**选手接口文档**

1.选手接口

void updateAge() 升级时代

void construct(BuildingType building\_type, Position pos, Position soldier\_pos) 建造命令 参数一次是建造的建筑类型，建造的位置和出兵位置(非造兵建筑空缺这个参数)

void upgrade(int unit\_id) 升级单位命令，参数为目标单位id

void sell(int unit\_id) 出售命令，参数为单位id

void toggleMaintain(int unit\_id) 修理命令，参数为目标单位id

2.游戏静态信息

选手通过bool变量ts19\_flag获取自己是0/1号玩家

选手通过变量int\*\* ts19\_map来获取地图信息，其中map是一个200\*200的int数组。其中0代表无路，1代表有路，2代表主建筑。

3.获取游戏的全局动态信息

struct State

{

int turn;

int winner;

\_resource resource[2];

Age age[2];

vector<Building> building[2];

vector<Solider> soldier[2];

};

游戏信息储存在这样一个结构体中，玩家可以通过指针state获取这些信息。

turn是当前游戏回合数目，winner为当前胜利者，其他为选手的资源和建筑，兵种

其中resource age building soldier这几个数组分别储存了两个选手的信息，其中[0]是player0的信息，[1]是player1的信息

其中building[i] solider[i]储存的是装有结构体Building，Soldier实例的向量（STL的vector）。

4.对应资源信息结构体的实例

struct \_resource

{

int building\_point;

int resource;

};

struct Position {

int x;

int y;

};

struct Soldier {

SoldierName soldier\_name; //soldier的类型

int heal; //soldier的血量

Position pos; //soldier的位置

int flag; //soldier的阵营

int unit\_id; //soldier的ID

int level; //solider的level

};

struct Building {

BuildingType building\_type; //building的类型

int heal; //building的血量

Position pos; //building的位置

int flag; //building的阵营

int unit\_id; //building的id

int level; //building的等级

};

注：1.选手是指在player.cpp文件的f\_player()函数中编写自己的代码，请不要在其他文件中写代码。

2.对战方式：

windows

首先打开server.exe程序

运行编译链接好的C++文件生成的.exe文件

运行一次代表连接一个AI，需要且仅需要运行两次。

Macos

同windows

注意g++编译命令 g++ main.cpp api\_player.cpp player.cpp communication.cpp -pthread –o filename

3.程序崩溃惩罚：

选手应该首先保证自己的程序没有非法调用，一旦程序崩溃，无论在游戏中局势怎样，崩溃方直接判定为负，如果两人在同一局崩溃，就判定为平局。

4.超时惩罚

首先，程序能够保证选手拿到的游戏局势是在调用f\_player时最新的，如果跨回合了，那么这一次执行f\_player时的状态就会落后当前游戏状态，但是再下一次重新执行f\_player时又会拿到最新状态。其次，如果选手程序超时，服务器不会等待你的命令，因此，超时会导致你落后别人回合。而你在之前第m回合的命令，超时了x轮，在服务器的m+x轮把你第m轮的命令发送到了，那么服务器会接受其中的合法命令。因此，超时会带来落后回合的惩罚，请尽量保证不要超时。

提醒：

1. 我们单回合指令上限是50条，请不要在一回合发出过多指令，如果超出一些，但并不太多，我们会随机截取50条，再检查合法性。如果一回合发出过多指令，如我们实验中测试发送4w条指令，通信速度会下降，每次会落后4回合左右。