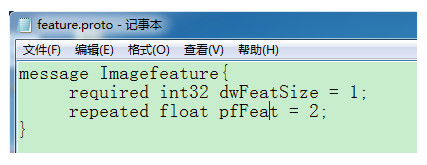
**7.Protobuf介绍**

Caffe中的Caffe.proto文件负责了整个Caffe网络的构建，又负责了Caffemodel的存储和读取。下面用一个例子介绍Protobuf的工作方式。

利用protobuffer工具存储512维度图像特征：

1.message 编写：新建txt文件后缀名改为proto,编写自己的message如下，并放入解压的protobuff的文件夹里；

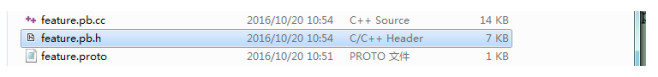


其中，dwFaceFeatSize表示特征点数量；pfFaceFeat表示人脸特征。

2.打开windows命令窗口(cmd.exe)---->cd空格，把protobuff的文件路径复制粘贴进去------>enter；

3.输入指令protoc \*.proto --cpp\_out=.    --------->enter

4.可以看到文件夹里面生成“ \*.pb.h”和“\*.pb.cpp”两个文件，说明成功了



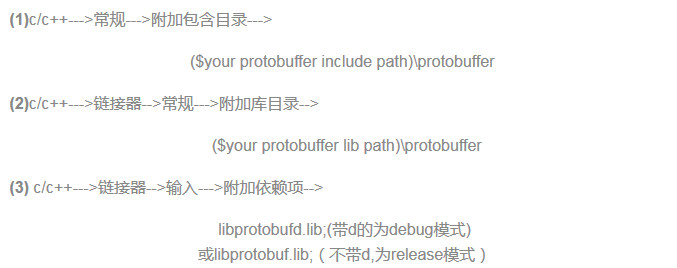
5.下面可以和自己的代码整合了：

(1) 新建你自己的工程，把“ \*.pb.h”和“\*.pb.cpp”两个文件添加到自己的工程里，并写上#include" \*.pb.h"

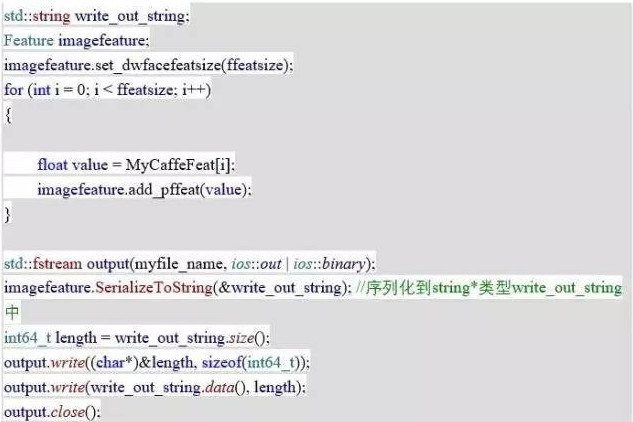
(2) 按照配库的教程把库配置下就可以了。

VS下Protobuf的配库方法：

解决方案---->右击工程名---->属性

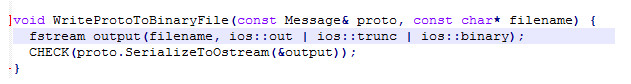


使用protobuf进行打包的方法如下代码：



## **（1）Caffe的模型序列化**

BlobProto其实就是Blob序列化成Proto的类，Caffe模型文件使用了该类。Net调用每个层的Toproto方法，每个层的Toproto方法调用了Blob类的ToProto方法，这样完整的模型就被都序列化到proto里面了。最后只要将这个proto继承于message类的对象序列化到文件就完成了模型写入文件。Caffe打包模型的时候就只是简单调用了WriteProtoToBinaryFile这个函数，而这个函数里面的内容如下：



至此Caffe的序列化模型的方式就完成了。

## **（2）Prototxt的简单说明**

Caffe网络的构建和Solver的参数定义均由此类型文件完成。Net构建过程中调用ReadProtoFromTextFile将所有的网络参数读入。然后调用上面的流程进行整个caffe网络的构建。这个文件决定了怎样使用存在caffe model中的每个blob是用来做什么的，如果没有了这个文件caffe的模型文件将无法使用，因为模型中只存储了各种各样的blob数据，里面只有float值，而怎样切分这些数据是由prototxt文件决定的。

Caffe的架构在框架上采用了反射机制去动态创建层来构建Net，Protobuf本质上定义了graph，反射机制是由宏配合map结构形成的，然后使用工厂模式去实现各种各样层的创建，当然区别于一般定义配置采用xml或者json，该项目的写法采用了proto文件对组件进行组装。