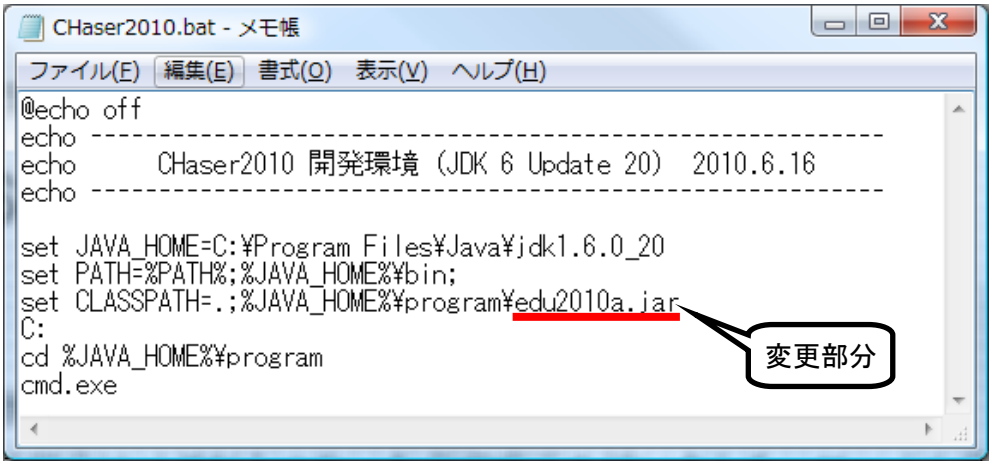
全国高校生プログラミングコンテスト

CHaser2010

ステップアップヒント２

8 競技部品（edu2010.jar）のバージョンアップについて

競技部品にバグがみつかったので修正を行いバージョンアップしました。ファイル名を「edu2010a.jar」に変更したのでバッチファイルの変更をお願いします。また、古い競技部品（edu2010.jar）は削除してください。今後さらにバージョンアップがあった場合はアルファベットを b、c という順で更新していきます。

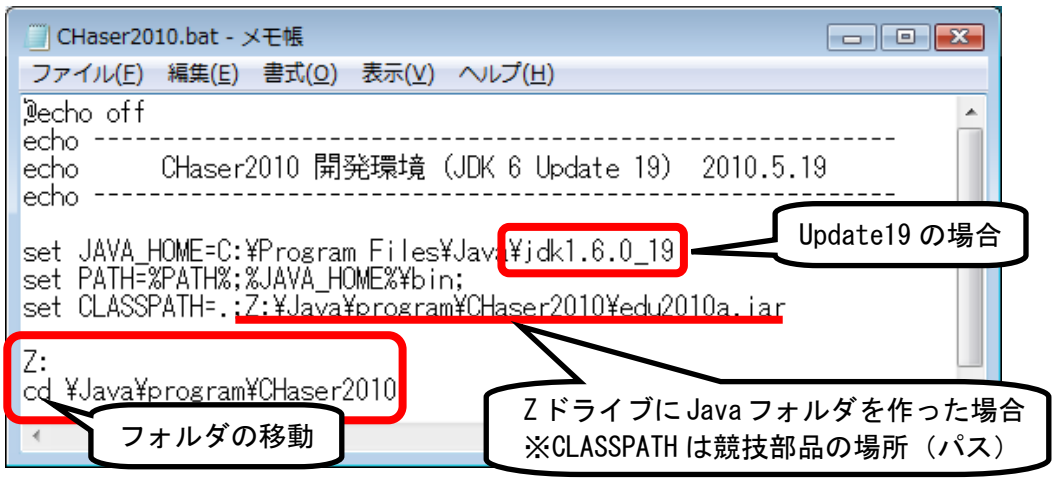


【画面 20】CHaser2010.bat の変更部分

9 開発環境について

ステップアップヒント１では、自宅などにあるパソコン（本人が管理者となるもの）を想定して JDK のインストールやフォルダの作成を行いました。しかし、学校のパソコンでは生徒が管理者の権限を持っていなかったり、JDK のバージョンも最新でない場合もあると思います。

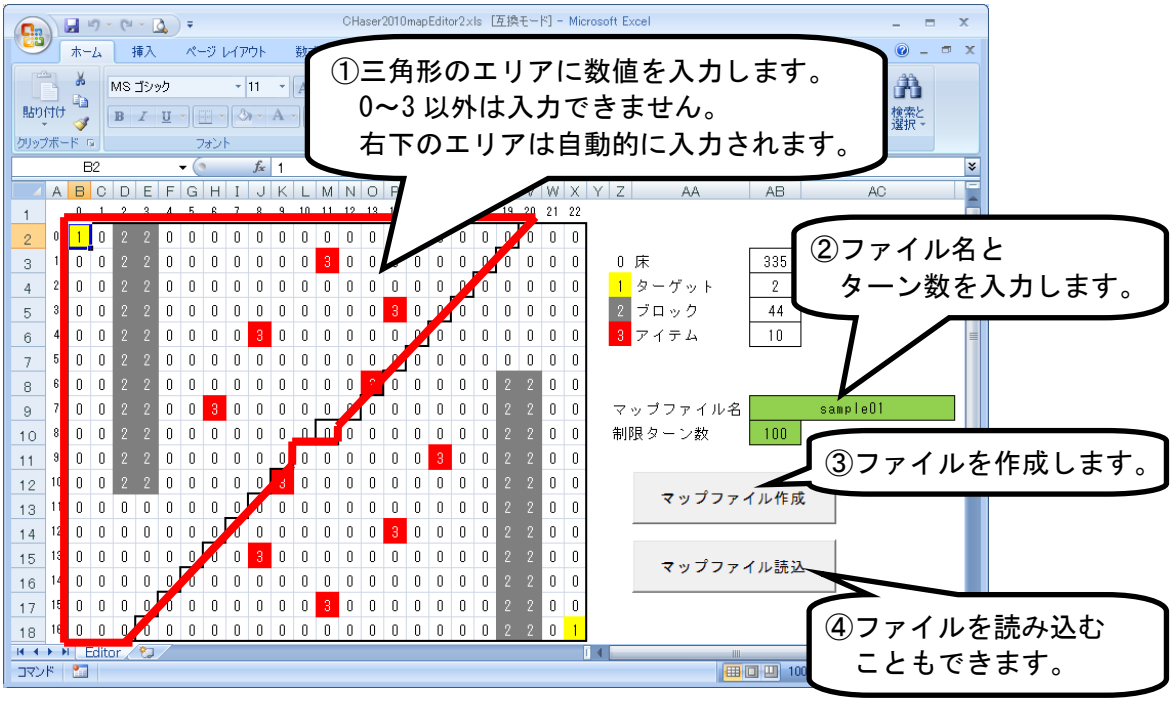
筆者の学校でも、生徒は管理者ではありませんし、JDK も Update19 と少し古いものを使っています。また、データはＣドライブでなく、Ｚドライブというネットワーク上のサーバに保存しています。このような環境の場合は、次のようにバッチファイルを変更してください。



【画面 21】筆者の学校でのバッチファイル

10 マップエディタについて

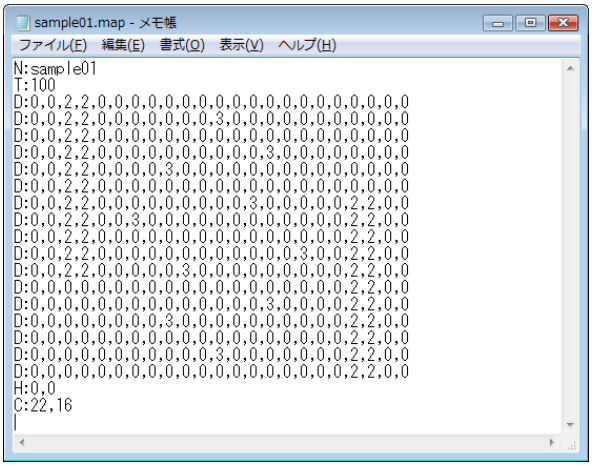
CHaser2010はマップファイルに記述された初期情報を読み込んでフィールドなどを設定する方式になりました。マップファイルはカンマ区切りのテキストファイルなのでメモ帳などで編集することもできますが、ここでは「マップエディタ」を全情研ホームページからダウンロードしてマップファイルを作成します。



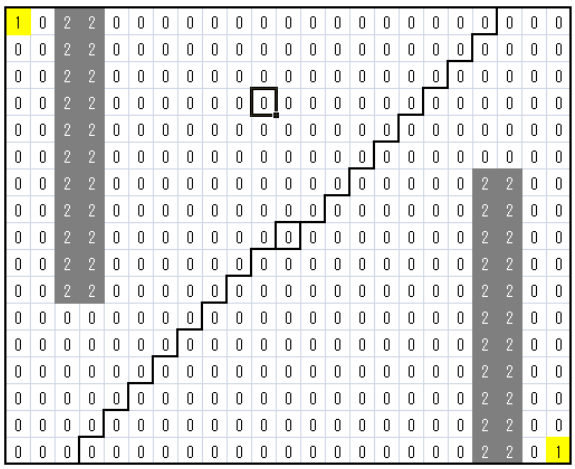
【画面 21】マップエディタの使い方

予戦は画面 23 と同じブロックの配置で行われ、スタート位置はクールが左上、ホットは右下に固定されます。アイテムをランダムに配置したマップファイルを複数用意しておき、ランダムに選んだフィールドで対戦させる予定です。

選手の皆さんも複数のマップファイルを作って、予戦に備えてください！



【画面 22】マップファイルの中身



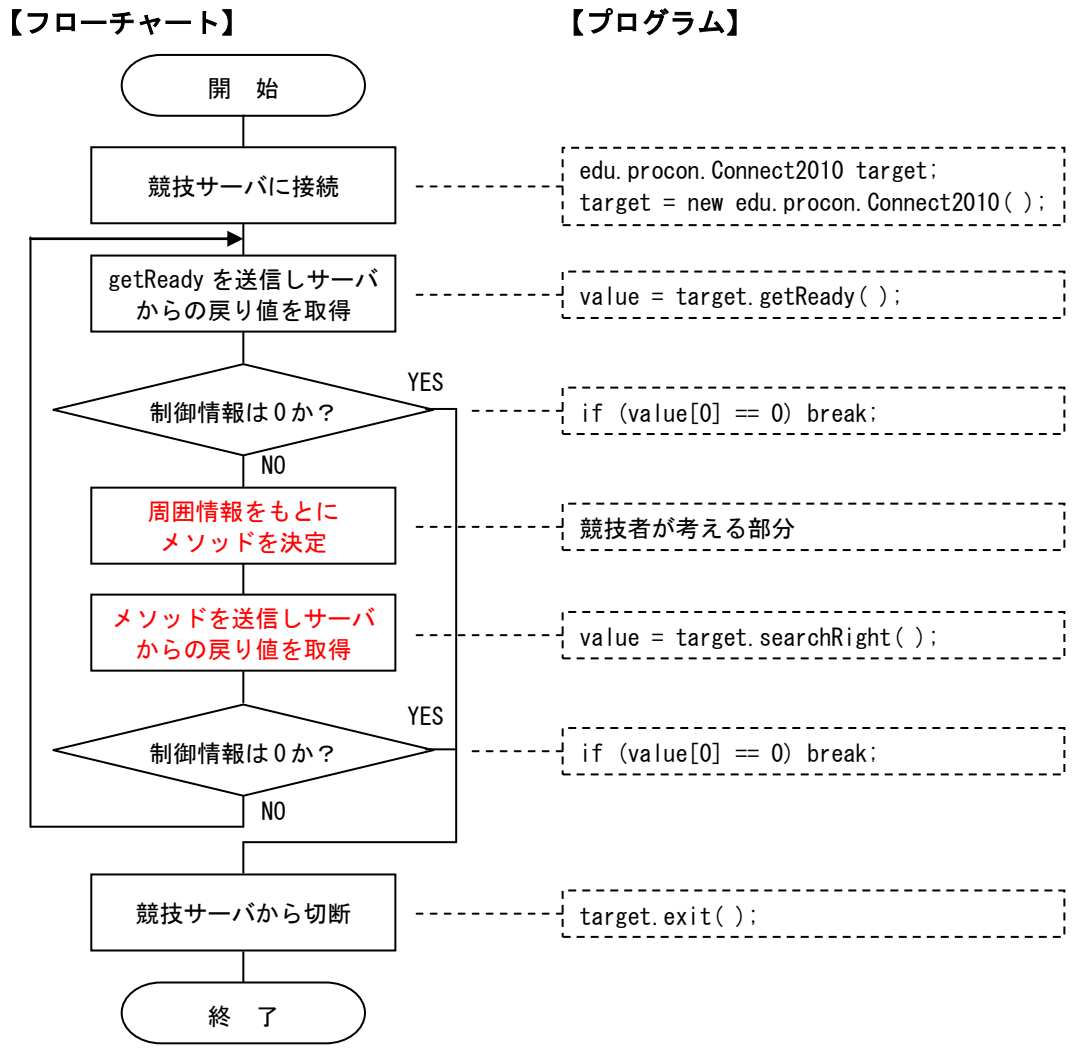
【画面 23】予戦のフィールド（アイテムなし）

11 プログラムの流れ

みなさんが作るクライアントプログラムと競技サーバのやりとりは、getReady メソッドを送信して周囲情報を取得した後、動作メソッド（walk 系、look 系、search 系、put 系）を選んで送信し最後に周囲情報を取得という流れを繰り返します。

従って、クライアントプログラムは図 2 のフローチャートのようになります。

みなさんは、サーバから得た周囲情報をもとにメソッドを選んで送信する部分を考えます。



【図 2】プログラムの大まかな流れ

12 サーバからの戻り値

getReady メソッドを送信した場合、サーバから送られてくる戻り値は、

制御情報（1）＋　周囲情報（9）＝　10 個の整数データ

となっています。戻り値は value という 10 個の領域を持つ配列に格納されます。

制御情報と周囲情報の値は表 1 と表 2 の通りです。

【表 1】制御情報の値

制御情報	
0	通信終了
1	通信続行

【表 2】周囲情報の値

周囲情報	
0	なし（床）
1	相手
2	ブロック
3	アイテム

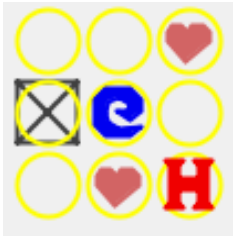
制御情報は 0 または 1 で、0 の場合、通信終了（試合終了）となり、exit メソッドでサーバから切断しプログラムを終了します。

周囲情報は、図 3 のように自分の周囲の 9 マスを左上から右下まで順番に値が格納されます。

例えば､図 4 の位置で getReady をした場合､表 2 より配列 value の値は図 5 のようになります。また、周囲 5（自分自身）は 0 となります。

周囲 1	周囲 2	周囲 3
周囲 4	周囲 5	周囲 6
周囲 7	周囲 8	周囲 9

【図 3】周囲情報の位置



0	0	3
2	0	0
0	3	1

【図 4】getReady をした場合の値

value[0]（制御情報）	1
〃 [1]（周囲 1）	0
〃 [2]（ 〃 2）	0
〃 [3]（ 〃 3）	3
〃 [4]（ 〃 4）	2
〃 [5]（ 〃 5）	0
〃 [6]（ 〃 6）	0
〃 [7]（ 〃 7）	0
〃 [8]（ 〃 8）	3
〃 [9]（ 〃 9）	1

【図 5】配列 value の値

※動作メソッド（walkUp など）を実行した後もサーバから戻り値が送信されます。順序は競技仕様（CHaser2010rule.pdf）をご覧ください。

13 サンプルプログラム２（モードを使ったプログラム）

①プログラムの保存先

program フォルダの下に「sample2010_02」という名前のフォルダを作り、次のプログラムを入力します。プログラム名は「sample2010_02.java」です。

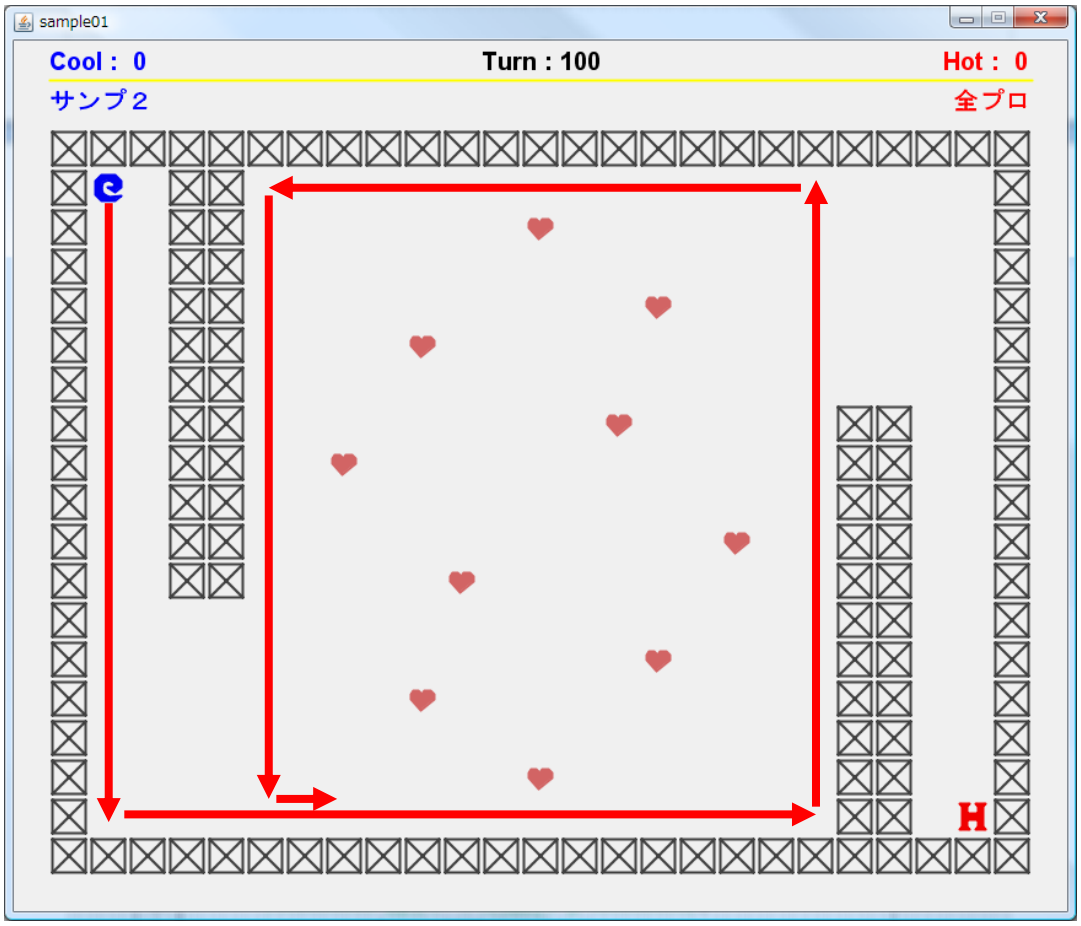
②プログラムの動作

・クール専用のプログラムです。
・まずブロックの直前まで下に移動します。
・次にブロックの直前まで右に移動します。以降はブロックの内側を反時計回り（上、左、下、右）に制限ターンになるまで移動を繰り返します。
・mode（モード）という変数を作り、動作の種類を記憶させています。

③対戦

・CHaser2010.bat を 3 つ起動させる。
・3 つの画面でそれぞれのコマンドを入力し、対戦させる。

サーバ　→　java　edu.procon.Server2010 －Fsample01.map
クール　→　java　sample2010_02 2009
ホット　→　java　sample2010_01 2010



【画面 24】対戦の様子

④サンプルプログラム２（sample2010_02.java）

```
/*********************************************************
  sample2010_02.java
*********************************************************/
public class sample2010_02 {
    public static void main(String[] args) {
        int[] value;
        value = new int[10];

        int mode = 1;

        /********** 競技サーバに接続する **********/
        edu.procon.Connect2010 target;
        target = new edu.procon.Connect2010("サンプ２", Integer.parseInt(args[0]));

        while (true) {
            value = target.getReady( );
            if (value[0] == 0) break;

            /********** mode の値で分岐する **********/
            switch (mode) {
                case 1:  // ブロック（壁）にぶつかるまで下に移動する
                    if(value[8] != 2){              // 下が壁でなければ、
                        value = target.walkDown( );  // 右に移動する
                    }
                    else{
                        value = target.walkRight( ); // 下が壁ならば、右に移動し、
                        mode = 2;                    // mode を 2 に変更する
                    }
                    break;
                case 2:  // ブロック（壁）にぶつかるまで右に移動する
                    if(value[6] != 2){
                        value = target.walkRight( );
                    }
                    else{
                        value = target.walkUp( );
                        mode = 3;
                    }
                    break;
                case 3:  // ブロック（壁）にぶつかるまで上に移動する
                    if(value[2] != 2){
                        value = target.walkUp( );
                    }
                    else{
                        value = target.walkLeft( );
                        mode = 4;
                    }
                    break;
                case 4:  // ブロック（壁）にぶつかるまで左に移動する
                    if(value[4] != 2){
                        value = target.walkLeft( );
                    }
                    else{
                        value = target.walkDown( );
                        mode = 1;
```

```
55                      }
56                      break;
57              }
58              /********** 制御情報が 0 だったら終了する **********/
59              if(value[0] == 0) break;
60          }
61
62          /********** 競技サーバから切断する **********/
63          target.exit( );
64      }
65  }
```

⑤プログラム説明

・変数の宣言

```
6       int[] value;
7       value = new int[10];
8
9       int mode = 1;
```

value は、サーバからの戻り値（制御情報 1＋周囲情報 9）を格納する配列です。

mode は、動作の種類を記憶させる変数です。mode を使うことで、どの動作をしているのかを管理することができます。サンプルプログラム 2 では表 3 のように 4 つのモードを用意しています。この方式ならば、今後プログラムを発展させるときモードを追加するだけで様々な動作ができるようになります。

【表 3】モードの種類と動作

mode	動　作
1	ブロック（壁）にぶつかるまで下に移動する
2	ブロック（壁）にぶつかるまで右に移動する
3	ブロック（壁）にぶつかるまで上に移動する
4	ブロック（壁）にぶつかるまで左に移動する

・競技サーバへの接続

```
11      /********** 競技サーバに接続する **********/
12      edu.procon.Connect2010 target;
13      target = new edu.procon.Connect2010("サンプ２", Integer.parseInt(args[0]));
```

edu.procon.Connect2010 クラスを使って競技サーバに接続します。Connect2010 の第 1 引数の文字列”サンプ２”はサーバに送信するチーム名で競技画面に表示されます。文字列は全角と半角の区別はなく 4 文字以内です。半角は全角に変換されます。また、5 文字以上の場合、先頭の 4 文字が有効です。

第 2 引数 Integer.parseInt(args[0])はプログラムの起動時に指定するポート番号ですコマンドライン引数を用いていますが直接「2009」のいうように定数を指定しても構いません。

また、ネットワーク上のサーバに接続する場合は IP アドレスを指定します。サーバの IP アドレスが 192.168.101.50 の場合、13 行目を次のように変更します。

```
target = new edu.procon.Connect2010("サンプ２", "192.168.101.50", Integer.parseInt(args[0]));
```

・getReady でサーバからの戻り値を得る

```
16      value = target.getReady( );
17      if (value[0] == 0) break;
```

getReady メソッドを送信するとサーバから戻り値が送信されてくるので、配列 value で受け取ります。value[0]に制御情報が格納されているので、0 だったら break し、while ループを抜けてプログラムを終了します。

・mode で分岐する

```
19      /********** mode の値で分岐する **********/
20      switch (mode) {
21          case 1:  // ブロック（壁）にぶつかるまで下に移動する
22              if(value[8] != 2){              // 下が壁でなければ、
23                  value = target.walkDown( );  // 右に移動する
24              }
25              else{
26                  value = target.walkRight( ); // 下が壁ならば、右に移動し、
27                  mode = 2;                    // mode を 2 に変更する
28              }
29              break;
30          case 2:  // ブロック（壁）にぶつかるまで右に移動する
                .
                .
                .
```

switch 文を使って mode の値で分岐を行います。mode = 1 の場合、ブロック（壁）にぶつかるまで下に移動するので、自分のすぐ下にブロックがあるかを if(value[8] == 2)で判断します。

もし、下がブロックだった場合は次に右に移動するので walkRight メソッドを送信し、mode を 2 にします。ブロックでなければ、walkDown メソッドを送信し上に移動します。同様に mode2 から 4 を作ります。

・制御情報が 0 だったら終了する

```
58      /********** 制御情報が 0 だったら終了する **********/
59      if(value[0] == 0) break;
```

メソッドを送信するとサーバから戻り値が返ってくるので、終了の判定をします。

・競技サーバから切断する

```
62      /********** 競技サーバから切断する **********/
63      target.exit( );
```

exit メソッドで競技サーバから切断します。

⑥注意

このプログラムは、右上からスタートするクール専用です。左下からスタートするホットで実行するとすぐに下のブロックにぶつかってしまいゲームオーバーとなります。また、モードが切り替わる際、移動する方向にブロックがあるかのチェックをしていないので、ブロックにぶつかってしまう場合があります。

これを回避するためにプログラム中で判断を追加しなければなりません。どこにどのような判断を追加したらよいか考えてみてください。