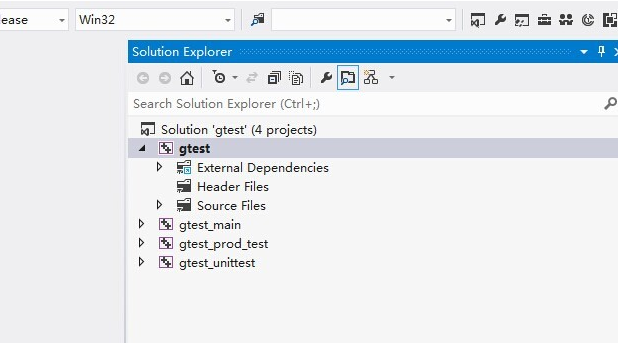
**1.下载**

首先下载google test  
<https://github.com/google/googletest/releases>

**2.编译**



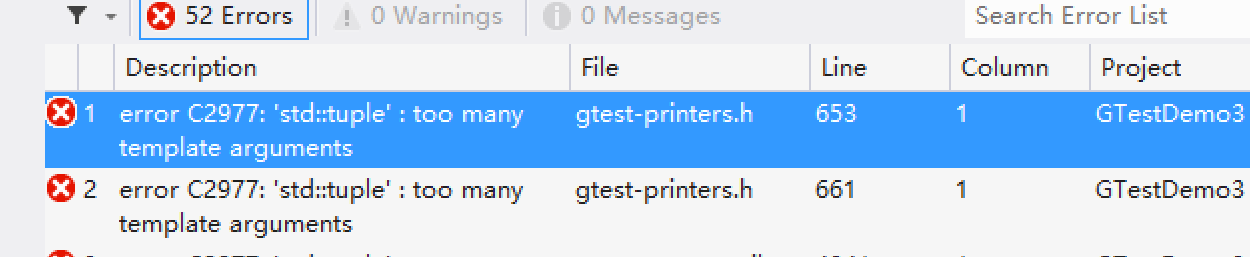
可以看到源代码文件夹里，有适应各种[操作系统](http://lib.csdn.net/base/operatingsystem" \t "_blank" \o "操作系统知识库)的编译脚本或者工程文件，比如cmake,make,msvc， msvc文件夹里就是一个支持visual studio的solution文件

打开gtest.sln

在VS中导入msvc/gtest.sln,然后重新编译，debug、release两个模式都要编译,编译后，在msvc/gtest/下有debug、release两个文件夹。

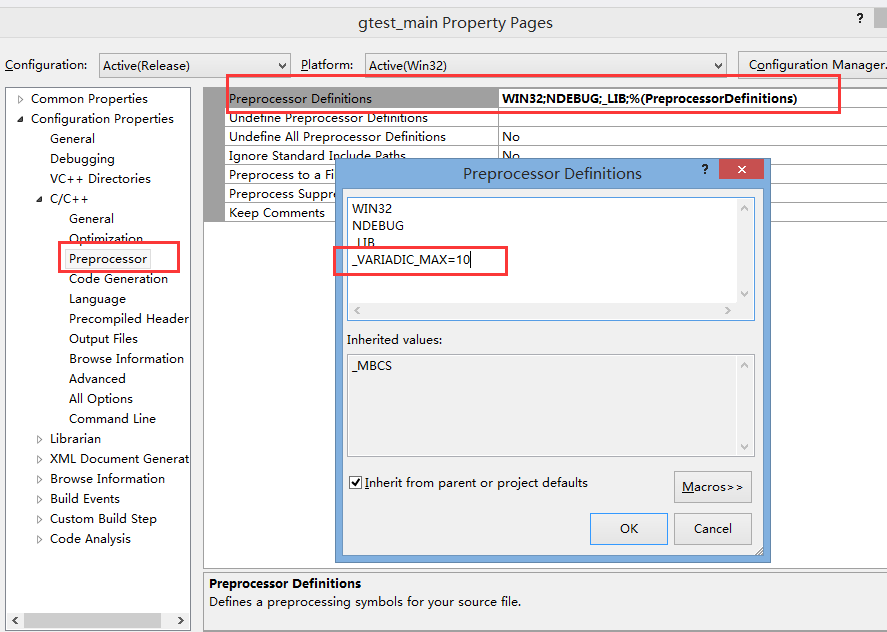
msvc\gtest\Debug - Debug方案下的二进制文件：gtest**d**.lib、gtest\_main**d**.lib（注意主文件名的d后缀）  
msvc\gtest\Release - Release方案下的二进制文件：gtest.lib、gtest\_main.lib

然后编译，会发现报错

error C2977 "std::tuple" too many template arguments

解决方法:在每个工程（project）的属性中的C/C++ -> Preprocessor （预处理）--> preprocessor definitions （预处理定义）中增加

\_VARIADIC\_MAX=10



为了方便使用，配置环境变量

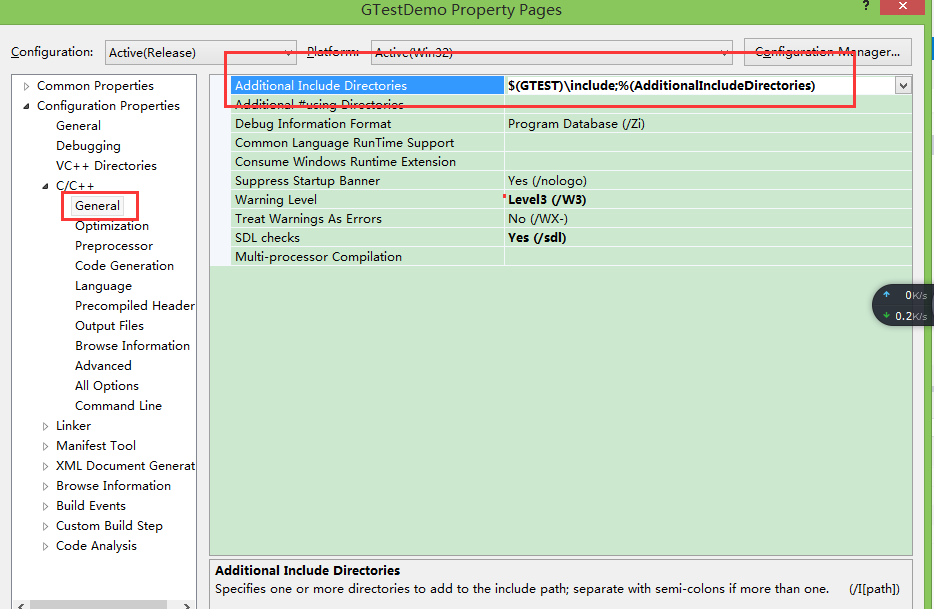
GTEST =D:\demo\googletest-release-1.8.0\googletest

#### 3.第一个Demo

新建一个Win32 Console Application，GTestDemo。接着设置工程属性，总结如下：

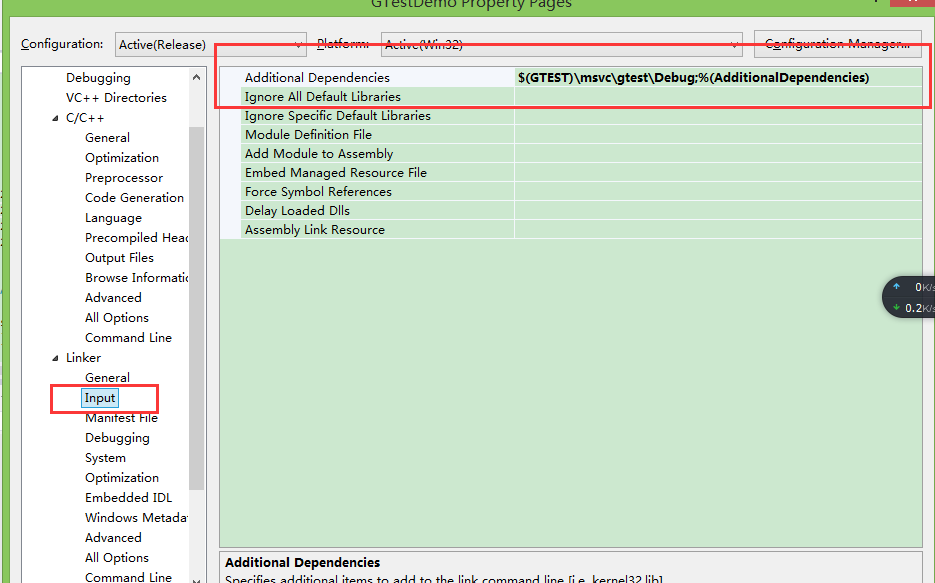
1. 设置gtest头文件路径

$(GTEST)\include



1. 设置gtest.lib路径

$(GTEST)\msvc\gtest\Debug\gtestd.lib



工程设置后了后，我们来编写一个最简单测试案例试试，我们先来写一个被测试函数：

sample.h

int fun(int a, int b);

sample.cpp

#include"stdafx.h"

#include"sample.h"

int fun(int a, int b)

{

return (a-b);

}

GtestDemo.cpp

#include "stdafx.h"

#include "gtest\gtest.h"

#include "sample.h"

TEST(fun, case1)

{

EXPECT\_LT(-2, fun(1, 2));

EXPECT\_EQ(-1, fun(1, 2));

ASSERT\_LT(-2, fun(1, 2));

ASSERT\_EQ(-1, fun(1, 2));

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

testing::InitGoogleTest(&argc, argv);

return RUN\_ALL\_TESTS();

}

代码说明：

testing::InitGoogleTest(&argc, argv)：

gtest的测试案例允许接收一系列的命令行参数，该方法将命令行参数传递给gtest，进行一些初始化操作

RUN\_ALL\_TESTS()：运行所有测试案例。

TEST这个宏，它有两个参数，官方的对这两个参数的解释为：[TestCaseName，TestName]

//EXPECT\_EQ是比较两个值是否相等

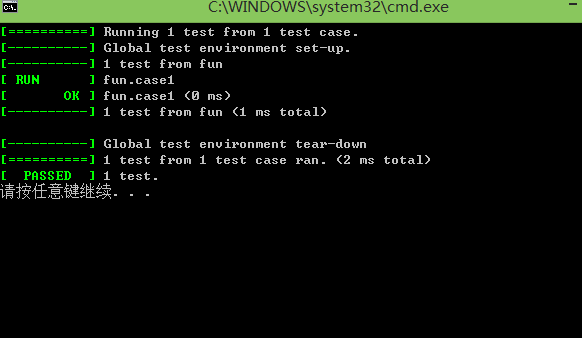
EXPECT\_NE

1. EXPECT\_\* 失败时，案例继续往下执行。

2. ASSERT\_\* 失败时，直接在当前函数中返回，当前函数中ASSERT\_\*后面的语句将不会执行。

Ctrl+F5直接运行

布尔值检查



#### 4.基础断言

测试正确错误.

|  |  |  |
| --- | --- | --- |
| **Fatal assertion** | **Nonfatal assertion** | **Verifies** |
| ASSERT\_TRUE(*condition*); | EXPECT\_TRUE(*condition*); | *condition* is true |
| ASSERT\_FALSE(*condition*); | EXPECT\_FALSE(*condition*); | *condition* is false |

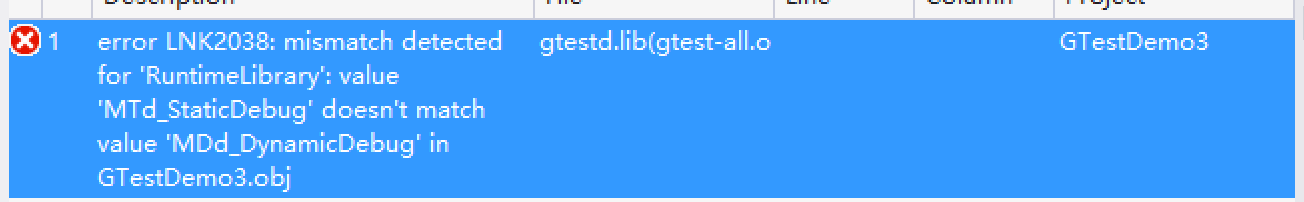
二元比较

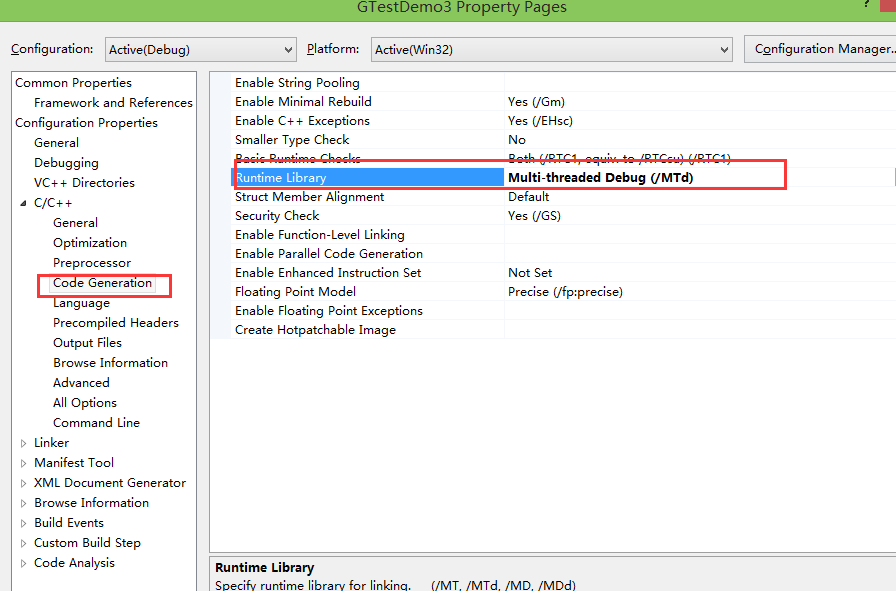
|  |  |  |
| --- | --- | --- |
| **Fatal assertion** | **Nonfatal assertion** | **Verifies** |
| ASSERT\_EQ(*expected*, *actual*); | EXPECT\_EQ(*expected*, *actual*); | *expected* == *actual* |
| ASSERT\_NE(*val1*, *val2*); | EXPECT\_NE(*val1*, *val2*); | *val1* != *val2* |
| ASSERT\_LT(*val1*, *val2*); | EXPECT\_LT(*val1*, *val2*); | *val1* < *val2* |
| ASSERT\_LE(*val1*, *val2*); | EXPECT\_LE(*val1*, *val2*); | *val1* <= *val2* |
| ASSERT\_GT(*val1*, *val2*); | EXPECT\_GT(*val1*, *val2*); | *val1* > *val2* |
| ASSERT\_GE(*val1*, *val2*); | EXPECT\_GE(*val1*, *val2*); | *val1* >= *val2* |

字符串比较

|  |  |  |  |
| --- | --- | --- | --- |
| **Fatal assertion** | | **Nonfatal assertion** | **Verifies** |
| ASSERT\_STREQ(*expected\_str*, *actual\_str*); | EXPECT\_STREQ(*expected\_str*, *actual\_str*); | | the two C strings have the same content |
| ASSERT\_STRNE(*str1*, *str2*); | EXPECT\_STRNE(*str1*, *str2*); | | the two C strings have different content |
| ASSERT\_STRCASEEQ(*expected\_str*, *actual\_str*); | EXPECT\_STRCASEEQ(*expected\_str*, *actual\_str*); | | the two C strings have the same content, ignoring case |
| ASSERT\_STRCASENE(*str1*, *str2*); | EXPECT\_STRCASENE(*str1*, *str2*); | | the two C strings have different content, ignoring case |

备注





因为版本不同

新建一个C++“Win32控制台应用程序”，然后右键项目，打开项目的属性窗口，依次选择“配置属性”——“VC++目录”，然后分别将“包含目录”和“库目录”中的路径选择为刚才我们将头文件和库文件放置的路径。

**$(GTEST)\include**

**$(GTEST)\msvc\gtest\Debug**

然后将属性窗口左上角的“配置”选项选为“Release”，然后和上一步的操作一样，进行头文件和库文件路径的配置，这一步是配置程序发布版的路径。

接下来在“配置属性”——“链接器”——“输入”中配置程序的附加依赖项，也就是库文件的名称，同样要在“Debug”和“Release”两种条件下配置，这样就完成了所有的配置。