Tic Tac Toe (CUI - C Language)

S.Matoike

目次

第 1 章

C 言語による三目並べ [Tic Tac Toe]

1.1　制作する三目並べ [Tic Tac Toe] の概要

プログラムを実行すると、盤面が表示され、×の石を置く場所を指定するよう促されます

画面上に示された番号を入力すると、その番号のスロットに×の石が置かれた盤面が表示され、次の手番の○に、石を置く場所を指定するように促されます

手番を交互に変えながらゲームは進み、縦、横、斜めの何れかに、先に一列に自分の石を並べた方が勝ちとなります

既に石の置かれているスロット番号を指定できませんし、スロット番号として 0 〜 8 以外の数値を指定することもできません

```
スタート！ [Tic Tac Toe]
/---|---|---\
| 0 | 1 | 2 |
|---|---|---|
| 3 | 4 | 5 |
|---|---|---|
| 6 | 7 | 8 |
\---|---|---/
'X' さんの turn です
石を置く場所 0 〜 8 を指定して下さい : 4
/---|---|---\
| 0 | 1 | 2 |
|---|---|---|
| 3 | X | 5 |
|---|---|---|
| 6 | 7 | 8 |
\---|---|---/
'O' さんの turn です
石を置く場所 0 〜 8 を指定して下さい : 2
/---|---|---\
| 0 | 1 | O |
|---|---|---|
| 3 | X | 5 |
|---|---|---|
| 6 | 7 | 8 |
\---|---|---/
'X' さんの turn です
石を置く場所 0 〜 8 を指定して下さい :
```

1.2　定数定義と関数プロトタイプ

printf 文 で盤面を標準出力（コンソール画面）に表示し、scanf 文 で石を置く場所を標準入力（キーボード）から受け取るので、 stdio.h を include します

文字列の比較を行う関数 strcmp() を使うので、 string.h を include します

これから作成する関数サブプログラムの関数プロトタイプを、予め宣言しておきます

盤面 board[9] は、左上から右下に向かって 0 から 8 までの 9 個の整数の配列で表現し、石を置く場所はこの番号で指定させるようにします

盤面に○の石が置かれたなら、board[9] 配列の中の指定された要素に、MARU を代入し、×の石は BATSU を代入することにします

勝ち負けの判定を行う際、引き分けを DRAW 、まだ勝敗がついていない場合（ゲーム継続中）は、NEXT という値を使うことにします

ソースコード 1.1　関数プロトタイプなど：Tic Tac Toe

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int lineSum(int, int, int);
5  int switchTurn(int);
6  void printBoard();
7  int slotNum(int);
8  int checkWinner();
9  void result(int);
10
11 int board[]={0,1,2,3,4,5,6,7,8};
12 #define MARU 10
13 #define BATSU -10
14 #define DRAW 100
15 #define NEXT 200
```

手番（turn）は、switchTurn() 関数が呼ばれる度に、MARU と BATSU を交互に割り当てて返します

ソースコード 1.2　手番の交代：Tic Tac Toe

```
1  int switchTurn(int turn){
2      if(turn==BATSU) return MARU;
3      return BATSU;
4  }
```

1.3　主処理

①盤面の表示、②次の手の入力、③入力された手の配置、④手番の交代、⑤勝敗の判定、を繰り返します

board[9] の各要素に予め振られた整数値の指定された場所に、手番 turn（MARU 又は BATSU をとる整数変数）を代入して、盤面に石を配置します

ソースコード 1.3　main 関数：Tic Tac Toe

```
int main(int argc, char *argv[]) {
    /* 先手後手を決定 */
    int turn=BATSU;
    if(1<argc){
        if( !strcmp(argv[1], "-r") ) turn = MARU;
    }
    printf("スタート！ [Tic Tac Toe]\n");
    int winner;
    do{
        /* ①盤面の表示 */
        printBoard();
        /* ②手を入力 */
        int num = slotNum(turn);
        /* ③手を盤面に配置 */
        board[num] = turn;
        /* ④手番の交代 */
        turn = switchTurn(turn);
        /* ⑤勝敗の判定 */
        winner=checkWinner();
    }while(winner==NEXT);
    /* 対戦結果の表示 */
    printBoard();
    result(winner);
    return 0;
}
```

1.4　盤面の表示

board[9] の配列には、MARU と BATSU の整数値が、それぞれの石の置かれたスロットに入っており、まだ石の置かれていないところには、初期値であるスロット番号が入っています

ゼロの文字コードからのオフセットを、ゼロの文字コードに加えるという方法で、数字 0 〜 8 の文字コードを生成しています

ソースコード 1.4　盤面の表示：Tic Tac Toe

```
void printBoard(){
    char bd[9];
    int i;
    for(i=0; i<9; i++){
        if(board[i]==MARU)         bd[i]='O';
        else if(board[i]==BATSU)   bd[i]='X';
        else                       bd[i]='0' + i;
    }
    printf("\n/---|---|---\\\n");
    printf("| %c | %c | %c |\n", bd[0],bd[1],bd[2]);
    printf("|---|---|---|\n");
    printf("| %c | %c | %c |\n", bd[3],bd[4],bd[5]);
    printf("|---|---|---|\n");
    printf("| %c | %c | %c |\n", bd[6],bd[7],bd[8]);
    printf("\\---|---|---/\n");
}
```

1.5　結果の表示

勝敗判定の結果を受け取り、それによって結果を表示しています

ソースコード 1.5　結果の表示：Tic Tac Toe

```
1  void result(int winner){
2      printf("\n");
3      switch(winner){
4              case DRAW:   printf("引き分け\t");     break;
5              case MARU:   printf("'O' の勝ち\t");  break;
6              case BATSU:  printf("'X' の勝ち\t");  break;
7      }
8      printf("またね！\n");
9  }
```

1.6　手の入力

0 〜 8 以外が入力された場合に再入力を促しています
盤面上に既に石が配置されているスロットを指定された場合にも、再入力を促します

ソースコード 1.6　手の入力：Tic Tac Toe

```
1  int slotNum(int turn){
2      char* fig="";
3      if(turn==MARU)              fig="'O'";
4      else if(turn==BATSU)        fig="'X'";
5      int num;
6      do{
7          printf("\n %s さんのturnです\n石を置く場所 0 〜 8 を指定して下さい：", fig
               );
8          /* while( getchar() != '\n' );  *//* 標準入力バッファのクリア */
9          scanf("%d", &num);
10         if(!(0<=num && num<9)){
11             printf("再指定：0 〜 8 を指定して下さい");
12             continue;
13         }
14         if(board[num]!=num){
15             printf("再指定：そこには既に石が置かれています\n");
16             continue;
17         }
18         break;
19     }while( 1 );
20     return num;
21 }
```

1.7　勝敗の判定

横の 3 行、縦の 3 列、2 つの斜め線のそれぞれのライン上で、 board[9] 配列の該当する要素を合計し、MARU が 3 つ並んでいたら合計が MARU*3 になり、BATSU が 3 つ並んだら合計は BATSU*3 にな

るはず、として判定しています（勝った方、 MARU か BATSU を返しています）
盤面の配列に 0 〜 8 のどれかが残っていたなら、それはまだ試合が継続中なので、NEXT を返します
盤面の配列に 0 〜 8 が残っていないのに勝敗が決まっていない場合は、引き分け DRAW を返します

ソースコード 1.7　勝敗の判定：Tic Tac Toe

```
1  int lineSum(int n1, int n2, int n3){
2      return board[n1] + board[n2] + board[n3];
3  }
4
5  int checkWinner(){
6     int i, line=0;
7     for(i=0; i<8; i++){
8         switch(i){
9            case 0: line=lineSum(0, 1, 2); break;
10           case 1: line=lineSum(3, 4, 5); break;
11           case 2: line=lineSum(6, 7, 8); break;
12           case 3: line=lineSum(0, 3, 6); break;
13           case 4: line=lineSum(1, 4, 7); break;
14           case 5: line=lineSum(2, 5, 8); break;
15           case 6: line=lineSum(0, 4, 8); break;
16           case 7: line=lineSum(2, 4, 6); break;
17        }
18        if(line==3*MARU)          return MARU;
19        else if(line==3*BATSU)    return BATSU;
20    }
21    for(i=0; i<9; i++){
22       if(0<=board[i] && board[i]<9)return NEXT;
23    }
24    return DRAW;
25 }
```

1.8　プログラムの最終的な形

ソースコード 1.8　完成：Tic Tac Toe

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int lineSum(int, int, int);
5  int switchTurn(int);
6  void printBoard();
7  int slotNum(int);
8  int checkWinner();
9  void result(int);
10
11 int board[]={0,1,2,3,4,5,6,7,8};
12 #define MARU 10
13 #define BATSU -10
14 #define DRAW 100
15 #define NEXT 200
16
17 int switchTurn(int turn){
18     if(turn==BATSU) return MARU;
```

```
19      return BATSU;
20  }
21
22  int main(int argc, char *argv[]) {
23      /* 先手後手を決定 */
24      int turn=BATSU;
25      if(1<argc){
26          if( !strcmp(argv[1], "-r") ) turn = MARU;
27      }
28      printf("スタート！ [Tic Tac Toe]\n");
29      int winner;
30      do{
31          /* ①盤面の表示 */
32          printBoard();
33          /* ②手を入力 */
34          int num = slotNum(turn);
35          /* ③手を盤面に配置 */
36          board[num] = turn;
37          /* ④手番の交代 */
38          turn = switchTurn(turn);
39          /* ⑤勝敗の判定 */
40          winner=checkWinner();
41      }while(winner==NEXT);
42      /* 対戦結果の表示 */
43      printBoard();
44      result(winner);
45      return 0;
46  }
47
48  void printBoard(){
49      char bd[9];
50      int i;
51      for(i=0; i<9; i++){
52          if(board[i]==MARU)          bd[i]='O';
53          else if(board[i]==BATSU)    bd[i]='X';
54          else                        bd[i]='0' + i;
55      }
56      printf("\n/---|---|---\\\n");
57      printf("| %c | %c | %c |\n", bd[0],bd[1],bd[2]);
58      printf("|---|---|---|\n");
59      printf("| %c | %c | %c |\n", bd[3],bd[4],bd[5]);
60      printf("|---|---|---|\n");
61      printf("| %c | %c | %c |\n", bd[6],bd[7],bd[8]);
62      printf("\\---|---|---/\n");
63  }
64
65  void result(int winner){
66      printf("\n");
67      switch(winner){
68              case DRAW:   printf("引き分け\t");     break;
69              case MARU:   printf("'O' の勝ち\t");   break;
70              case BATSU:  printf("'X' の勝ち\t");   break;
71      }
72      printf("またね！\n");
73  }
74
75  int slotNum(int turn){
76      char* fig="";
77      if(turn==MARU)              fig="'O'";
```

```
78      else if(turn==BATSU)        fig="'X'";
79      int num;
80      do{
81          printf("\n %s さんのturnです\n石を置く場所 0 ～ 8 を指定して下さい：", fig
                );
82          /* while( getchar() != '\n' );  *//* 標準入力バッファのクリア */
83          scanf("%d", &num);
84          if(!(0<=num && num<9)){
85              printf("再指定：0 ～ 8 を指定して下さい");
86              continue;
87          }
88          if(board[num]!=num){
89              printf("再指定：そこには既に石が置かれています\n");
90              continue;
91          }
92          break;
93      }while( 1 );
94      return num;
95  }
96
97  int lineSum(int n1, int n2, int n3){
98      return board[n1] + board[n2] + board[n3];
99  }
100
101 int checkWinner(){
102     int i, line=0;
103     for(i=0; i<8; i++){
104         switch(i){
105             case 0: line=lineSum(0, 1, 2); break;
106             case 1: line=lineSum(3, 4, 5); break;
107             case 2: line=lineSum(6, 7, 8); break;
108             case 3: line=lineSum(0, 3, 6); break;
109             case 4: line=lineSum(1, 4, 7); break;
110             case 5: line=lineSum(2, 5, 8); break;
111             case 6: line=lineSum(0, 4, 8); break;
112             case 7: line=lineSum(2, 4, 6); break;
113         }
114         if(line==3*MARU)          return MARU;
115         else if(line==3*BATSU)    return BATSU;
116     }
117     for(i=0; i<9; i++){
118        if(0<=board[i] && board[i]<9)return NEXT;
119     }
120     return DRAW;
121 }
```

参考文献

[1]