## 17.7 五子棋

**1. 题目描述**

五子棋的游戏规则是两人对弈，使用黑白两色棋子，轮流下在棋盘上，当一方先在横线、竖线、斜对角线方向形成五子连线，则取得胜利。

**2. 设计要求**

⑴ 在内存中，设计数据结构存储游戏需要的数据。

⑵ 满足五子棋游戏的游戏规则。

⑶ 实现简单的人机对战功能。

**3. 数据结构设计**

在游戏运行过程中，需要在内存中保存游玩的状态，也就是棋盘，可以设计以下结构保存关卡信息。

struct BoardType

{

// 棋盘，0表示空位，1表示玩家棋子，2表示电脑棋子

int map[BOARDSIZE][BOARDSIZE];

// AI下棋的权重，权重取对AI最有利和对玩家最有害两者的最大值

int weight[BOARDSIZE][BOARDSIZE];

}

**4. 算法设计**

对于该款游戏，我们需要以下函数提供基本功能：

InitGame：初始化游戏的游玩状态

PlayerTurn：处理玩家回合的输入

UpdateWeight：根据棋盘局势更新电脑的下棋权重

ComputerTurn：电脑回合的处理

Update：负责根据内存中的游玩状态更新游戏的画面函数

CheckWin：判断是否有一方取得胜利的函数

人机对抗部分主要由UpdateWeight与ComputerTurn函数组成，其中UpdateWeight函数负责计算棋盘上每个空位的权重，便于电脑判断在哪里下棋是较为合理的。这里给出了一种简单的设计思路，UpdateWeight算法设计如下：

其中board数据类型为前文定义的BoardType，保存了棋盘当前的状态以及对应位置上的权重。

UpdateWeight：根据棋盘状态更新权重

输入：棋盘状态board

输出：无

1. 逐行逐列判断棋盘上该位置是否为空位，记行号为i，列号为j：

1.1. 若board.map[i][j]!=0，该位置不为空，board.weight[i][j]=-1；

1.2. 若board.map[i][j]==0，该位置为空：

1.2.1. 记对AI有利的权重为aiWeight=0；

1.2.2. 记对玩家有害的权重为playerWeight=0；

1.2.3. 若该位置放置AI的棋子，分别判断周围8个方向

相连的AI棋子个数，每个方向相连的个数记作cnt，

令aiWeight+=10^cnt；

1.2.4. 若该位置放置玩家的棋子，分别判断周围8个方向

相连的玩家棋子个数，每个方向相连的个数记作cnt，

令playerWeight+=10^cnt；

1.2.5. board.weight[i][j]=max(aiWeight，playerWeight)；

ComputerTurn函数负责根据处理电脑的回合，模拟人类与玩家进行对弈，其中出现的UpdateWeight函数为上文中的更新权重算法， board数据类型为前文定义的BoardType，保存了棋盘当前的状态。

ComputerTurn：电脑回合的处理

输入：无

输出：无

1. 调用UpdateWeight函数，更新权重；

2. 记maxVal=0，maxX=0，maxY=0，记录最大权重出现的位置；

3. 逐行逐列比较棋盘上的权值与maxVal的关系，记行为i，列为j：

3.1. 若board.weight[i][j]>maxVal：

3.1.1. maxVal= board.weight[i][j]；

3.1.2. maxX=i；

3.1.3. maxY=j；

4. 点(maxX，maxY)为权值最大的位置，board.map[maxX][maxY]=2；

CheckWin函数负责判断是否有一方获胜，其中board数据类型为前文定义的BoardType，保存了棋盘当前的状态。

CheckWin：根据棋盘状态更新权重

输入：棋盘状态board

输出：胜利方

1. 逐行逐列判断棋盘上该位置是否为空位，记行号为i，列号为j：

若board.map[i][j]!=0，该位置不为空：

1.1. 分别判断点(i,j)与周围8个方向相连且相同的棋子个数，

记作cnt；

1.2. 若cnt>=5，判断board.map[i][j]的类型：

1.2.1. 若board.map[i][j==1，玩家的棋子，返回玩家胜利；

1.2.2. 若board.map[i][j==2，电脑的棋子，返回电脑胜利；

2. 返回无人胜利；

**7. 思考题**

⑴ 在玩家游玩游戏的过程中，可能会出现误操作等需要“悔棋”的情况，需要用到何种数据结构保存玩家的上一步状态，又如何实现呢？

⑵ 上述算法设计中给出了一种简单的人机对抗算法的设计思路，请自行查阅资料，设计一款更加高效合理的权重算法。