Gomoku Ver 0.1.0 //2018.3.15

将15x15棋盘模拟成15x15的二维数组，开始前每个位置数据均初始化为0；

规定：黑棋记为1，白棋记为-1，一次只能着一子且已着子位置不能继续着子。

默认AI为黑棋(先行)。

设计思路：

AI行子依据优先度高低进行逐级计算，若满足高优先度条件则着子，有多种同优先度选择另行讨论(正常着子情况下)。

AI优先度：1.若能取胜，则制胜；  
 2.若不能取胜，检查是否对方有取胜条件，若有，则进行防守(阻止)；

3.若上两种皆无，进行正常行棋。

正常行棋时 按取胜>防守的优先度进行赋予权重，取权重高的走法行棋(若有多种则选取第一种)

由于正常行棋算法权重计算依据前两种算法实现，而防守算法则依据取胜算法进行调整，故先讨论第一种情况。

算法逻辑：

对于每个位置a[i][j]

for (i=0;i<15;i++)

for (j=0;j<15;j++)

检索从该点起单方向连续n项的值并求和

下以正向为例

a[i][j+1]~a[i][j-n-1] //横向 求和记为add11(n)

a[i+1][j]~a[i+n-1][j] //纵向 add12(n)

a[i+1][j+1]~a[i+n-1][j+n-1]//斜45° add13(n)

a[i-1][j-1]~a[i-n+1][j-n+1]//斜45° add14(n)

//默认在a[x][y]表述合法的情况下，实际算法需确保数据合法

//为表述方便正向求和统称为add1，反向统称为add2，正反统称add

终局条件if add==±5 则胜利(+)/失败（-）；

if ∀a[i][j]!=0&&∀add!=±5 则平局；

轮到AI行棋的情况下：

1. 取胜情况
2. 活4 记为win1:

if ∃a[i][j]==0使得 add1(4)以及add2(4)中共有一项==3 且该项add(5)==3则着子;

1. 双活3 记为win2：

if ∃a[i][j]==0使得 add1(4)以及add2(4)中共有两项==2且该项add(5)==2则着子

1. 双眠3 记为win3

if ∃a[i][j]==0 使得 add1(5)以及add2(5)中有两项==2且此两项的add(4)==3 则着子

1. 活3+眠3 记为win4

if ∃a[i][j]==0 使得 add1(5)以及add2(5)中有一项==2且该项add(4)==3

另有一项==2且该项add(4)==2 则着子

二．防守

依据取胜情况将add取负值判断（暂时略）

三．正常行棋

规定行棋优先度：(构成)活3>=冲4>活2>=眠3>眠2

1. 活3：

if ∃a[i][j]==0 使得 add1(4)==2且add2(2)>=0 或 add2(4)==2且add1(2)>=0 则着子

1. 冲4：

if ∃a[i][j]==0 使得 add1(5)==2, add1(4)==3且add2(2)>=0 或 add2(5)==2,add2（4）==2且add1(2)>=0 则着子

1. 活2：

if ∃a[i][j]==0 使得 add1(3)==1且add2(2)>=0 或 add2(3)==1且add1(2)>=0 则着子

1. 眠3：

if ∃a[i][j]==0 使得 add1(4)==1，add1(3)==2且add2(2)>=0 或 add2(4)==1, add1(3)==2且add1(2)>=0 则着子

1. 眠2：

if ∃a[i][j]==0 使得 add1(3)==1且add2(2)>=0 或 add2(3)==1且add1(2)>=0 则着子

//2018.3.16 Version 0.1.0