samurAI
READKICHHI

情報理工学コース　02141014 菊地　翔馬

2016 年 12 月 15 日

きっちーの課題

今回、わたくしきっちーに与えられた課題は、侍たちが占領→移動→隠伏の動作をしたとき、侍が隠伏している可能性のあるマスの全列挙である。よって、上記以外の行動、(占領→隠伏、隠伏したまま、etc...)) は想定している行動ではない。しかし、想定外の動作について検証しなくていいということではないので、他に考えるべきことは別の機会に考え、実装していきたいと思う。

想定している行動

上でも述べたように、占領→移動→隠伏の行動について考える。この中でも占領について少し細かく分類するべき行動がある。

0.1 無駄のない占領

占領のとる行動の中で一番可能性の高い行動である。侍が占領できる範囲全てを新しく自陣にする選択である。masakarin なら後ろ一マスを除いた 7 マス。

0.2 できるだけ無駄のない占領

後半になる程、相手の陣地も自分の陣地も増えていく無駄のない占領は難しくなっていき、少し無駄が生じることがある。前のターンまでに、一度自陣に占領してそのままのマスが占領できる範囲に存在する時である。なので、最悪の場合、1 マスしか自陣に変更できるマスがなかった、ということが想定される。

0.3 セグフォな占領

プログラムを書く際、セグフォを出させるような動きである。15 × 15 の二次元配列でフィールド情報を管理しているが、その配列外を占領しようとする行動がある。その場合もセグフォを出しながら想定した。

与えられる入力

変化したマスには 1 が代入されている。

関数の説明

typedef を使用しているので、以下は読み替えてほしい。

vector¡int¿ = vi

vector¡vector¡int¿¿ = vvi

0.4 vv_print(vvi)

二次元配列 vvi の出力である。基本的にデバック用。

0.5 init(&vvi)

二次元配列 vvi の初期化。15 × 15 の大きさを確保している。要素は全て 0 初期化している。

0.6 check0(vvi)

二次元配列 vvi の要素が全て 0 であるかどうかチェックする関数。後の関数内で使う。

0.7 rotation(int &x,int &y,int n)

二次元ベクトル (x,y) を n × 9 0 度回転させる関数。後の関数内で使う。計算方法については、「ベクトル　回転行列」とかでググってください。

0.8 masakari(int x,int y,vvi ,n)

この後の、katana,yari についても同様。与えられた vvi フィールドの (x,y) で、ある基準方向から n × 9 0 度回転した方向を向いたとき、占領できる範囲を占領したかどうか見る関数。その範囲のマスの値が 1 であれば、そのマスを 0 にする。dx,dy は (0,0) を中心としたとき、侍が占領できるマスである。tmp に探索する座標を加えることで、中心をずらしながら占領できるマスをずらすことができる。

0.9 masakari_check(int x,int y,vvi ,n)

この後の、katana_check,yari_check についても同様。上の masakari を使って、masakari の存在する可能性があるマスを探す。探す手順は、全座標を全て見てか、侍の方向を 90 度回転して、また全座標を見直す。もし check0 が true、つまりフィールドが全て 0 であれば占領しうる可能性があるということである。ad は侍が占領した時にいた位置から隣接する、隠伏できるマスである。隠伏できるマスに 1 に 1 を加える。

出力

二次元配列 exist の侍が存在しうるマスには数字が代入されている。数字が大きいほど隠伏する可能性は高い。

今後の課題

今回はとりあえず、侍が存在するマスを列挙することを目標にプログラムを実装した。入力や出力に関しては、実際のものと合うかどうかは知らない。なので、実際のプログラムに合致するように入力と出力を変更する必要がある。また今回は、同時に複数の変化があった時に対応していない。check0 が false になるからである。入力によっては変更するべき。そもそも変化が 1 で表されてる訳でもないと思うので治していきたい。