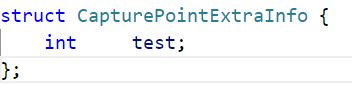
# CAP点扩展：

数据结构：

CAP点类CapturePoint，位于文件common/capture\_point.h

保留了结构体CapturePointExtraInfo用于扩展CAP点信息：



如果结构体中出现任何二级间址（如指针，数组，其他有二级地址的对象如vector），需要在函数from\_char\_array和to\_char\_array中添加对二级地址的序列化处理。

地图结构：

目前的地图中已经保留了CAP点结构，以类SimpleGridMap成员变量std::vector<CapturePoint> cap\_points;的格式储存。

在地图的序列化函数from\_char\_array和to\_char\_array中已经有对CAP点列表的处理，不需要额外修改，CAP点的数据会接在普通地图和反射板地图后面。

**Free状态中对CAP点的处理可以使用如下路线：**

通过AGVInstance获取当前地图指针，直接修改其指向的成员变量cap\_points

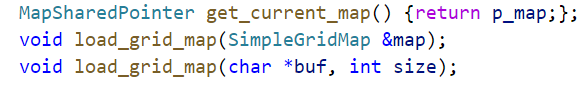
**建图过程中对CAP点的处理：**

在AGVInstance中建立一个链表，暂存CAP点信息、当时的点云数据和当时的位姿估计。在函数AGVInstance::painting\_map\_thread\_function结束的时候（即建图结束的时候），遍历之前建立的列表，使用scan-matching优化每一个CAP点的位姿，并将优化后的CAP点加入到p\_map指向的成员变量cap\_points

（可参考代码中对反光板的处理，也是记录的反光板数据和当时的激光点云，最后做优化）

# 本地地图扩展：

AGVInstance中预留的地图处理接口：



**详情见agv.h文档**

本地地图模块单独写成一个文件，在main.cpp中调用。

在main函数中初始化模块，

在网络消息处理函数handle\_package(TCPDataPackage \*, TCPDataPackage \*)中，选择相应的网络指令码，调用get\_global\_agv\_instance()获取当前的AGVInstance，调用相关接口即可。

# 二维码定位扩展：

AGVInstance中预留的定位接口：



二维码模块单独写成一个文件，初始化时分配专用的线程，监测摄像头，读取二维码。

**当检测到二维码并获取到精确定位时**，调用get\_global\_agv\_instance()获取当前的AGVInstance，调用其update\_last\_position函数，等级参数为2，2 。表示自己的位姿等级为2，接下来要把系统位姿设置成2级，屏蔽掉激光雷达的定位信息。（没有禁用里程计，里程计依然在用于短程估计）

**当丢失二维码时**，调用update\_last\_position函数，等级参数为2，1 。表示将系统位姿还原到1级，让激光雷达可用。可以传入一个时间戳为0的位姿，在不更新系统位姿的情况下改变等级