C言語入学式
モグラたたき

配列

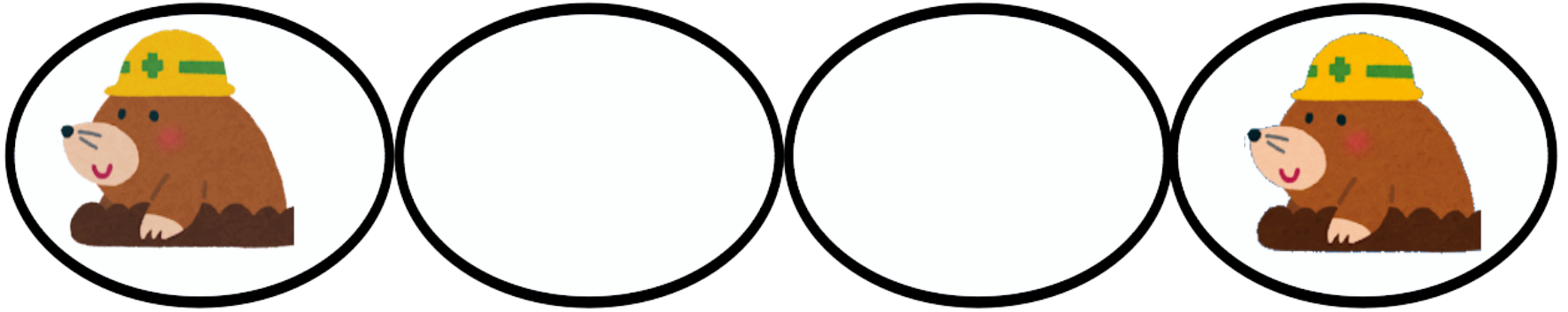
変数とデジタル表現と配列(1)

- **モグラたたき**なので、 たぶんモグラのいる部屋というか穴というか、 とにかくモグラがいる(かもしれない)場所がズラッと並んでいるはずです (そのなかに、モグラの「いる場所」と「いない場所」があります)。 そして、その場所は**同じ部屋/穴が並んでいるはず** -> **こういうときは配列の出番**です！
- 配列は、**同じ入れ物が並んでいるもの**です。 配列の各要素がモグラさんの部屋ということですね
- データ表現: モグラがいるところを1、いないところは0と決めます(ここは人為的な約束事)

表現	モグラ	備考
0	いない	C言語では0から数え始める習わし
1	いる	

変数とデジタル表現と配列(2)

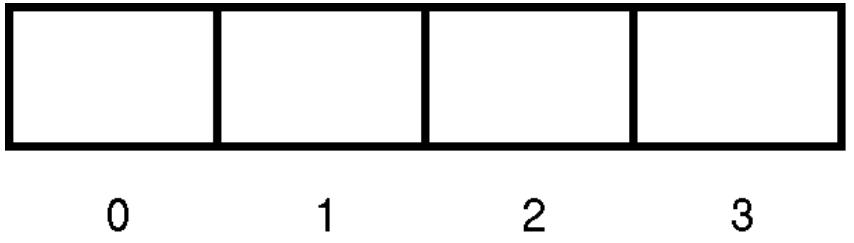
- モグラのいる場所を表現する配列を、変数名moguraと決めます(ここはプログラマの裁量)
  - **部屋の大きさが分かるように、変数宣言の際には型の指定も必要**です (下例は整数配列、次頁を参照)
- 変数を使う際、配列の書式には `配列の名前[インデックス]` を使います



インデックス	0	1	2	3
モグラがいる?	1	0	0	1

```
mogura[0]              mogura[1]              mogura[2]              mogura[3]
```

配列



- 配列とは、たんに**同じ形の箱が連なっている**入れ物(の群れ)です
- 代入や読み出すときは、単に**変数として`array[数字]`を使う**と覚えましょう
- `array[数字]` … 数字で「何番目の箱か」を指定します。 この数字を**インデックス(index)**と呼びます。
  - 注意点: C言語は0から数える習わしであることを思い出して！
- 同じ入れ物をたくさん並べたデータなので、 この変数群つまり配列は一連のシリーズとして使われます。 よって、 **たいていは繰り返し文とセット**で使います。

```
[変数宣言の書式]

型指定   変数名[大きさ];
int      mogura[4];       // 同じ書式だけど、宣言文の場合、インデックスではなく大きさ
```

(脚注) 英語で配列を array と言います

例: ジャンケンで配列で表現してみる(あえてイマイチな例)

- 配列として変数arrayを宣言した場合、 変数の代入や読み出しの対象は、 一つ一つの箱になります。
- 例: ジャンケンの `jibun` や `aite` を `array[0]` や `array[1]` といった配列の要素に書き換えましょう
  下のコードを `array[0] -> jibun` と(心の目で?)読み替えてみてください

```
array[0] = 1;               //    変数に値1を代入; 変数はjibunに相当するarray[0]
scanf("%d",   &array[0]);  // キーボードから入力; 「&変数」と書くから、この形、ok?
printf("%d\n", array[0]);  // ディスプレイに出力; 変数の値を読み出しています
```

- ジャンケンの手の配列はダメな例です ;_;
  - もっと分かりやすい名前をつけた方がいいので、ジャンケンの配列 `janken_te` とか作ることにすると?
  - 自分の手(整数変数 `jibun` 相当)を `jibun_te[0]` に代入、 コンピュータの手(整数変数 `aite` 相当)を `jibun_te[1]` に代入と考えましょう
  - もともと同じデータを並べていないので、 ジャンケンの場合は逆に読みにくいと思いませんか？
  - 前ページで述べたように、同じデータをたくさん並べる場合に配列を使うべきです

(脚注) C言語では、ジャンケンをどう表現するべきなのか？というと、 配列ではなく構造体(struct)を使う場面になります

モグラたたき(1次元,乱数なし)

「くりかえし」について考える(1)

```
// もぐらのいる場所を設定
// 位置0に「いる」 つまり mogura[0]=1;
// 位置1に「いない」
// 位置2に「いない」
// 位置3に「いる」
// もぐらのいるいないを出力
// 位置0の値を出力、つまり printf("%d", mogura[0]);
// 位置1の値を出力
// 位置2の値を出力
// 位置3の値を出力
```

- 課題510(次頁)をベタに書くと左のようになります (注: コメントだけが書いてあります)

- 課題510は、 くりかえし(または「ループ」)を使わず、 まずはベタに書いてみようという課題なのですが、 はっきりいって面倒ですよね?
- まだ4つだから我慢できそうですが、 これが二次元マス目9x9で合計81箇所となったらイヤというか無理ですよね? だからループ(繰り返し)を使えるようになりましょう！
- 課題520(次次頁)では「コードの前半部分(配列の設定)は一行の省略形でも書ける」をやってもらいますが、 前半は短く書けても、後半は楽になりません (そして省略形で書けるからといって2次元マス目81ヶ所を書くのもイヤでしょうな:-)

(脚注) 出力時は、読みやすいように printf("mogura[0] = %d\n", mogura[0]); としてくれると嬉しいです

課題510

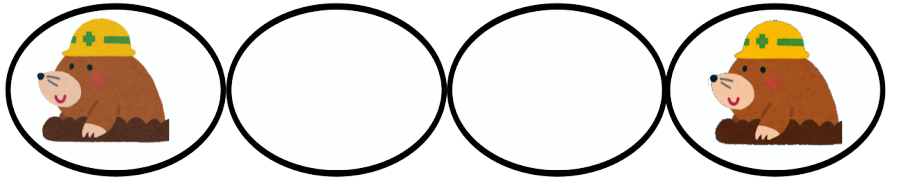
```
// (1) 最初の行に必ず書く呪文

int main () {
  // (2) 整数変数の配列 mogura を宣言

  // (3) 配列に値を代入: いる,いない,いる,いない

  // (4) 出力: 変数の値をすべて出力してください

  return(0);
}
```



- 穴が4つあるとします。 穴の下にモグラが「いる」「いない」「いない」「いる」状態を表現してください
- 繰り返しを使わず、ベタに代入を4回、出力を4回書いてみましょうか

ヒント: 課題510〜540では、まだ、くり返しを使わなずベタに書く想定です。

もちろん自信がある人は「くりかえし文」を使ってもいいですけど... まぁ、そこは各自におまかせします

課題520

```
// (1) 最初の行に必ず書く呪文

int main () {
  // (2) 整数変数の配列 mogura の宣言と代入
  //     モグラは「いる」いない」いない」いる」

  // (3) 出力: 変数の値をすべて出力してください

  return(0);
}
```

- もう少し楽な書き方を覚えましょう
  - 4回も代入するのは面倒ですよね?いまは4回だから、ベタ書きでもいいですが、これが100回とかになると、もう…
- この課題520では
  - 1回(つまり1行)で、配列の宣言と代入を行ってください
  - 後半(出力)は前課題510と同様ベタに出力

ヒント: 配列の宣言と代入(を一気に行う)

課題530

```
// (1) 最初の行に必ず書く呪文

int main () {
  // (2) 整数変数の配列 mogura の宣言と代入
  //     モグラは「いる」いない」いない」いる」

  // (3) 代入: jibun に 1 を代入

  // (4) 条件文と出力: jibunの場所にモグラがいる?
  //     いれば atari、いなければ hazure と出力

  return(0);
}
```

- モグラたたきの準備です。 次の問題540では、ユーザが「モグラの(予想)位置」をキーボードから入れ、atariかhazureを表示させます
- その前準備として**(ゲームっぽくはないですが)ユーザが叩く位置をコード内に決め打ちで書いて、 モグラたたきを動かしてみましょう**
  - 整数変数jibun(ユーザが予想した「モグラの位置」を代入する変数)を宣言し、値に1を代入してください (ユーザが場所1を叩いたという意味です)
  - そこ(jibun)にモグラがいたら atari、いなければ hazure と表示してください
  - hazureと表示されないといけません

(脚注) 面倒な人(キーボードからの入力わかるよ！という人)は、本課題を飛ばして次の課題540へ進んでもokです。 こういうのを鬱陶しいと思うかもしれませんが、 コードは、ステップバイステップで、動作を確認しながら書いていくべきなのです (お作法の伝授)

課題540

```
int main () {
  // (2) 整数変数の配列 mogura の宣言と代入
  //     モグラは「いる」いない」いない」いる」

  // (3) 入力: キーボードからjibunに値を読み込む

  // (4) 条件文と出力
  //     jibun の場所にモグラがいるか?
  //     いれば atari、いなければ hazure と出力

}
```

- 課題530を改造し、 ユーザが予想する「モグラの位置」をキーボードから読み込んで変数jibunに代入してください
- もう少し対話的なモグラたたきです！
  - モグラの位置はソースコードのとおりなので、すでに答えを知っていますよね?;-)
  - 次節の課題610ではモグラの位置をコンピュータに決めさせます (やっとゲームっぽい！)

ヒント: (定番のキーボードからの)入力

課題550(旧590)

```
int main () {
  // (A) 一回だけ実行する部分

  for ... 省略 ... { // 繰り返し文

     // (B) くりかえし実行する部分

  }
}
```

- モグラたたき1次元版が、ひととおり完成です (この後もう少しゲームらしくしていきます)
- 課題540を改造しキーボードから3回挑戦できるようにしてください (「モグラの位置」を3回入力できるようにする)
- 課題540の(1)〜(4)のうち、1回だけ実行する部分と繰り返す部分がどこか考えましょう

```
[実行例]

$ ./.aout
0
atari
1
hazure
2
hazure
```

ヒント: 繰り返し文

演習: くりかえし文(入力)

```
// 入力を2回くりかえす
scanf("%d", &mogura[0]);
scanf("%d", &mogura[1]);

// [抽象化]
// くりかえし文
for (i = 0; i < ?; i++) {
    scanf("%d", &mogura[ ? ]);
}
```

- ここでは入力を2回くりかえしています
- これをfor文で書き直す場合、左下のくりかえし文の`?`部分(2ヶ所)には、 それぞれ何が入りますか?
- ヒント
  - ベタ書きされているコードの規則性を読み解いて、 for文に変換する訓練です **(規則性の解読 or 抽象化)**
  - **規則性を見出すのはfor文の中に書くコード**の話ね

演習: くりかえし読みこむ(ベタに)

```
int main () {
  // (2) 整数変数の配列 mogura の宣言と代入
  //
  //    つまり、以前ベタ書きした次の処理を
  //    scanfで書き直してください
  //    位置0にmogura[0]に値を読みこむ
  //    位置1にmogura[1]に値を読みこむ
  //    位置2にmogura[2]に値を読みこむ
  //    位置3にmogura[3]に値を読みこむ

  // (3) 入力:        (540と同じ,省略)
  // (4) 条件文と出力 (540と同じ,省略)
}
```

- 課題540の「(2) 整数変数の配列 mogura の宣言と代入..」の部分を改造します。 ベタにキーボードから繰り返し入力を読みこんでください。

```
[実行例]
$ ./a.out
1
0
0
1
1
hazure
```

課題560: くりかえし読みこむ(forを使う)

```
int main () {
  // (2) 整数変数の配列 mogura の宣言と代入
  //
  //    つまり、以前ベタ書きした次の処理を
  //    くりかえし文+scanfで書き直してください
  //    位置0に「いる」 つまり mogura[0]=1;
  //    位置1に「いない」
  //    位置2に「いない」
  //    位置3に「いる」

  // (3) 入力:        (540と同じ,省略)
  // (4) 条件文と出力 (540と同じ,省略)
}
```

- **前頁のベタ書きを抽象化できるか？を問う課題**
- (2)の部分で「くりかえし文」を使い、 キーボードから繰り返し入力を読みこんでください。

```
[実行例]
$ ./a.out
1
0
0
1
1
hazure
```

演習: くりかえし文(出力)

```
// 出力を2回くりかえす
printf("%d", mogura[0]);
printf("%d", mogura[1]);

// [抽象化]
// くりかえし文
for (i = 0; i < ?; i++) {
    printf("%d", mogura[ ? ]);
}
```

- ここでは出力を2回くりかえしています
- これをfor文で書き直す場合、左下のくりかえし文の`?`部分には何が入りますか?

デバッグの方法: よくわからないときは、こうやってすべてを出力してみてください

演習: くりかえし文(出力),インデックスあり(1)

```
printf("???", ???, ???);
printf("???", ???, ???);

[実行例]（入力部分は省略）

$ ./a.out
0 1
1 0
```

- 代入した結果を確認したいです
- ???のところには何があてはまりますか?
- まずは**ベタ書き**で、 ユーザがキーボードから入力した配列moguraの値を確認できるようにしてください
- ただし配列を見やすくするため、 **出力は`インデックス 値`(例: 0 1)**形式とします

解答: くりかえし文(出力),インデックスあり(1)+for版の解説

```
[実行例]（入力部分は省略）

$ ./a.out
0 1
1 0

printf("%d %d\n", 0, 1);
printf("%d %d\n", 1, 0);
                  A  A
                  |  |ここは配列の値
                  | ここが規則的にずれる
```

for 文を左上から右下へ順番に読み解くと、こうなります (これを省略して3行に書いている)

```
for (i = 0; i < 2; i++) {
    printf("%d %d\n", i, mogura[i]);
}

i = 0（最初に1回だけ実行）
i < 2（0 < 2 なので条件を満たす）
printf("%d %d\n", 0, mogura[0]);を実行
i++を実行して，iの値は1
i < 2（1 < 2 なので条件を満たす）
printf("%d %d\n", 1, mogura[1]);を実行
i++を実行して，iの値は2
i < 2（2 < 2 なので条件を満たさない）
くりかえし終了
```

課題570: 代入した結果を確認したい

```
[実行例]

$ ./a.out
1
0
1
0
0 1
1 0
2 1
3 0
0
atari
}
```

- **前問のベタ書きを抽象化できるか？を問う課題**
- ユーザがキーボードから入力した配列moguraの値を確認できるようにします。 **課題560に配列の中身を出力するコードをfor文で書いて付け加えてください**
- 配列を見やすくするため出力は `インデックス 値` (例: 0 1)としてください。 実行例(左)の5行め〜8行目が、この出力に当たります
- 実行例(左)の最後の2行が、もぐらたたきです。

```
0
atari
```

0を入れたらatariと出ている様子です(じゃんけん1回のみ)

演習: くりかえし文(出力),インデックスあり(2)

```
printf("???", ???, ???);
printf("???", ???, ???);

[実行例]（入力部分は省略）

$ ./a.out
mogura[0] = 1
mogura[1] = 0
```

- ユーザがキーボードから入力した配列moguraの値を確認できるようにしてください。 **???のところには何があてはまりますか?**
- さらに、見やすくするため、 **出力は`mogura[インデックス] = 値`(例: mogura[0] = 1)**形式とします
- よくわからないなら、まずは**ベタ書き**で書いてみて、 規則性を考え、for文にしていきましょう

課題580: くりかえし文(総合)

- いままでの全てを合体(+アルファ)してください。概要と実行例だけがヒント

```
int main () {
// (2) 配列 mogura の宣言
// (3) もぐらの位置を入力
// (3) 配列 mogura を出力(確認のため)
// (4) もぐらたたき(条件文と出力)を3回
}
```

```
[実行例]
$ ./a.out
1
0
1
0
mogura[0] = 1
mogura[1] = 0
mogura[2] = 1
mogura[3] = 0
1
hazure
2
atari
0
atari
```

課題590: くりかえし文(総合), 580を視覚的にわかりやすく

- 課題580を、もうすこし視覚的に分かりやすくしてください。 もぐらのいる位置は`0`(アルファベット大文字のオー)、いない位置は`.`(ドット)
  - たとえば`1 0 1 0`の場合`0 . 0 .`と出力

```
int main () {
// (2) 配列 mogura の宣言
// (3) もぐらの位置を入力
// (3) 配列 mogura を出力(確認のため)
//     条件文も必要
// (4) もぐらたたき(条件文と出力)を3回
}
```

```
[実行例]
$ ./a.out
1
0
1
0

0 . 0 .

1
hazure
2
atari
0
atari
```

(脚注) この文字で図を書くアスキーアート表現は2次元モグラで使います

モグラたたき(1次元,乱数あり)

乱数を使い、よりゲームらしくしましょう。 また、混乱した時は配列を出力してみましょう(printf debug)

課題610

```
int main () {
  // (2) 整数変数の配列 mogura の宣言

  // (3) モグラの位置を乱数で決める

  // (B) 繰り返し
     // (4) 入力:キーボードからjibunに値を代入
     // (5) 条件文と出力
     //     jibun の場所にモグラがいるか?
}
```

- ユーザが入力するのではなく、 **コンピュータにモグラの位置を決めさせて**ください
- ヒント:
  - 入力するループを、どう変えればよいでしょうか?
  - どういう式を書けば、 乱数で、 モグラの「いる」「いない」を決めることができるでしょうか? (ジャンケンの理屈を少しひねれば出来そうですよね?)

ヒント: 乱数

課題620

```
int main () {
  // (2) 整数の変数tensuuと配列moguraを宣言
  // (3) モグラの位置を乱数で決める
  // (B) 繰り返し
     // (4) 入力:キーボードからjibunに値を代入
     // (5) 条件文と出力:
     //     jibun の場所にモグラがいるか?
     //     atari の時は tensuu に加算
  // (6) tensuu を表示
}
```

- 課題610を元に、最後に「点数」を表示してください。 点数とは、もぐらを叩けた回数(atari の回数の合計)のことです
- 整数変数 tensuu を使ってください

課題630

```
//（1）最初の行に必ず書く呪文
//      定数

... 以下、課題620と同様 ...
```

- 4とか3とか数字がいろいろ出てきてソースコードが読みにくいですね
- 定数を使い、課題620を綺麗にしてください。モグラ配列の大きさは YOKO, くりかえす回数の 3 は SAIDAI という定数にしましょう
- また、モグラ配列を初期化する繰り返し文では i ではなく yoko という変数名を使ってください

ヒント: 定数(defineの使い方)

コラム: なお普通の変数や関数には小文字、定数には大文字を使う習わしがあります

課題690

```
// (1) ヘッダファイル
//      定数

int main () {
  // (A)一度だけ処理する部分
  //    (2) 宣言
  //    (3) モグラの位置決め(乱数)

  // (B) 繰り返し
     // (4) 入力
     // (5) 条件文と出力

  // (6) tensuu を表示
```

```
}
```

- [1次元モグラの総合課題] 改造テーマが2つ
  - (a) 配列サイズを 9 にしてください
  - (b) モグラの割合を約3割にしてください
- 課題630がうまく出来ていれば、(a)は簡単ですが、問題は(b)ですね。 これは、ジャンケンよりも少し複雑なコードが必要でしょう
- ぱっと思いつかない人は、 まず、 次頁以降のステップバイステップの課題(690a 〜)で訓練してみましょう

ヒント: アルゴリズム: (0〜9の目が出る)10面サイコロをふったとして、0,1,2の目がでる割合は合計3割ですよね?

(脚注) ぱっと考えつく案として (1)浮動小数点が必要なコード (2)整数だけで書けるコードの両方があります

課題690a 穴の数を9へ

```
//（1）ヘッダファイル
//      定数

int main () {
  //（A)一度だけ処理する部分
  //    （2）宣言
  //    （3）モグラの位置決め(乱数)

  //（B）繰り返し
     //（4）入力
     //（5）条件文と出力
```

```
  //（6）tensuu を表示
}
```

- 「穴の数を9へ」増やします
- そもそも課題630ができていることが大前提です。 まず、ここをTA/SAさんに確認してもらってください
- 課題630では穴(配列)の数を定数YOKO、繰り返しの回数を定数SAIDAIとしました。 定数を変えればループの回数も変わりますよね?OK?

【復習】 乱数

```
x = rand();          // rand関数を実行するたび x には違う値が入ります
                     // 値の範囲は 0〜約21.5億 (2,147,483,647)です
                     // 最大値は定数 RAND_MAX として定義されています

// 使い方の例

// ジャンケンは 3 で割ったさいの剰余(あまり)で十分
int x;
x = rand() % 3;                                  // x の値は 0 1 2

// 割合
double y;
y = (double) rand() / (double) RAND_MAX;         // x は 0.0 〜 1.0

// パーセンテージ
double y;
y = 100 * (double) rand() / (double) RAND_MAX; // x は 0.0 〜 100.0
```

課題690b もぐらの配置ぐあいを確認する(確率1/2)

```
// (1) ヘッダファイル
//      定数

int main () {
  // (A)一度だけ処理する部分
  //     (2) 宣言
  //     (3) モグラの位置決め(乱数)

  // (B) 繰り返し
     // (4) 入力
     // (5) 条件文と出力

  // (6) tensuu を表示
}
```

- この課題は単なる確認です 「乱数でモグラの位置を決めた直後のmogura配列を表示」してみてください
  - 「穴の数を9へ」増やしたので、そこへモグラを配置します
  - モグラがいる場所は乱数で確率1/2で決めます。 2面体サイコロを振っているとおもってもよいです
- 690aが正しく動いていれば、9個のうち4つか5つにモグラがいるはずです。
  - ただし乱数関数の性能が良くないのと、 たかだか9個なので、 モグラが偏っていることがあります。 その場合は、もう一度実行しなおしてみてください (何度やっても偏っているなら何か間違っています)

課題690c もぐらの配置割合を変える

```
// (1) ヘッダファイル
//      定数

int main () {
  // (A)一度だけ処理する部分
  //    (2) 宣言
  //    (3) モグラの位置決め(乱数)

  // (B) 繰り返し
     // (4) 入力
     // (5) 条件文と出力

  // (6) tensuu を表示
}
```

- 乱数でモグラの位置を決める部分を調整します
- 前課題が正しく動いていれば、9個のうち4つか5つにモグラがいるはずです
  - だから今は、だいたい割合1/2でモグラが存在するわけです
- この割合を9個のうち3個くらいにしてください
  - 割合1/3でモグラが存在するようにするということです
- 乱数をどう使うか？を考えてみてください

```
x = rand(); // rand関数を実行するたび x は違う値
          // 値の範囲は 0〜約21.5億 (2,147,483,647)
```

(脚注) ジャンケンは 0 1 2 の3通りの値が欲しかったので、剰余でしたけどね。 他にもやり方は考えられるよね?

課題690d もぐらの配置割合を変える

```
// (1) ヘッダファイル
//      定数

int main () {
  // (A)一度だけ処理する部分
  //     (2) 宣言
  //     (3) モグラの位置決め(乱数)

  // (B) 繰り返し
     // (4) 入力
     // (5) 条件文と出力

  // (6) tensuu を表示
```

```
}
```

- 乱数でモグラの位置を決める部分を調整します
- では、モグラの割合を約3割にするには?
- 9個のうちだいたい3個くらいということですね。 でも割合1/3(=0.3333..)ではなく0.3です (じゃんけんを素直に応用すると、それは違います)
- 乱数をどう使うか？を考えてみてください

```
x = rand(); // rand関数を実行するたび x は違う値
          // 値の範囲は 0〜約21.5億 (2,147,483,647)
```

課題695 もぐらの叩き具合を表示する

```
// (1) ヘッダファイル
//      定数

int main () {
  // (A)一度だけ処理する部分
  //    (2) 宣言
  //    (3) モグラの位置決め(乱数)

  // (B) 繰り返し
     // (4) 入力
     // (5) 条件文と出力
     // (5A)記録（進行状況表示のため）

  // (C) 進行状況を表示
```

```
}
```

- 課題700番台ではアスキーアートでゲームの進行状況を表示するので、1次元の最後に、その練習をしましょう
- まずは課題610への機能追加で練習します。うまくできたら、課題620や690に状況表示機能を追加してみてください
- Thinking Time
  - 「(5A)記録 (進行状況表示のため)」と「(C) 進行状況を表示」が追加されています
  - 進行状況の表示のためには、どういった情報を保存していればよいか？を少し考えてみてください

課題695a もぐらの叩き具合を表示する: 進行状況?

```
.O.XX...
```

- 表示例(上図)の説明
  - 2番目を叩いたらモグラがいた！
  - 4と5番目も叩いたがモグラはいなかった
  - 残りは、まだ叩いていない

- 課題720では次のような表示分けをするので、これと同じにしてください
  - O ... atari
  - X ... hazure
  - . は、まだ叩いていない場所
  - 注: 使う文字は大文字アルファベットのオーとエックスそして特殊文字のドットです (OscarのOと、X-rayのX)

課題695b もぐらの叩き具合を表示する: 記録?

```
.O.XX...
```

- 上の表示例では3種類の文字が使われています
  - O ... atari
  - X ... hazure
  - . は、まだ叩いていない場所

- 「叩いていない」はデフォルトなので記録する必要は無し (つねに . を出力すれば良い)
- 「叩いた場所」と「叩いた結果」を何かに記録しておかないと表示できません。 さて、どうするとよいでしょうか？ 考えてみてください
  - 必要な変数は1つ?2つ?
  - その変数の種類は?

(脚注) 答え：整数配列(変数名 rireki)が1つあれば記録とれますよね？使う値には一工夫が必要でしょう

課題695c もぐらの叩き具合を表示する: 出力

```
// (C)

// (C.1) くりかえし文(rireki 配列)

  // (C.2) 条件文(各要素ごとに条件判定)
  // もし「モグラがいた」ら O を出力
  // もし「モグラがいなかった」ので X を出力
```

```
    // どららでもない場合 . を出力
```

- 「(C) 進行状況を表示」では(記録をとっている)変数を元に「モグラ叩き」状況を出力します
  - (C.1) くり返し文で rireki 配列の要素を順番に調べて行きます
  - (C.2) 各配列要素を条件文で調べ、値に応じて ＯＸ . を出力していきます

モグラたたき(2次元,乱数あり)

2次元にすると、もっとモグラたたきっぽくなります

2次元配列

- 1次元配列は箱が横に並んでいます

```
a[0] a[1] a[2]
```

- 2次元配列はマス目つまりスプレッドシート(例:Excel)状で、 これに縦横の場所を指定する数字がつくというだけのことです

```
a[0][0] a[0][1] a[0][2]
a[1][0] a[1][1] a[1][2]
a[2][0] a[2][1] a[2][2]
```

- Excelのマス目はアルファベットと数字で指定してますけど、 たてよこ両方とも数字でも大丈夫だ(慣れる)よね? (ただしC言語なので数字は0から始まることに注意)

```
A1 B1 C1
A2 B2 C2
A3 B3 C3
```

演習: 2次元に慣れる(1)

[実行例]

```
1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 1
```

- 対角線にそって 1 を、対角線以外は 0 を出力してください (左上から右下に沿ってのみ 1、いわゆる正規行列の一例)
- 配列を初期化して値を埋めこむのではなく、 繰り返しと条件文だけで書いてください
- どこかで改行も必要ですね
- 頭を使う課題ですけど、 実際のコードの長さは数行です

演習: 2次元に慣れる(2)

[実行例]

```
1 0 1 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0
1 0 1 0 1 0 0 0 0
0 1 0 1 0 1 0 0 0
0 0 1 0 1 0 1 0 0
0 0 0 1 0 1 0 1 0
0 0 0 0 1 0 1 0 1
0 0 0 0 0 1 0 1 0
0 0 0 0 0 0 1 0 1
```

- 対角線にそって縞模様?状に 1 それ以外は 0 を表示してください。 うまく説明できませんが実行例のように3列の縞々
- 配列を初期化して値を埋めこむのではなく、 繰り返しと条件文だけで書いてください
  - どこかで改行も必要です
  - 0から始まるから一つずれる可能性に注意
- 一発では書けないので、具体例を書いてみて考えてください
  - i が 0 で j が 0 か 2 のときに 1 を出力
  - i が 1 で j が 1 か 3 のときに 1 を出力

演習: 2次元に慣れる(3)

[実行例]

```
1 0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 1 0
0 0 1 0 0 0 1 0 0
0 0 0 1 0 1 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 1 0 1 0 0 0
0 0 1 0 0 0 1 0 0
0 1 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 1
```

- 対角線にそって X 字型に 1 それ以外は 0 を表示してください
- 配列を初期化して値を埋めこむのではなく、 繰り返しと条件文だけで書いてください
  - どこかで改行も必要です
  - 0から始まるから一つずれる可能性に注意
- 一発では書けないので、具体例を書いてみて考えてください
  - i が 0 で j が 0 か 8 のときに 1 を出力
  - i が 1 で j が 1 か 7 のときに 1 を出力

演習: 2次元に慣れる(4)

[実行例]

```
0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 1
0 0 0 1 0 1 1 0 1
0 0 1 0 0 1 0 0 0
1 1 1 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 1
```

- 乱数で1か0を決めて、(確認のため)もぐらのいる位置をを表示してください
  - もちろん、この部分も二重ループです
  - 注: 1 or 0 を決める**ロジックしだいで、かなり出力は変わります。** 左の実行例は約3割に調整済みの例です
  - **もちろん、ゲームの場合、これを表示してはいけません (ユーザに答えを教えてます:-)**
- 表示は、配列を初期化して値を埋めこむのではなく、 繰り返しと条件文だけで書いてください
  - どこかで改行も必要です
  - 0から始まるから一つずれる可能性に注意

(脚注) 課題690をクリアしている前提なので、もぐらを約3割ってところは大丈夫ですよね?

課題710

[実行例]

```
$ ./mogura
1
2
atari
2
3
hazure
6
8
```

```
atari
tensuu = 2
```

- 1次元版のモグラ完成形(課題690)を参考に、2次元版のモグラを作ってください。
- 仕様
  - モグラの初期値も乱数で適当に配置
  - もちろんユーザは毎回2つ数字を入れます
  - 3回挑戦できます
- あとは、ノーヒントでトライ

課題720

[実行例（9x9）]

```
6
7
atari!
. X . . . . . . .
. X . . . . . X .
. . X X X . . . .
. . . . . . . . .
. . . X . X . X .
. . . . . . . . .
. . . . . . . O .
. . . . . . . . .
. . . . . . . . .
```

- 分かりやすいように、モグラたたきの現状を二次元のマス目で表示するようにしてください
  - O ... atari
  - X ... hazure
  - . は、まだ叩いていない場所
  - 注: 使う文字は大文字アルファベットのオーとエックスそして特殊文字のドットです (OscarのOと、X-rayのX)

課題730

[実行例（9x9)]

```
7
8
atari!
. X . . . . . . .
. X . . . . . X .
. . X X X . . . .
. . . . . . . . .
. . . X . X . X .
. . . . . . . . .
. . . . . . . O .
. . . . . . . . .
. . . . . . . . .
```

- 課題720を改造し、 入力の指定を 0 スタートではなく 1〜9 で指定できるようにしてください
- たとえば今までモグラの位置を 01などと指定していたところが12 になります
  - どうすると最小の変更で実現できるか?
  - C言語では0から始まるので、 配列のインデックスと、もぐらの位置の指定が一つずれることに注意しないとうまく動きません
  - 配列の境界に気をつけようという練習:-)

課題740

```
[実行例 (9x9)]

7
8
atari!
  1 2 3 4 5 6 7 8 9
1 . X . . . . . . . 1
2 . X . . . . . X . 2
3 . . X X X . . . . 3
4 . . . . . . . . . 4
5 . . . X . X . X . 5
6 . . . . . . . . . 6
7 . . . . . . . O . 7
8 . . . . . . . . . 8
9 . . . . . . . . . 9
  1 2 3 4 5 6 7 8 9
```

- 課題730を改造して、モグラたたきの様子が見やすいように 1 … 9 の枠をつけてください
  - これも配列の境界まわりのあつかいで頭をひねる問題:-)

モグラたたき
発展課題

発展課題INDEX

- 810: モグラたたきを何回できるか?指定できるようにしてください
- 820: モグラたたきの回数に上限なし。数字の9を入れたら終了
- 830: モグラには下っ端とボスがいます。下っ端とボスでは異なる点数にしてください
  - たとえばボスは10点、下っ端は1点
  - モグラの合計は穴の約3割
  - その割合はボス1:下っ端2
- 840: 点数を3種類に増やしてください。
  - たとえば、ボスが10点、 中間(管理職)?は5点、 下っ端が1点
  - モグラの合計は穴の約3割(前問題(830)と同じ)
  - モグラの割合は、ボス1:中間1:下っ端1
- 850: 840の割合を1:2:3にしてください。 ただし、モグラの合計は穴の約3割とすることは同じとします

(脚注) 課題810と820はジャンケンの発展課題と同様なので簡単でしょう。 830はデータ表現の工夫次第で楽になると思います。

850は何面体サイコロを使って、どう判定するか？を工夫すれば簡単ではないかと思います

課題810: モグラたたきの回数を指定できる

```
// (1) ヘッダファイル
//      定数

int main () {
  // (A)一度だけ処理する部分
  //    (2) 宣言
  //    (3) モグラの位置決め(乱数)

  // (B) 繰り返し
     // (4) 入力
     // (5) 条件文と出力

  // (6) tensuu を表示
}
```

- 以下、課題690(1次元)もしくは710(2次元)を元に改造する想定です (もちろん、より完成形の課題740を元にしてもOKです)
- 以下に変更点だけヒントを書いておきます
- (A) セクション(一度しか処理しない部分)
  - [新規] 新しい整数変数jougenを宣言します。繰り返しの回数を決める変数です
  - [新規] キーボードからjougenに値を代入します
- (B) セクション(くり返しの部分)
  - [変更] 最大jougen回もぐらたたきができます

課題820: モグラたたきの回数制限なし,9で終了

```
// (1) ヘッダファイル
//      定数

int main () {
  // (A)一度だけ処理する部分
  //    (2) 宣言
  //    (3) モグラの位置決め(乱数)

  // (B) 繰り返し
     // (4) 入力
     // (5) 条件文と出力

  // (6) tensuu を表示
}
```

- 以下、課題690(1次元)もしくは710(2次元)を元に改造する想定です (もちろん、より完成形の課題740を元にしてもOKです)
- 以下に変更点だけヒントを書いておきます
- (A) セクション(一度しか処理しない部分)
  - ここは710(or 740)と変わらないはずです
- (B) セクション(くり返しの部分)
  - くりかえす部分だけの変更です
  - まずは繰り返しを無限ループにして、
    - キーボードから叩く場所を読みこんだ際、 数字が9の場合ループを終了する (注: 2次元モグラの場合tate,yokoいずれかが9)

課題830: 下っ端モグラとボスモグラ

```
// (1) ヘッダファイル
//      定数

int main () {
  // (A)一度だけ処理する部分
  //    (2) 宣言
  //    (3) モグラの位置決め(乱数)

  // (B) 繰り返し
     // (4) 入力
     // (5) 条件文と出力

  // (6) tensuu を表示
}
```

- モグラには下っ端とボスがいて、下っ端とボスでは異なる点数たとえば下っ端は1点,ボスは10点
- 以下に変更点だけヒントを書いておきます。
  - そもそも「モグラがいるか否か?」と「点数の違い」の二種類のデータをあつかう必要があるので、 データ表現の工夫しだいでコードの変更量がだいぶ変わります
  - 案1: 位置と得点それぞれに配列を使う。つまり2つの2次元配列を使います
  - 案2: 配列は1つのままで、mogura配列の値に2種類の意味をもたせます。 (a)配列の値が1以上なら種類はともかくモグラがいる (b)点数は配列の値そのものを利用