


本地对局平台 Game Platform.cpp 使用方法

1. 更改 bot 源文件

首先需要对 bot 的 C++ 源代码进行一些修改。步骤如下：

1. 按照下图，将 BotzoneIO 命名空间中的 read() 函数的参数改为 string line，并删除函数前两行。



```
namespace BotzoneIO
{
    using namespace std;
    void read(string line) // 函数参数改为:
    {
        // 读入输入 (平台上的输入是单行)
        string line;
        getline(cin, line);
        Json::Value input;
        Json::Reader reader;
        reader.parse(line, input);
        startTime = clock();
    }
}
```

2. 按照下图，将主函数的参数改为 int argc, char* argv[] (即命令行参数)，然后将 Botzone 平台上的输入以命令行的形式读入，作为参数传至 BotzoneIO::read() 函数中。

```
int main(int argc, char* argv[])
{
    if (argc <= 1)
    {
        std::cout << "Too few command line arguments" << std::endl;
        return -1;
    }
    string line(argv[1]);
    BotzoneIO::read(line);
    prepareData(); // 记录myHand、remainingCards
}
```

3. 如需通过对局平台自动调参，需要将所调参数也以命令行参数的形式传入 bot 源文件中，并对对局平台的代码做适当修改以达到自动调参的目的。

2. 编译生成 exe 文件并复制到对局平台所在目录

将 bot 源文件编译，生成 exe 格式的文件，并复制到 Game Platform.cpp 的同一目录下。对局需要 3 个 bot，即 3 个 exe 格式的文件。然后，将这 3 个文件分别重命名为 bot0.exe、bot1.exe、bot2.exe。

由于进行大量对局耗时较长，就不要用蒙特卡洛了，直接采用 findBestAction() 给出的策略就行了。

3. 运行 Game Platform.cpp

运行 Game Platform.cpp，即可开始自动对局。若 Game Platform.cpp 编译失败，可以试试将文件中 get_response() 函数中的 strcpy_s 和 strcat_s 改成 strcpy 和 strcat。若是 Linux 系统还需将 get_response() 函数中的 _popen 和 _pclose 改成 popen 和 pclose。

修改主函数中的 total_games 可以更改对局总数，各个 bot 的获胜次数记录在数组 player_winning_times[] 中，并在所有对局结束后输出。每局对局开始会输出各个 bot 的初始手牌（不包括 3 张公共牌）和 3 张公共牌，对局中会输出每个 bot 的出牌记录，最后会输出胜者（如果农民获胜，两个农民都会输出，它们的获胜次数也都会加 1）。

对局平台不会检查 bot 的出牌是否合法，谁先出完牌谁就获胜。对局包括了叫分阶段（同样不会检查叫分是否合法）。