

android studio教程

一、建新项目

重新启动Android Studio，在“Welcome...”窗口中选“Start a new Android Studio project”，按提示一步一步“Next”。

首先是起名：

Application name（应用名）：就是你要做的程序的名称，也将是项目名称。

Company Domain（公司名）：一般的格式是“类别.部门.公司”，这个练习只使用了“类别.部门”，填的是“exercise.myself”，意思就是“我自己.练习”。

你修改上面两项，软件就会自动更新下面的一项：

Package name（包名）：这很重要，是你的程序的标识，C/C++代码会引用它。你不必现在记住它，因为写代码时随时可以在java代码中查到它。

然后是项目路径：

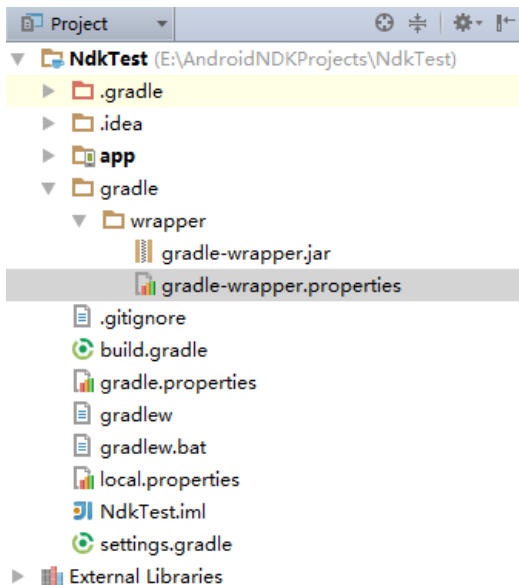
Project location（项目路径）：就是说这个项目的文件装在哪个文件夹里，这个可以随便设，只要自己记得住。

后面有一步选模板，最好选“Empty Activity”，其他模板会给你预备一堆没用的组件。

二、改gradle代码

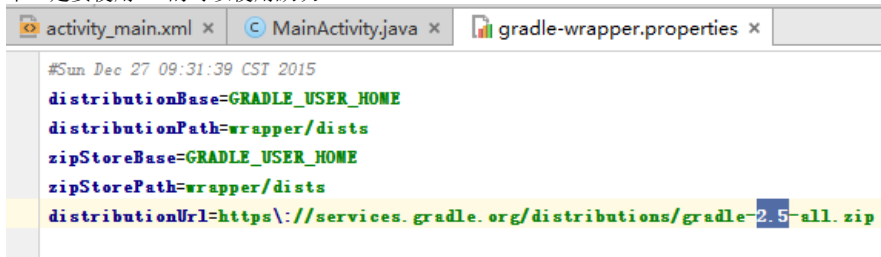
对于不需要C/C++的项目，这一步完全可以跳过。但是不需要C/C++，也不用看这篇文章了。

打开项目后，在界面左上方选“Project”，再把目录展开成这样：



上图中选中的“gradle-wrapper.properties”，是马上要修改的。它在目录中的位置，可表示为“gradle/wrapper/gradle-wrapper.properties”。

不一定要使用2.5的可以使用默认



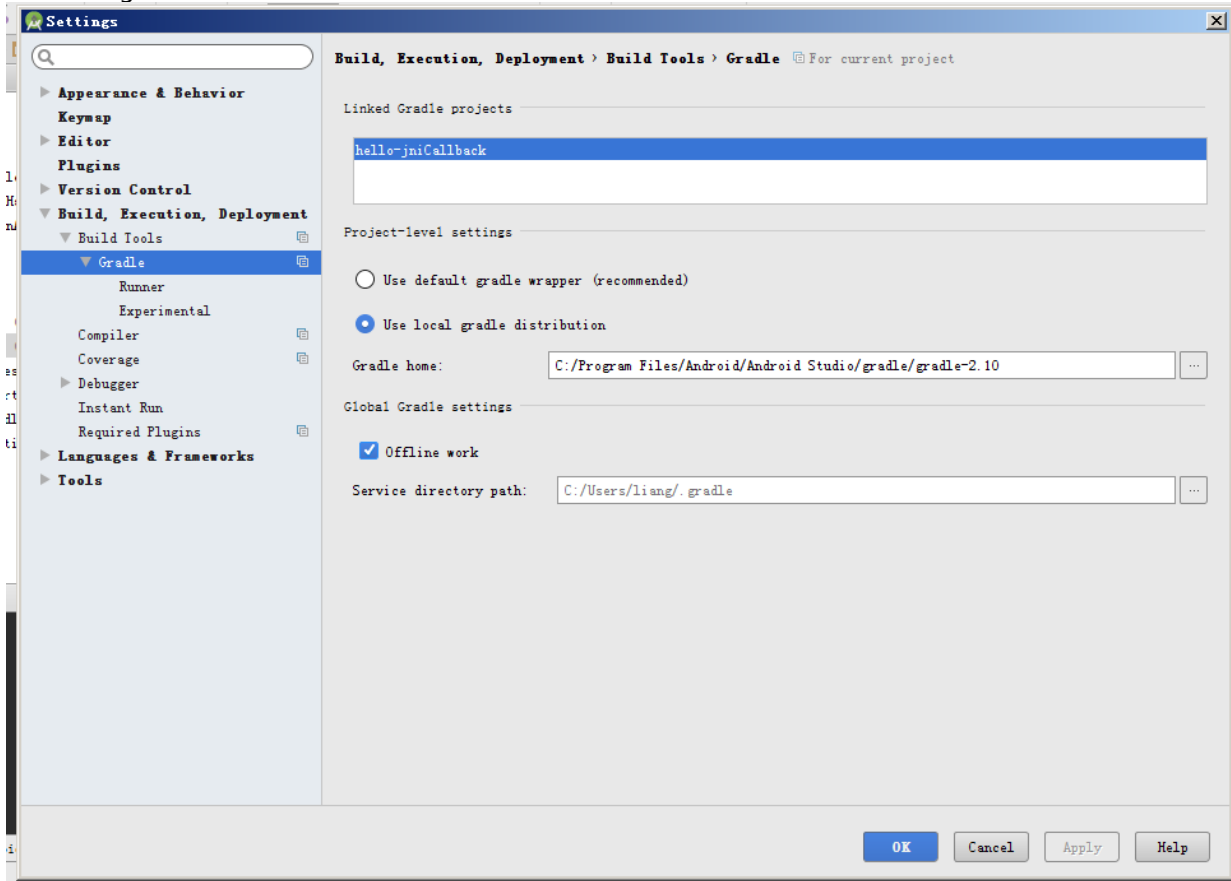
以下自己实践教程

```
#Thu Jun 02 18:25:42 CST 2016
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
```

```
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-2.5-all.zip
```

设置gradle

file->settings



如法炮制，修改build.gradle，将“:1.3.0”改为“-experimental:0.2.0”。

这是因为目前的NDK需要一个叫“experimental”的插件，它目前的版本是0.2.0。

注意：在现在以及后面的代码修改中，一个字符也不要错，仔细看本文的说明，一个冒号、一个点也不要漏掉，不然调试时报错，你都看不懂（调试不会说“你这里缺了一个分号”，它只会说一句让你丈二和尚摸不着头脑的话，甚至跟真正的错误八竿子也打不着）。改完之后，立刻会冒出一个***警告，说gradle已经被你改了，你得在项目中更新它。这件事待会儿再做。



以上使用了较旧的插件可以使用以下新版的插件

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle-experimental:0.7.0'
    }
}

// NOTE: Do not place your application dependencies here; they belong
// in the individual module build.gradle files
```

接下来修改app/build.gradle（注意，有两个build.gradle，刚才改的是根目录下的，现在改的是app目录下的）。

这里要改的地方特别多，先用纯文本标一下（你别复制这段代码，因为可能有些数字和你的项目不匹配，你就照着这个模板改你的代码）。

再次提醒：必要的地方，一个字也别漏掉，一个也别多，哪怕是一个大小写的区别。

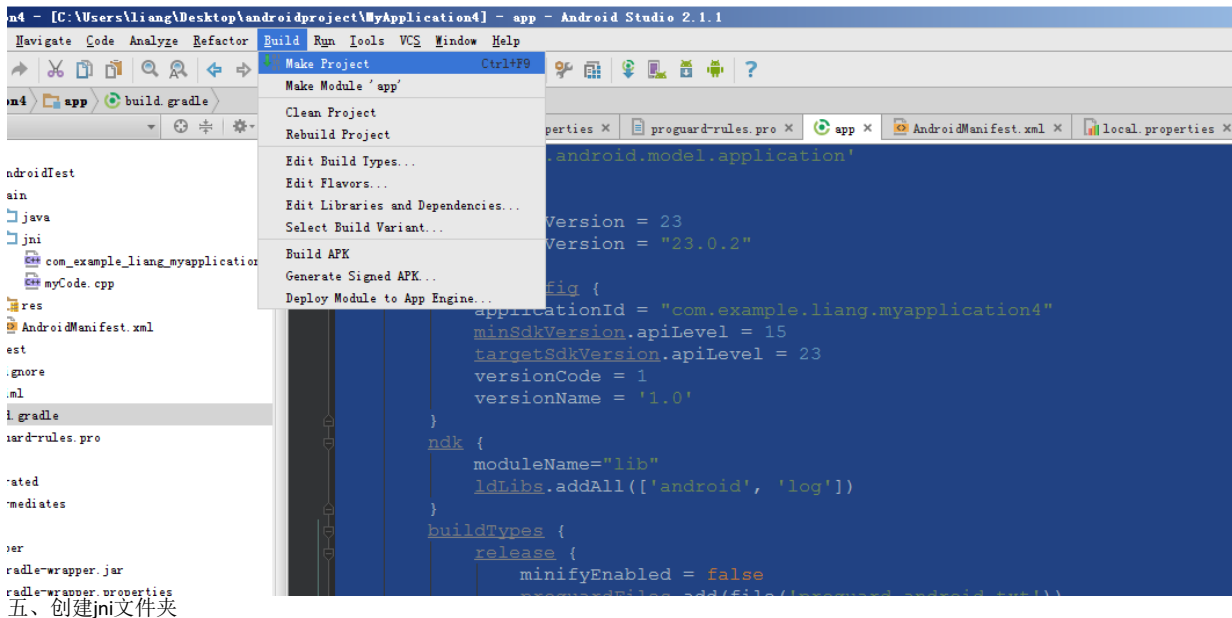
以下自己实践教程

```
apply plugin: 'com.android.model.application'
model {
    android {
        compileSdkVersion = 23
        buildToolsVersion = "23.0.2"

        defaultConfig {
            applicationId = "com.example.liang.myapplication4" //程序包名
            minSdkVersion.apiLevel = 15
            targetSdkVersion.apiLevel = 23
            versionCode = 1
            versionName = '1.0'
        }
        ndk {
            moduleName="lib" //这是将来so文件的名称，自己取
        }
        ldLibs.addAll(['android', 'log']) //加载系统默认的库
    }
    buildTypes {
        release {
            minifyEnabled = false
            proguardFiles.add(file('proguard-android.txt'))
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.4.0'
}
```

三、更新gradle



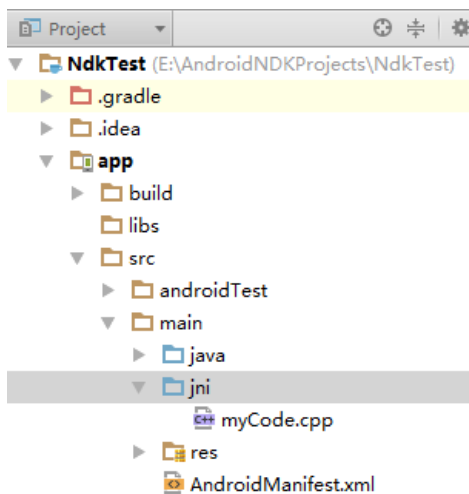
五、创建jni文件夹

在app/src/main文件夹上点右键，在弹出菜单中选择“New”、“Folder”、“JNI Folder”，按提示进行。有一个“Change Folder Location”选项，不需要勾选，因为jni文件夹采用默认的位置（在main文件夹中）就行。然后main目录下会出现jni文件夹。

六、创建C++源文件

在jni文件夹上点右键，在弹出菜单中选择“New”、“C/C++ Source File”。在出现的对话框中，给即将创建的C++文件取名（不要加后缀，软件会自动加上的）。有一个选项“Create associated header”，不要勾选它，现在不需要创建头文件。

之后，jni文件夹中出现了“myCode.cpp”，我们先不管它，先到Android部分做一些准备工作。



七、建一个演示按钮

这是纯Android操作，可能你已经熟悉了，但初学者需要知道。

编辑主窗口的xml文件（在此范例里是“activity_main.xml”）。如果你在界面上看不到它，就到Project目录的“app/src/main/res/layout/”中找到它，双击它打开编辑窗口。

窗口中有个手机图样，是模拟软件显示的效果，上面有一行字“Hello World”，是AndroidStudio自动加上的显示文字的控件。双击它，会打开一个小窗口，上面有“id”栏，在这里给该控件取名为“textView”（实际上可以随便取，但为了和后面的步骤衔接，就取这个名吧），这是java代码将要引用的。

再添加一个按钮，方法是：在左边的组件列表中找到“Button”，拖到手机图上。

当然你也可以用编辑代码的方式加这个按钮，这里对纯Android操作只说最简单的方法。

双击这个按钮打开小窗口，给它的“id”取名叫“button1”。

然后点上方的“MainActivity.java”选项卡，进入主窗口代码编辑界面，把代码改成这样：

```
package myself.exercise.ndktest;
```

```
import android.app.Activity;
import android.os.Bundle;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
```

```
public class MainActivity extends Activity {
```

```
    TextView textView; //声明一个标签
    Button button1; //声明一个按钮
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        textView = (TextView)findViewById(R.id.textView); //在xml中找到标签
```

```
        button1 = (Button)findViewById(R.id.button1); //在xml中找到按钮
```

```
        button1.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
```

```
            public void onClick(View v) {
                //点按钮以后发生的事
            }
        });
    }
}
```

新手注意：顶端的这一句：“package myself.exercise.ndktest;”，表明这个项目的包名是“myself.exercise.ndktest”，以后引用包名时，可以到这儿来拷贝。

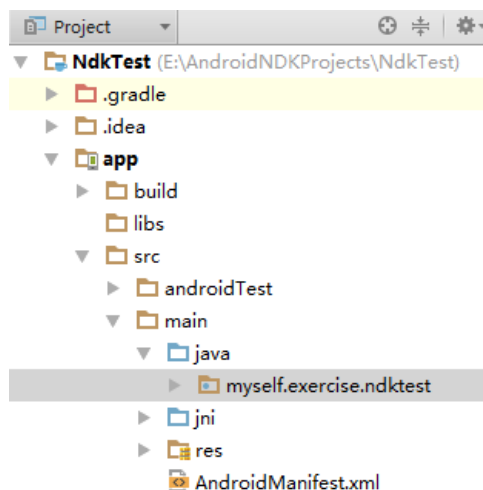
另外，在Android Studio中输入类名（比如“TextView”）时，如果你没写错它却显示为红色，表明Android Studio还没有识别这个名称，上面的“import”还缺点什么，你有两种办法：

1. 重新输入它，等待一个提示框出现，再按Tab键，Android Studio会自动帮你完成输入，把相应的“import”添加到代码中，比如添加“import android.widget.TextView;”，这时就不报错了。

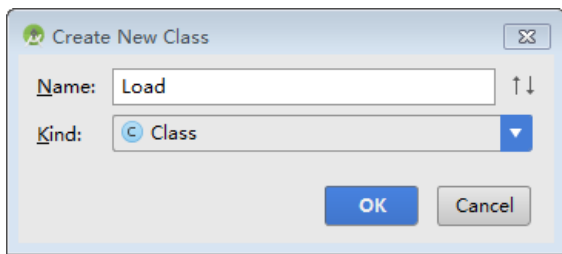
2. 单击它，当提示框出现时，按“Alt + 回车键”。

八、建一个新的java类，用来加载C++库

在“Project”目录中找到“app/src/main/java/myself.exercise.ndktest”：



在这个文件夹上点右键，在弹出菜单中选择“New”、“Java Class”，在弹出窗口中取名“Load”（不要加后缀，软件会自动加上）：



点“OK”之后，“myself.exercise.ndktest”文件夹中就出现了新的类“Load”（其实它的全名是“Load.java”）。双击它打开它的代码窗口，把代码改成这样：

```
package myself.exercise.ndktest;
```

```
public class Load {
    static { //载入名为“1b”的C++库
        System.loadLibrary("1b");
    }
    public native int addInt(int a, int b); //调用库里的方法“addInt”，这是计算a和b两个整数相加
}
```

```
package myself.exercise.ndktest;

public class Load {
    static { //载入名为“1b”的C++库
        System.loadLibrary("1b");
    }
    public native int addInt(int a, int b); //调用库里的方法“addInt”，这是计算a和b两个整数相加
}
```

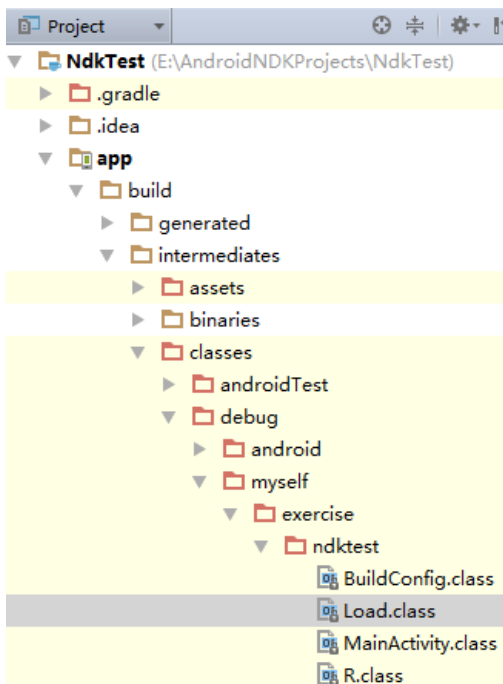
现在“addInt”是红色的，因为前面的“native”还没有找到下落，以后等native函数（C/C++函数）有了，这里自然就不红了。

现在记住几个关键的名称：

- 即将创建的C++库的名称：1b
- 加载此库的java包名：myself.exercise.ndktest
- 加载此库的java类名：Load
- 从此库中调用的方法：int addInt(int a, int b)（参数是两个整数，返回是整数）

九、生成头文件

走一遍主菜单“Build>Make Project”，这一步有没有效果，要看这里：



原来是没有“Load.class”的，但是“Make Project”之后，它就加进来了。前提是“Make Project”要成功，不成功，什么也不会改变。至于不成功的原因，可能是你漏了哪个字符，可能是没有严格按本指南操作，也可能你的软件版本与本文所说的不一样。软件开发是由一堆一堆人为的规矩组成的，不是理论物理，不是纯粹数学，没有真理，我们永远挣扎在别人制定的规矩中，没办法，因为我们没有比尔·盖茨那样的本事。

点界面左下角的“Terminal”标签，打开命令行窗口：

```
Terminal
+ Microsoft Windows [版本 6.1.7601]
X 版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

E:\Android\NDKProjects\NdkTest>
```

在这句话的末尾输入：

`cd app/build/intermediates/classes/debug`

按回车键，进入“debug”目录：

```
Terminal
+ Microsoft Windows [版本 6.1.7601]
X 版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

E:\Android\NDKProjects\NdkTest>cd app/build/intermediates/classes/debug

E:\Android\NDKProjects\NdkTest\app\build\intermediates\classes\debug>
```

接着输入：

`javah -jni myself.exercise.ndktest.Load`

按回车键。

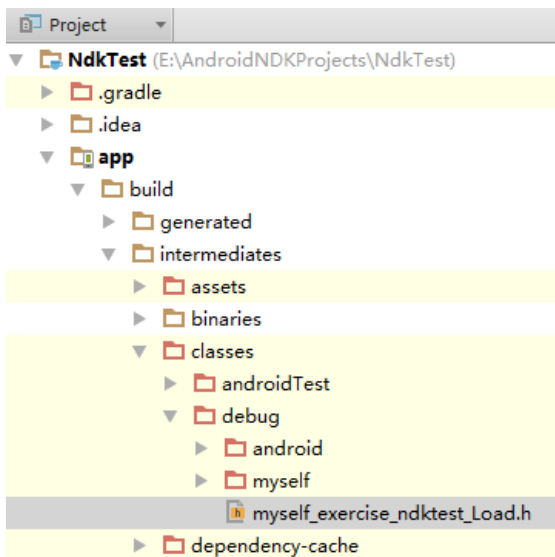
```
Terminal
+ Microsoft Windows [版本 6.1.7601]
X 版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

E:\Android\NDKProjects\NdkTest>cd app/build/intermediates/classes/debug

E:\Android\NDKProjects\NdkTest\app\build\intermediates\classes\debug>javah -jni myself.exercise.ndktest.Load

E:\Android\NDKProjects\NdkTest\app\build\intermediates\classes\debug>
```

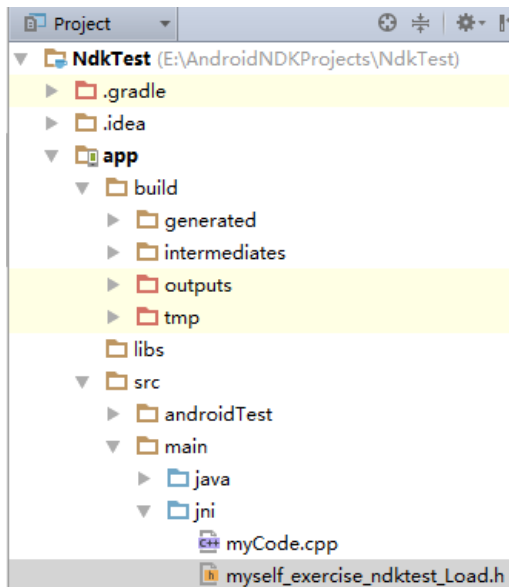
注意：“myself.exercise.ndktest”是包名，“Load”是调用C++库的类的名称，你开发自己的软件时，这些要改的。要是你没写错，“debug”目录下会有一个头文件生成：



它的后缀是“h”，表明它是C/C++头文件；它的名字仍然由上述包名和类名组成，只不过点变成了横杠。

右键单击此文件，在弹出菜单中选择“Cut”。

然后找到“app/src/main/jni”目录，在jni文件夹上右键单击，在弹出菜单中选择“Paste”，在接下来的对话框中什么也不改直接点“OK”，你会看到头文件已经被转移到了jni文件夹中，与我们先前建立的源文件“myCode.cpp”文件在一起。



双击此文件，可以看到它的代码：


```

/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class myself_exercise_ndktest_Load */

#ifndef _Included_myself_exercise_ndktest_Load
#define _Included_myself_exercise_ndktest_Load
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:    myself_exercise_ndktest_Load
 * Method:   addInt
 * Signature: (II)I
 */
JNIEXPORT jint JNICALL Java_myself_exercise_ndktest_Load_addInt
    (JNIEnv *, jobject, jint, jint);

#ifdef __cplusplus
}
#endif
#endif

```

C++的普通语法不解释了，这里特别的一段是：

```

JNIEXPORT jint JNICALL Java_myself_exercise_ndktest_Load_addInt
    (JNIEnv *, jobject, jint, jint);

```

它声明了方法“addInt”，并指定了可以调用它的java包是“myself.exercise.ndktest”、可以调用它的java类是“Load”。“jint”与“JNICALL”之间有一个红色警告，不管它，这是假报错。上面有一句“extern “C””，表明此方法是向外输出的，是给java程序调用的。在这里，我们什么也不用改。

十、编辑源文件

双击“myCode.cpp”打开其代码编辑窗口，写入以下代码：

```

#include "myself_exercise_ndktest_Load.h"

JNIEXPORT jint JNICALL Java_myself_exercise_ndktest_Load_addInt (JNIEnv *, jobject, jint a, jint b) {
    return a + b;
}

```

这是方法“addInt”的实现——两个整数相加，得到一个整数结果。

十一、调用库方法

刚才在“Load.java”中加载了库，调用了库方法，现在在主窗口中调用“Load.java”所调用的这个方法。主窗口的代码刚才已经贴过了，现在，在“点按钮以后发生的事”这句话下面添加：

```


Load load = new Load();
int r = load.addInt(100, 50);
textView.setText("C++库计算“100+50”的结果是：" + String.valueOf(r));

```

整个过程是：

1. 调试或发布时，“myCode.cpp”将被编译为库“1b”。
2. “Load”加载了库“1b”，调用了库中的方法“addInt”。
3. 主窗口的按钮点击事件，通过“Load”调用这个方法，算出100+50等于多少。

十二、调试

手机连好，点界面上方的调试按钮 。过一会儿弹出一个窗口，在其中选择已连好的手机，点“OK”，等一会儿到手机上看结果。不出意外的话，手机上会出现刚做的软件，点其中的按钮会显示C++计算的结果：



C++库计算“100+50”的结果是：150

New Button