

プログラミング演習 IIIA

msweeper

21-403 伊藤幸太郎

2024 年 7 月 9 日

第 1 章

プログラムの仕様

1.1 遊び方

1.1.1 ルール

msweeper は、盤面上に隠れている M を取り除くという趣旨のゲームである。盤面には、M がある点と無い点が存在する。ゲーム開始時にはこれらの情報はすべて隠されている。

プレイヤーは、盤面上の任意の点を開くことができる。ある点 P を開くと以下のいずれかが発生する。

P に M が無いとき

P の周囲の点に隠されている M の数が公開される。ここでいう周囲の点とは、以下のように P に隣接する 8 点である。Q がボード外にはみ出すような場所に P が位置する場合、はみ出す Q は周囲の点に含まない。

Q0	Q1	Q2
Q3	P	Q4
Q5	Q6	Q7

図 1.1 P の周囲の点の位置関係

P に M があるとき

ゲームオーバーとなる。

M のある点を開かず、全ての M がない点（安全な点）を開くことができれば、ゲームクリアである。

1.1.2 使用方法

ゲームを開始すると、以下のように盤面が表示され、入力待ち状態になる。

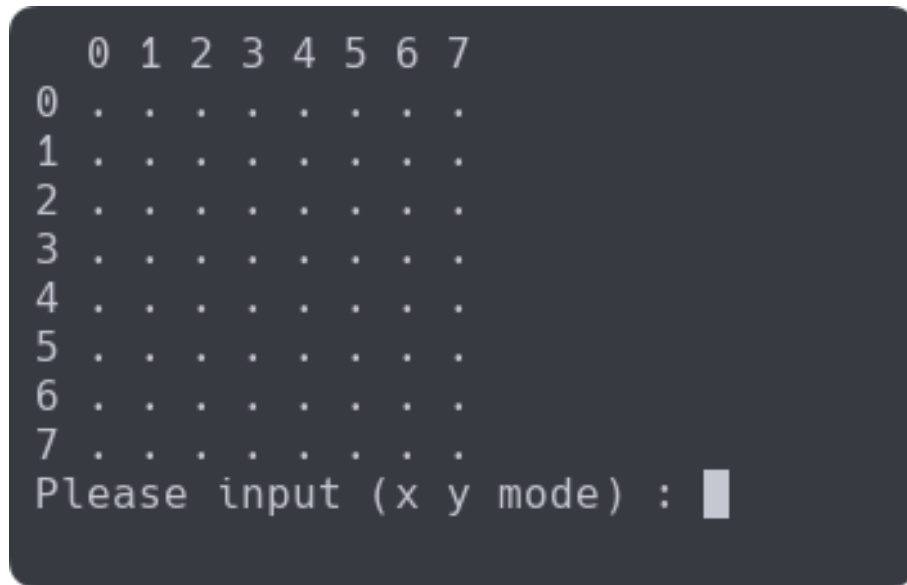


図 1.2 ゲーム開始時の画面

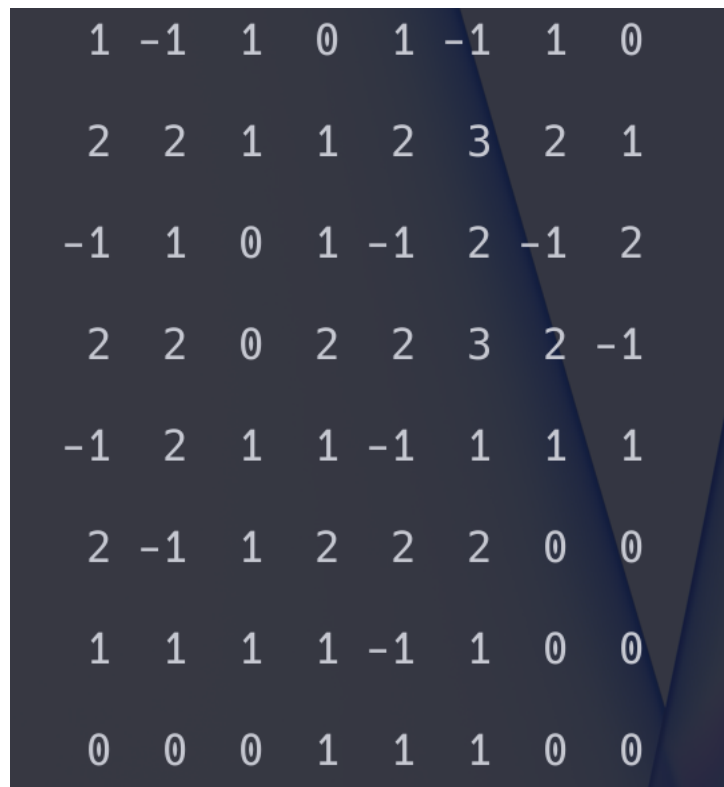
x は横方向の座標、y は縦方向の座標を入力する。mode は 3 種類あり、それぞれ

- s (x, y) の点を開ける。
- m M があると思う点に M マークを付ける。
- q M があるか無いか不明なときに?マークを付ける。

という機能である。なお、マークが付いている状態で再度同じ点に対して m や q を用いると、マークが外れる。

1.2 データ構造

まず、M と数字が配置された盤面（答えの盤面）を表現するために、`int` 型の二次元配列 `board` を用意し、データを格納した。M がある点は `-1` とし、M がない点は周囲の点にある M をカウントした数を格納した。また、開いた場所は `-2` とした。これにより後述するクリアしているかチェックする関数にて、二次元配列の全ての要素が `0` 未満であればクリアしているという判定手法を取ることができる。



1	-1	1	0	1	-1	1	0
2	2	1	1	2	3	2	1
-1	1	0	1	-1	2	-1	2
2	2	0	2	2	3	2	-1
-1	2	1	1	-1	1	1	1
2	-1	1	2	2	2	0	0
1	1	1	1	-1	1	0	0
0	0	0	1	1	1	0	0

図 1.3 生成された盤面の例

また、実際に盤面を表示するために `char` 型の二次元配列 `disp_board` を用意した。初期状態ではドットのみが格納されており、隠されていることを示している。そしてプレイヤーが開くたびに、`board` の対応した位置の数字を文字として格納するようにした。

1.3 各関数の仕様

1.3.1 put_M()

■書式 `void put_M(int board[8][8]);`

■引数 答えの盤面を表現する `int` 型の二次元配列

■処理内容 盤面上に M を配置する。

1.3.2 put_num()

■書式 `void put_num(int board[8][8]);`

■引数 答えの盤面を表現する `int` 型の二次元配列

■処理内容 盤面上にヒントの数字を配置する

1.3.3 start_disp()

■書式 `void start_disp(char disp_board[8][8]);`

■引数 盤面表示用の `char` 型の二次元配列

■処理内容 盤面表示用の `char` 型の二次元配列の全ての要素に対して、初期のすべての点が隠れている状態を示すドットを代入する。また、座標の数字とともに盤面として表示させる。

1.3.4 game()

■書式 `void game();`

■処理内容 ゲームに関わるすべての動作を行う。ある関数の戻り値に応じて他の関数を実行させる。また、ゲームクリア及びゲームオーバー処理を行う。

1.3.5 disp()

■書式 `void disp(int x, int y, char disp_board[8][8]);`

■引数 座標、盤面表示用の `char` 型の二次元配列

■処理内容 ゲーム進行中の盤面の表示を行う。

1.3.6 input()

■書式 `char input(int *x, int *y);`

■引数 座標

■戻り値 入力されたモードの文字

■処理内容 標準入力から入力を行う。

1.3.7 mark_m()

■書式 void mark_m(int x, int y, int board[8][8], char disp_board[8][8]);

■引数 座標、答えの盤面を表現する int 型の二次元配列、盤面表示用の char 型の二次元配列

■処理内容 開かれてない点に M マークを付ける。既に付いている場所であれば、マークを取り消す。

1.3.8 mark_question()

■書式 void mark_question(int x, int y, int board[8][8], char disp_board[8][8]);

■引数 座標、答えの盤面を表現する int 型の二次元配列、盤面表示用の char 型の二次元配列

■処理内容 開かれてない点に?マークを付ける。既に付いている場所であれば、マークを取り消す。

1.3.9 open()

■書式 int open(int x, int y, int board[8][8], char disp_board[8][8]);

■引数 座標、答えの盤面を表現する int 型の二次元配列、盤面表示用の char 型の二次元配列

■戻り値 開いた場所が M なら 1、そうでなければ 0

1.3.10 check_clear()

■書式 int check_clear(int board[8][8]);

■引数 答えの盤面を表現する int 型の二次元配列

■戻り値 クリアしていたら 0、そうでなければ 1

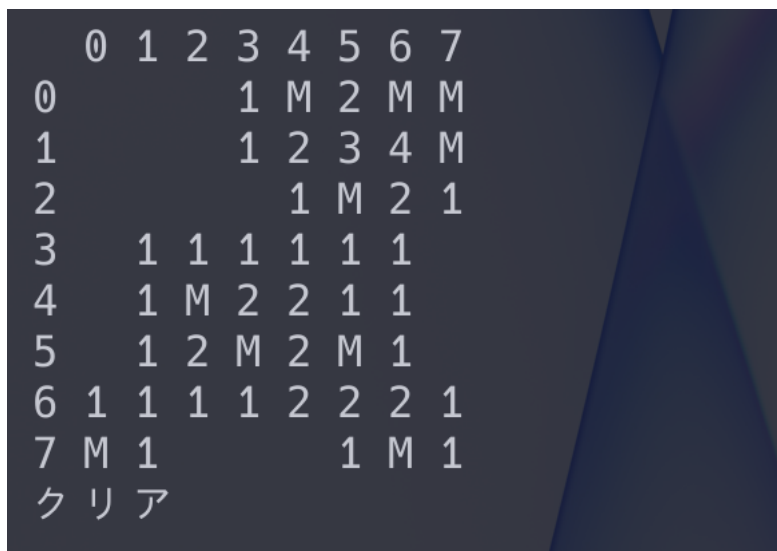
第 2 章

プログラムの正しさの検証

2.1 ゲームクリア判定

ゲームクリアの判定は、`check_clear()`で行っている。盤面上では M は -1、開いた点は -2 としているため、すべての点が 0 未満であればクリアしている。これを用いて判定を行っている。

実際にプレイして正常にクリアの判定と終了処理が行われた様子を以下に示す。



	0	1	2	3	4	5	6	7
0				1	M	2	M	M
1				1	2	3	4	M
2					1	M	2	1
3		1	1	1	1	1	1	
4		1	M	2	2	1	1	
5		1	2	M	2	M	1	
6	1	1	1	1	2	2	2	1
7	M	1				1	M	1
ク リ ア								

図 2.1 ゲームクリアした様子

2.2 ゲームオーバー判定

ゲームオーバーの判定は、`open()`の戻り値によって行っている。先述の通り、戻り値が 1 であれば開いた点が M であるため、その時点でゲームオーバーである。終了処理は `game()` にて行っている。

実際にプレイし、正常にゲームオーバーの判定と終了処理が行われた様子を以下に示す。

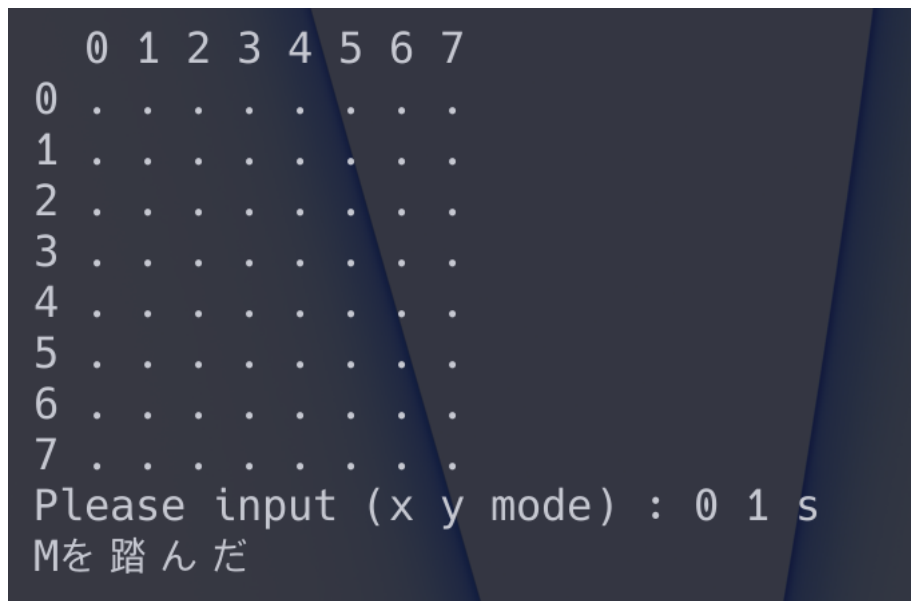


図 2.2 ゲームオーバーした様子

2.3 正しくない入力への対処

盤面の範囲外の座標を指定したり、存在しないモードが指定された場合、以下のようにエラーと表示し、もう一度入力させる。

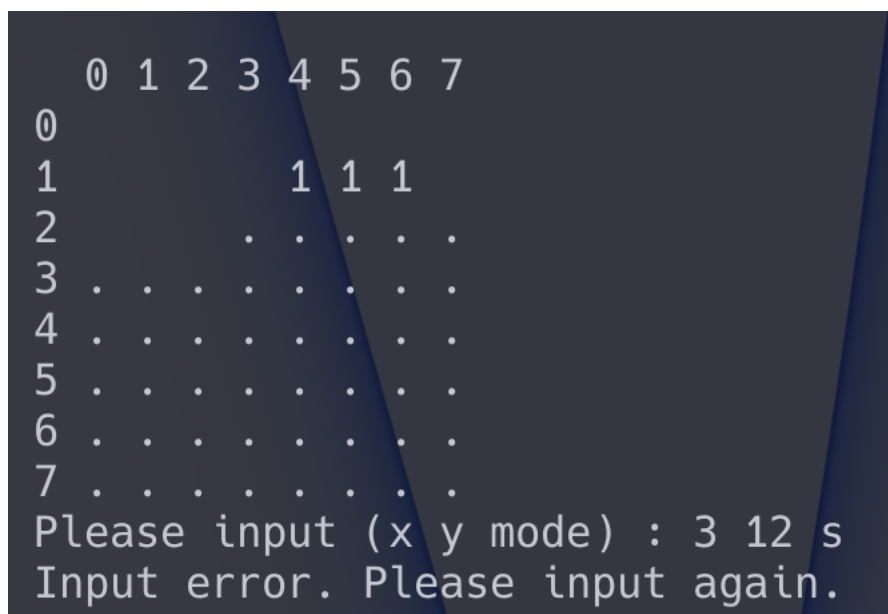


図 2.3 入力エラーの様子

第 3 章

改善点

3.1 予期せぬ入力に対しての弱さ

今回のプログラムでは、入力を `scanf()` で行っている。そのため、予期せぬ入力（例：4 s 2 など、2 つ目のパラメータに文字が入力される等）がされたときに、異常な動作をしてしまう。これを解決するためには、`fgets()` を利用する等、`scanf()` を使わない手法を用いる必要がある。

3.2 非効率な M の位置

`put_M()` にて M を配置する際、`rand()` を用いて位置を決定している。しかしこの方法だと、既に配置されているところに再びぶつかってしまい、再度 `rand()` を実行しないといけない可能性が高くなる。これを解決するためには、行 × 列のサイズの配列を用意し、その中に配列のインデックスと同じ数字を格納する。そしてその配列をシャッフルし、配列内の数字を順に用いて位置を決定することによって、解決できると考えられる。

3.3 データ構造の複雑さ

今回のプログラムでは、2 種類の二次元配列を利用し、それぞれをリンクさせながら盤面を表現した。しかし、構造が少々複雑で分かりづらい。また、マークが付いているか付いていないかは、実際に表示用の盤面に文字が格納されているかどうかで判定しており、スマートなやり方では無い。

改善策としては、構造体を活用することによって、答えの盤面の情報と、表示用の盤面の情報をリンクしやすくなったり、マークが付いているかどうかを `flag` 等で判別することができ、よりわかりやすくスマートな実装にできると考えられる。また、ユーザーが盤面のサイズを変更できる機能を実装する際にも、扱いやすくなると考えられる。

付録 A

プログラムリスト

Listing A.1 j21403.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 void put_M(int board[8][8]);
6 void put_num(int board[8][8]);
7 void start_disp(char disp_board[8][8]);
8
9 void game();
10 void disp(int x, int y, char disp_board[8][8]);
11 char input(int *x, int *y);
12 void mark_m(int x, int y, int board[8][8], char disp_board[8][8]);
13 void mark_question(int x, int y, int board[8][8], char disp_board[8][8]);
14 int open(int x, int y, int board[8][8], char disp_board[8][8]);
15 int check_clear(int board[8][8]);
16
17 int main(void) {
18     game();
19     return 0;
20 }
21
22 void game() {
23     int x, y, flag = 0;
24     char mode;
25     int board[8][8];
26     char disp_board[8][8];
27     for(int i = 0; i < 8; i++) {
28         for(int j = 0; j < 8; j++) {
29             board[i][j] = 0;
30         }
31     }
32     put_M(board);
33     put_num(board);
34     start_disp(disp_board);
35     while(!flag) {
36         mode = input(&x, &y);
37         if(mode == 'm') {
38             mark_m(y, x, board, disp_board);
39             disp(y, x, disp_board);
40         } else if(mode == 'q') {
```

```

41         mark_question(y, x, board, disp_board);
42         disp(y, x, disp_board);
43     } else {
44         if(open(y, x, board, disp_board) == 1) {
45             printf("Mを踏んだ\n");
46             break;
47         } else {
48             disp(y, x, disp_board);
49         }
50         if(check_clear(board) == 0){
51             //クリア
52             flag = 1;
53             printf("クリア\n");
54         } else {
55             continue;
56         }
57     }
58 }
59 }
60
61 void disp(int x, int y, char disp_board[8][8]) {
62     printf(" 0 1 2 3 4 5 6 7\n");
63     for(int i = 0; i < 8; i++) {
64         printf("%d ", i);
65         for(int j = 0; j < 8; j++) {
66             printf("%c ", disp_board[i][j]);
67         }
68         printf("\n");
69     }
70 }
71
72 char input(int *x, int *y) {
73     char mode;
74     while(1){
75         printf("Please input (x y mode) : ");
76         scanf("%d %d %c", x, y, &mode);
77         if(*x < 0 || *x > 7 || *y < 0 || *y > 7 || (mode != 's' && mode != 'm' && mode != 'q'
78     )) {
79             printf("Input error. Please input again.\n");
80         } else {
81             break;
82         }
83     }
84     return mode;
85 }
86
87 int open(int x, int y, int board[8][8], char disp_board[8][8]) {
88     //開いた場所は-2とする
89     if(board[x][y] == -1) {
90         //gameover
91         return 1;
92     } else if(board[x][y] == 0) {
93         board[x][y] = -2;
94         disp_board[x][y] = ' ';
95     } else if(board[x][y] == -2) {
96     } else {
97         disp_board[x][y] = '0' + board[x][y];

```

```

97         board[x][y] = -2;
98     }
99     return 0;
100 }
101
102 void mark_m(int x, int y, int board[8][8], char disp_board[8][8]) {
103     if(disp_board[x][y] == 'M') {
104         disp_board[x][y] = '.';
105     } else if(board[x][y] == -2) {
106     } else {
107         disp_board[x][y] = 'M';
108     }
109 }
110
111 void mark_question(int x, int y, int board[8][8], char disp_board[8][8]) {
112     if(disp_board[x][y] == '?') {
113         disp_board[x][y] = '.';
114     } else if(board[x][y] == -2) {
115     } else {
116         disp_board[x][y] = '?';
117     }
118 }
119
120 int check_clear(int board[8][8]) {
121     int flag = 0;
122     for(int i = 0; i < 8; i++) {
123         for(int j = 0; j < 8; j++) {
124             if(board[j][i] >= 0) {
125                 flag = 1;
126                 break;
127             }
128         }
129         if(flag) return 1;
130     }
131     return 0;
132 }
133
134 void start_disp(char disp_board[8][8]) {
135     printf(" 0 1 2 3 4 5 6 7\n");
136     for(int i = 0; i < 8; i++) {
137         for(int j = 0; j < 8; j++) {
138             disp_board[i][j] = '.';
139         }
140     }
141     for(int i = 0; i < 8; i++) {
142         printf("%d ", i);
143         for(int j = 0; j < 8; j++) {
144             printf("%c ", disp_board[i][j]);
145         }
146         printf("\n");
147     }
148 }
149
150 void put_M(int board[8][8]) {
151     srand((unsigned)time(NULL));
152     int row, column;
153     for(int i = 0; i < 10; i++) {

```

```

154         do {
155             row = rand() % 8;
156             column = rand() % 8;
157         } while(board[row][column] == -1);
158         board[row][column] = -1;
159     }
160 }
161
162 void put_num(int board[8][8]) {
163     for(int i = 0; i < 8; i++) {
164         for(int j = 0; j < 8; j++) {
165             //(i,j)がMだったときのスキップ処理
166             if(board[i][j] == -1) {
167                 continue;
168             }
169             //(i,j)の周りのMの数をカウントし、(i,j)をインクリメント
170             for(int k = -1; k <= 1; k++) {
171                 for(int l = -1; l <= 1; l++) {
172                     if(i+k < 0 || i+k > 7 || j+l < 0 || j+l > 7){
173                         continue;
174                     }
175                     if(board[i+k][j+l] == -1) {
176                         board[i][j] += 1;
177                     }
178                 }
179             }
180         }
181     }
182 }

```