lib**

**========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========**

# Step 3. Create a unit test project

Now, it's time to create a unit test project.

1. Right click on Solution->Add->New Project with a name **unittest\_SimpleMath** as a Win32 Console. We've just added 3rd prodject to our solution:   
     
   
2. We need to add the two paths as we've done in Step 2:   
   Right click on our new project, **unittest\_SimpleMath**.   
   On the Properties Pages, add include path:  
   **C:\GTEST\gtest-1.7.0** and **C:\GTEST\gtest-1.7.0\include**.   
     
   
3. This project needs additional path to the initial project (**SimpleMath**) which we want to be tested.   
     
   
4. Let's add new references (**GoogleTest** and **SimpleMath**) to **unittest\_SimpleMath**.  
   Right click on **unittest\_SimpleMath**->References...  
   Under Property Pages->Add New References...   
     
      
     
   
5. Great. Our Unit Test project has been set up.  
   Final step will be making a test case.

# Step 4. Create a Test Case

Now, we need to create a test case.   
Type in the following lines of code:

**// unittest\_SimpleMath.cpp : Defines the entry point for the console application.**

**#include "gtest/gtest.h"**

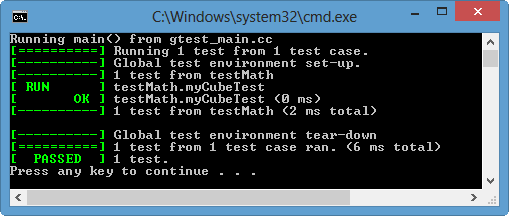
**#include "simplemath.h"**

**TEST(testMath, myCubeTest)**

**{**

**EXPECT\_EQ(1000, cubic(10));**

**}**

Here, we're testing the **cubic()** function we wrote before, and it compares the output of 10^3 with 1000 using macro **EXPECT\_EQ**.  
If we run the **unittest\_SimpleMath**, we get the test result:   
  
   
  
OK!   
We passed our first Google Test!