# 访问数组：

## 基本类型数据数组

相对比较简单，因为我们JNIEnv 提供了一系列的方法来给我们将java 数组转换从JNI中数据类型对应的数组，而且JNI中的基本数据类型又大多和C语言只是名字上面的不同，实质都是差不多的，所以我们很容易就能使用他们

## 引用类型数据的数组

### JNI 引用

#### 为什么JNI 中有引用

由于本地代码不能直接通过引用操作JVM内部的[**数据结构**](http://lib.csdn.net/base/datastructure)，要进行这些操作必须调用相应的JNI接口来间接操作所引用的数据结构。JNI提供了和Java相对应的引用类型，供本地代码配合JNI接口间接操作JVM内部的数据内容使用。如：jobject、jstring、jclass、jarray、jintArray等。因为我们只通过JNI接口操作JNI提供的引用类型数据结构，而且每个JVM都实现了JNI规范相应的接口，所以我们不必担心特定JVM中对象的存储方式和内部数据结构等信息。

### 什么是JNI 引用

分类：局部引用和全局引用 和 弱全局应用

### 局部引用

通过NewLocalRef和各种JNI接口创建（FindClass、NewObject、GetObjectClass和NewCharArray等）

局部引用通过DeleteLocalRef 手动释放对象。 访问一个很大的java 对象的时候，使用完了就需要释放，因为占用了太多的耗时操作； 其次，在创建大量的局部引用的时候，我们需要在创建后就立刻释放。

为什么要手动释放：

释放的方式有两种：

一个是本地方法执行完毕后JVM自动释放，另外一个是自己调用DeleteLocalRef手动释放。

大部分情况下，我们在实现一个本地方法时不必担心局部引用的释放问题，函数被调用完成后，JVM 会自动释放函数中创建的所有局部引用。尽管如此，以下几种情况下，为了避免内存溢出，我们应该手动释放局部引用：   
1、JNI会将创建的局部引用都存储在一个局部引用表中，如果这个表超过了最大容量限制，就会造成局部引用表溢出，使程序崩溃。经[**测试**](http://lib.csdn.net/base/softwaretest)，Android上的JNI局部引用表最大数量是512个。当我们在实现一个本地方法时，可能需要创建大量的局部引用，如果没有及时释放，就有可能导致JNI局部引用表的溢出，所以，在不需要局部引用时就立即调用DeleteLocalRef手动删除

### 全局引用：

调用NewGlobalRef基于局部引用创建

共享：可以跨方法，跨线程使用，直到它被手动释放才会失效。

它和局部引用不同，全局引用只能通过NewGlobalRef 才能创建，而且必须要手动释放，它会阻止它所引用的对象被GC回收。

每一个JNI引用被建立时，除了它所指向的JVM中对象的引用需要占用一定的内存空间外，引用本身也会消耗掉一个数量的内存空间。作为一个优秀的程序员，我们应该对程序在一个给定的时间段内使用的引用数量要十分小心。短时间内创建大量而没有被立即回收的引用很可能就会导致内存溢出。

当我们的本地代码不再需要一个全局引用时，应该马上调用DeleteGlobalRef来释放它。如果不手动调用这个函数，即使这个对象已经没用了，JVM也不会回收这个全局引用所指向的对象。

### 弱全局引用

调用NewGlobalRef基于局部引用创建

可以跨方法，跨线程使用，但是弱引用不会阻止GC回收它所引用的对象。

它的特点就是JNI 会根据内存情况来自己回收对象。所以，当我们定义一些引用，这些引用不一定要阻止GC回收它所指向的对象时，弱应用就是一个最好的选择。只不过，我们使用这个弱引用前，我们要旁段这个检查这个缓存的弱引用是否活动

创建的时候我们就需要使用：newWeakGlobalRef， 销毁就使用：DeleteGlobalWeakRef

# JNI 处理异常

1．保证程序不奔溃，java代码会继续运行；

2. C 程序要能够继续执行

JNI 自己抛出的异常，没有办法抓到，我们只能清除异常代码，然后执行自己的异常代码。

和java 不同，JNI中抛出的异常ThrowNew 不被处理的话，不会立即终止本地方法的执行。异常发送后，JNI 程序员必须手动处理。

更多高级应用，我们可以将异常写成JNI的工具类，通过工具类来处理jni 工程中所有的异常。

# 缓存策略

直接使用static 举例说明

1. C 语言中的static 关键字
2. 静态全局变量

存储在静态区，所以只需要定义一次即可使用，以后无需再定义

作用范围：只在定义它的源文件中有效，其他文件中不能使用这个变量。而且在定义它的源文件中，只有在定义了这个变量后面的代码才能使用

1. 静态局部变量

存储在静态区，同样是只需要一次定义就可以了。

作用范围：只能在定义它的函数中有效

缓存策略与弱引用的问题

举例说明。不恰当的缓存和弱引用的联合使用带来的问题

# JNI基础课程总结：

什么是JNI

JNI 与NDK 的关系

什么是静态库，什么是动态库

JNIEnv 结构体的说明

JNI基本数据类型

JNI 访问静态和非静态方法

JNI 访问构造函数

JNI 中字符串的处理

JNI 中访问java 中的数组

JNI 引用//

JNI 异常处理机制

JNI 缓存机制