编程大赛设计指南

框架程序版本：vs2005 及其以上。

框架程序中涉及到 UDP 通信文件一般不修改，主要阅读 zfc.cpp，zfc.h，recvCmd.cpp，recvCmd.h，sendCmd.cpp，sendCmd.h 六个文件。现在进行依次解读。

阅读此文档前请先阅读任务书。

zfc.cpp：（一般不修改）

首先定义变量：

string teamName="";//团队名称

string serverIp="";//服务器 IP 地址

ushort serverPort=0;//服务器端口号

string myRole=""; //"POL" OR "THI"//角色名称

ushort localPort=0;//客户端端口号

对运行参数，进行分析，存储。

如何运行/如何设置参数：

参数在任务书中有说明，一般为 teamname serverip 31000 POL 31002 格式或者 teamname serverip 31000 THI 31001 格式。

- 编写批处理文件。

步骤：

1. 编译运行完成的工程，生成 debug 文件，找到 zfc.exe 文件。
2. 在同级文件夹中新建记事本文件，输入 zfc teamname serverip 31000 POL 31002 或 zfc teamname serverip 31000 THI 31001，保存。然后把文件后缀名从 txt 改为 bat。前提是文件的后缀名没有隐藏才能修改。这时就生成了批处理文件，运行时先打开服务器，再打开该文件就可以运行。警察和小偷分别需要一个批处理文件，一共需要打开两个文件。
3. 可参考 demo 文件夹中的 demo.exe 和两个批处理文件（右键，点击编辑可查看）。注意批处理文件的第一个参数 demo 是与 demo.exe 同名。如果你生成的是 zfc.exe，则需要把批处理文件中第一个单词改为 zfc，或者把 zfc.exe 改为 demo.exe。

- 在编译器中设置参数。

步骤：（以 vs2005 举例）

1. 打开“项目”菜单，最后一项“zfc 属性”。
2. 左侧“配置属性”下的“调试”窗口。
3. 右侧“命令参数”中输入 teamname serverip 31000 POL 31002 或 teamname serverip 31000 THI 31001。
4. POL 命令和 THI 命令分别表示警察和小偷。如果两者需要同时使用 IDE 调试，请复制一份工程，分别打开，输入不同参数。

以上 serverip 一般指 127.0.0.1，即本机 IP。

zfc.h：（一般不修改）

我们注意到该文件中定义了

extern string teamName;

extern string myRole; //“POL” OR “THI”

extern ushort localPort;

说明在别的文件中可调用这三个参数。但是一般有用的只有 myRole 参数，POL 和 THI 分别表示警察和小偷，在其他文件操作中，判断这个参数，可运行不同角色的相关代码。

zfc.cpp 和 zfc.h 文件在一般情况下，不需要修改，只需要记得使用 myRole 参数就可以。

recvCmd.cpp：（补全两个函数）

```
int recvIni(char *str)
{
    //接收初始化指令处理
    return 0;
}
```

这个函数收到的消息格式在任务书中声明：

INI 指令：INI[maxx,maxy]<m,n>(4,1)<id1,x1,y1;…;idk,xk,yk;>。

其中 maxx 为地图的宽度，maxy 为地图的高度。<m,n>表示警察视距为 m,小偷视距为 n。(4,1)表示 4 个警察和 1 个小偷，id1 为角色的 id（id>=0），x1，y1 为角色 id1 的初始坐标，坐标取值范围>=0。如果警察客户端，则只能收到警察初始坐标；如果是小偷客户端，则只能收到小偷的初始坐标。

在该函数中，对 char *str 这个字符串进行分析，把地图宽高，警察小偷视距，数量，初始坐标，用全局变量存储起来，以便之后使用。

设计一般考虑角色是警察或者角色是小偷两种情况。

```
int recvInf(char *str)
{
    //接收 INF 指令
    return 0;
}
```

这个函数收到的消息格式在任务书中声明：

INF[轮数](id1,x1,y1;…;idm,xm,ym;)<id4,x4,y4;>(x5,y5;…;xk,yk;)

INF 指令由服务器发给各角色，通知该角色视距内的场景信息；[轮数]中的“轮数”代表当前的行动轮数；id1 为角色 id，x1、y1 为角色 id1 的坐标。第一个小括号()中的坐标为各警察的坐标；尖括号<>中的坐标为小偷的坐标；第二个小括号中是障碍的坐标。括号中的内容可以为空，但括号仍然存在；没有明确通知的地方为平地。

在该函数中，对 char *str 这个字符串进行分析，把警察小偷位置和视距内障碍物坐标数据，用全局变量存储起来，以便之后使用。

这时进行智能分析，应该怎么走比较合理。

然后把移动信息发送给服务器。

MOV[轮数](id1,方向; …;idn,方向;)

MOV 指令由各角色发给服务器；[轮数]需要回传 INF 指令中的轮数；（）中的 id 为角色 id；字符 D 代表东，N 代表南，X 代表西，B 代表北，T 表示不动。所有角色移动信息必须完整填写到 MOV 指令中；如果向不能移动的方向发了移动指令，则保持原地不动。

然后调用 sendCmd.cpp 中的 sendCmd(char *buf)函数

举例：

```
char buf[100]=”MOV[4](1,T;2,T;3,T;4,T;)”;
```

sendCmd(buf);

注意：sendCmd 函数定义在 sendCmd.cpp 文件中，不能被 recvCmd.cpp 中函数调用。

这时可以在 recvCmd.h 中加入一句：

extern int sendCmd(char *buf);

然后就可以调用了。

实际上整个工程只需要补全 recvCmd.cpp 中的这两个文件。

而需要阅读的文件也只有上 zfc.cpp，zfc.h，recvCmd.cpp，recvCmd.h，sendCmd.cpp，sendCmd.h 六个文件。