C 言語：スライド・パズル（15Puzzle）

○×工業高校 機械工学科 2 年

2023 年 10 月 11 日

1　ゲームの概要

4 × 4 に区切った盤面上の各タイルに、0〜15 の番号が割り振られている。0 が割り振られたタイルは空欄になっていて、他のタイルはその空欄にスライドさせて移動することができる。最初は不規則に並べられている盤面ですが、空欄の上下、あるいは空欄の左右のタイルを選んでは、空欄の方向にスライドさせる事によって、最終的に 1 から 15 まで規則正しく並んだ盤面状態を目指すゲームである。

```
% ./slidetile

[11][ 3][ 6][10]
[ 9][ 1][ 8][14]
[ 2][ 4][ 7][12]
[13][15][  ][ 5]

Select number:6

[11][ 3][  ][10]
[ 9][ 1][ 6][14]
[ 2][ 4][ 8][12]
[13][15][ 7][ 5]

Select number:11

[  ][11][ 3][10]
[ 9][ 1][ 6][14]
[ 2][ 4][ 8][12]
[13][15][ 7][ 5]

Select number:11

[11][  ][ 3][10]
[ 9][ 1][ 6][14]
[ 2][ 4][ 8][12]
[13][15][ 7][ 5]

Select number:
```

2 主処理 (main)

```
    int main(void) {
        int sel;
        init();
        do {
            disp();
            do {
                printf("\nSelect number:");
                scanf("%d", &sel);
            } while( sel>ROW*COLUMN );
            moveTile(sel);
        } while( check() );

        return 0;
    }
```

3 盤面の初期化 (init)

乱数を使ってタイルをシャフルするので、時刻を乱数の種に指定することによって、プログラムを起動する度に、異なるタイル配置になる様にしている

```
    void init(){
        int i, j, n;
        srand((unsigned)time(NULL));
        for (i = 0; i < ROW; i++){
            for (j = 0; j < COLUMN; j++){
                n = i * COLUMN + j;
                tiles[n].num = n;
                tiles[n].row = i;
                tiles[n].clm = j;
            }
        }
        shuffle(1000);
    }
```

4 シャッフル (shuffle)

乱数を使って、タイルを不規則に選択し移動させている

```
    void shuffle(int n){
        int sel, min = 0, max = ROW*COLUMN;
        do{
            sel = (rand() % (max - min + 1)) + min;
            moveTile(sel);
            n--;
        } while (0 < n);
    }
```

5 盤面の表示 (disp)

```
    void disp(){
        int i, j, n;
        printf("\n");
        for (i = 0; i < ROW; i++){
            for (j = 0; j < COLUMN; j++){
                n = i * COLUMN + j;
                if (tiles[n].num == 0){
                    printf("[␣␣]");
                } else {
                    printf("[%2d]", tiles[i * COLUMN + j].num);
                }
            }
            printf("\n");
        }
    }
```

6 タイルのスライド (moveTile)

```
    int findTileNum(int num){
        int i, j;
        for (i = 0; i < ROW * COLUMN; i++){
            if (tiles[i].num == num){
                return i;
            }
        }
        return -1;
    }
    void swapTile(int n1, int n2){
        int tmp = tiles[n1].num;
        tiles[n1].num = tiles[n2].num;
        tiles[n2].num = tmp;
    }
    void swapTileR(int c, int r1, int r2){
        int n1 = r1 * COLUMN + c;
        int n2 = r2 * COLUMN + c;
        swapTile(n1, n2);
    }
    void swapTileC(int r, int c1, int c2){
        int n1 = r * COLUMN + c1;
        int n2 = r * COLUMN + c2;
        swapTile(n1, n2);
    }
    void moveTile(int sel){
        int i, j;
        int s = findTileNum(sel);
        int sr = tiles[s].row;
        int sc = tiles[s].clm;
        int z = findTileNum(0);
        int zr = tiles[z].row;
```

```
        int zc = tiles[z].clm;
        if (sr == zr){
            if (sc < zc){
                for (j = zc; j > sc; j--){
                    if (0 <= j - 1){
                        swapTileC(sr, j, j - 1);
                    }
                }
            } else if (zc < sc) {
                for (j = zc; j < sc; j++){
                    if (j + 1 < COLUMN){
                        swapTileC(sr, j, j + 1);
                    }
                }
            }
        }
        if (sc == zc){
            if (sr < zr){
                for (j = zr; j > sr; j--){
                    if (0 <= j - 1){
                        swapTileR(sc, j, j - 1);
                    }
                }
            } else if (zr < sr){
                for (j = zr; j < sr; j++){
                    if (j + 1 < ROW){
                        swapTileR(sc, j, j + 1);
                    }
                }
            }
        }
    }
```

7 完成チェック (check)

```
    enum BOOLEAN check(){
        int i, j, n;
        for (i = 0; i < ROW; i++){
            for (j = 0; j < COLUMN; j++){
                n = i * COLUMN + j;
                if (tiles[n].num != n){
                    return true;
                }
            }
        }
        return false;
    }
```

8 各種宣言など

これは冒頭に記述する

```
    #include <stdio.h>
    #include <stdlib.h>
    #include <time.h>

    // 定数
    #define ROW 4
    #define COLUMN 4
    // 構造体
    struct Tile{
        int num;
        int row;
        int clm;
    };
    enum BOOLEAN{
        false, /* false = 0, true = 1 */
        true
    };
    // function prototypes
    /*
    int findTileNum(int);
    void swapTile(int,int);
    void swapTileR(int,int,int);
    void swapTileC(int,int,int);
    void moveTile(int);
    void shuffle(int);
    enum BOOLEAN check();
    void disp();
    void init();
    */

    static struct Tile tiles[16];   /* 16 = ROW * COLUMN */
```

9　プログラムの全体

ソースコード 1　スライド・パズル

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// 定数
#define ROW 4
#define COLUMN 4
// 構造体
struct Tile{
    int num;
    int row;
    int clm;
};
enum BOOLEAN{
  false, /* false = 0, true = 1 */
  true
};
// function prototypes
/*
int findTileNum(int);
void swapTile(int,int);
void swapTileR(int,int,int);
void swapTileC(int,int,int);
void moveTile(int);
void shuffle(int);
enum BOOLEAN check();
void disp();
void init();
*/

static struct Tile tiles[16];   /* 16 = ROW * COLUMN */

// サブプログラム
int findTileNum(int num){
    int i;
    for (i = 0; i < ROW * COLUMN; i++){
        if (tiles[i].num == num){
            return i;
        }
    }
    return -1;
}
/* ============================================================ */
void swapTile(int n1, int n2){
    int tmp = tiles[n1].num;
    tiles[n1].num = tiles[n2].num;
    tiles[n2].num = tmp;
}
/* ============================================================ */
void swapTileR(int c, int r1, int r2){
    int n1 = r1 * COLUMN + c;
    int n2 = r2 * COLUMN + c;
```

```
53      swapTile(n1, n2);
54  }
55  /* ============================================================ */
56  void swapTileC(int r, int c1, int c2){
57      int n1 = r * COLUMN + c1;
58      int n2 = r * COLUMN + c2;
59      swapTile(n1, n2);
60  }
61  /* ============================================================ */
62  void moveTile(int sel){
63      int j;
64      int s = findTileNum(sel);
65      int sr = tiles[s].row;
66      int sc = tiles[s].clm;
67      int z = findTileNum(0);
68      int zr = tiles[z].row;
69      int zc = tiles[z].clm;
70      if (sr == zr){
71          if (sc < zc){
72              for (j = zc; j > sc; j--){
73                  if (0 <= j - 1){
74                      swapTileC(sr, j, j - 1);
75                  }
76              }
77          } else if (zc < sc) {
78              for (j = zc; j < sc; j++){
79                  if (j + 1 < COLUMN){
80                      swapTileC(sr, j, j + 1);
81                  }
82              }
83          }
84      }
85      if (sc == zc){
86          if (sr < zr){
87              for (j = zr; j > sr; j--){
88                  if (0 <= j - 1){
89                      swapTileR(sc, j, j - 1);
90                  }
91              }
92          } else if (zr < sr){
93              for (j = zr; j < sr; j++){
94                  if (j + 1 < ROW){
95                      swapTileR(sc, j, j + 1);
96                  }
97              }
98          }
99      }
100 }
101 /* ============================================================ */
102 void shuffle(int n){
103     int sel, min = 0, max = ROW*COLUMN;
104     do{
105         sel = (rand() % (max - min + 1)) + min;
106         moveTile(sel);
107         n--;
108     } while (0 < n);
109 }
110 /* ============================================================ */
```

```
111 enum BOOLEAN check(){
112   int i, j, n;
113     for (i = 0; i < ROW; i++){
114         for (j = 0; j < COLUMN; j++){
115             n = i * COLUMN + j;
116             if (tiles[n].num != n){
117                 return true;
118             }
119         }
120     }
121     return false;
122 }
123 /* ============================================================ */
124 void disp(){
125   int i, j, n;
126     printf("\n");
127     for (i = 0; i < ROW; i++){
128         for (j = 0; j < COLUMN; j++){
129             n = i * COLUMN + j;
130             if (tiles[n].num == 0){
131                 printf("[␣␣]");
132             } else {
133                 printf("[%2d]", tiles[i * COLUMN + j].num);
134             }
135         }
136         printf("\n");
137     }
138 }
139 /* ============================================================ */
140 void init(){
141     srand((unsigned)time(NULL));
142     int i, j, n;
143     for (i = 0; i < ROW; i++){
144         for (j = 0; j < COLUMN; j++){
145             n = i * COLUMN + j;
146             tiles[n].num = n;
147             tiles[n].row = i;
148             tiles[n].clm = j;
149         }
150     }
151     shuffle(1000);
152 }
153 // メイン
154 int main(void) {
155     int sel;
156     init();
157     do {
158         disp();
159         do {
160             printf("\nSelect␣number:");
161             scanf("%d", &sel);
162         } while( sel>ROW*COLUMN );
163         moveTile(sel);
164     } while( check() );
165 
166     return 0;
167 }
```