関数化しよう
2023/07/07 version

関数化とは、、、そう！多重下請け構造を作ることだ！(こらこら)

注意

- プロトタイプ宣言について
  - 一旦わすれましょう
  - 書かなくても動きます(コンパイラに警告されますけど動きます)
  - 試行錯誤しながら、遠く離れた2ヶ所を編集するのは難しいです
  - 付録で少しプロトタイプ宣言のコツについて語っています
- 不安がある人は教科書を読み直しましょう

関数のふんわりした話(たとえ話)

- だいたいのプログラムは「入力」「計算する(プログラムの実体)」「出力」
  - 今までは**main(プログラム全体)だけ**だったので、 **入力は「ユーザがキーボードから文字や数字を入れる」**、 **出力は「モニタへ文字や数字が出てくる」**ことでした
- (長い)プログラムは「mainから**下請け(関数)**に仕事を依頼する」作業の連続です
  - 関数への依頼(**発注**)は、mainから関数へデータを**渡す** (関数へ**データを入力**と言えます)
  - 関数からの**納品**はmainへデータ(関数での計算結果)を**返す**こと (mainへデータを**出力**と言えます)
  - 「渡す/返す」「入力/出力」「発注/納品」どれでも、なじめる表現でokです。 ただ問題文は「渡す/返す」表現なので慣れてくださいね

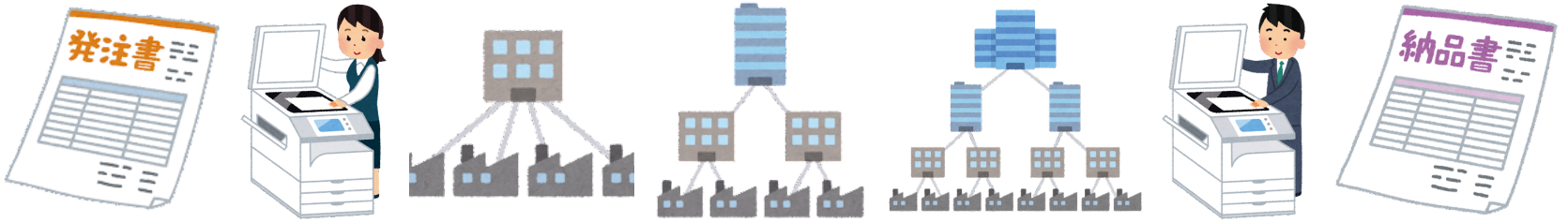
```
// プログラム(全体)
(キーボードから)入力 -> プログラム -> (モニタへの)出力

// 関数(プログラムの一部の動作); 「渡す/返す」と「入力/出力」なじめる表現でok(問題文は前者の表現なので慣れてね)
mainから渡す -> 関数 -> mainへ返す
mainから入力 -> 関数 -> mainへの出力
mainから発注 -> 関数 -> mainへの納品
```

(脚注) そもそも「プログラムの実行」とは、 人間からコンピュータ(下請け)へ仕事を発注することですよね?

関数のふんわりした話(たとえ話)

```
// プログラム(全体)
(キーボードから)入力 -> プログラム -> (モニタへの)出力

// 関数(プログラムの一部の動作); 「渡す/返す」と「入力/出力」なじめる表現でok(問題文は前者の表現なので慣れてね)
mainから渡す -> 関数 -> mainへ返す
mainから入力 -> 関数 -> mainへの出力
mainから発注 -> 関数 -> mainへ納品
mainから発注 -> 下請 -> mainへ納品
```

- (長い)プログラムは「mainから**下請け(関数)**に仕事を依頼する」作業の連続です



(脚注) 発注時じつは(変数の)コピーを下請け(関数)に渡し、 下請け(関数)からの納品(返り値)もコピーです

プログラムたとえ話「学食」

1. 食券を買う
2. 食券を渡す
3. 該当する食事をもらう
4. 食べる
5. 食器を返す

関数たとえ話「食券を買う」

- だいたいのプログラムは「入力」「計算する(プログラムの実体)」「出力」
- 学食の券売機も同じでしょ?
  1. お金を入れる(入力)
  2. メニューボタンが押される
  3. (裏側で)なにか計算(仕事)している
     - たぶん裏で -> 印刷 -> (チケットサイズに切る?) -> 食券を下へ落とす
  4. 食券とおつりが出てくる(出力)

(脚注) ちなみに右の写真は昔の学食(-2019)の券売機です。よ〜くみるとメニューが違う

擬似コード「食券を買う」

食券を買う動作は次のようになるでしょう

1. おかねをいれる
2. (裏側で印刷したり準備)
3. 食券が出てくる

```
// 擬似コードで書くと、こんな感じでしょうか
// おかね(IN) -> 券売機 -> 食券(OUT)

食券 食券を買う（おかね）{
    裏側で行われる食券販売機の実体を書くところ;
    return 食券;
}
```

```
// 関数の雛形

食券の型 食券を買う(型 おかね) {
        裏側で行われる食券販売機の実体を書くところ;
        return 食券;
}

int 食券を買う(int okane) {
        int shokken;
        // 裏側で行われる食券販売機の実体を書くところ;
        return shokken;
}
```

(脚注) ここではメニューは一つしかないと簡単化。 ふつうは食券購入時に選択の動作もありますよね

食券の型?その実体はメニューの番号=整数かな?

- お金は整数型 int でいいですよね?
- 食券の型は?と言うと、 **実世界を、どうデジタル表現するか?はプログラム設計者しだいだから(仕様書しだい)**が答え
- 食券の情報は料理の番号(int)で良いよね?

```
// いろいろ省略した(ハリボテ)版
// - ここではメニューが１つだけ
// - おつりの計算とかも省略
int 食券を買う(int okane) {
      int shokken = 1; // メニュー1つだけ,決め打ち
      return shokken;
}
```

全体を書くとこんな感じかな? それぞれの関数(いわば下請け)に指示を与える情報は、これでok? 例えば「食器を返す」関数はメニュー関係ないので引数不要ですよね?

```
int main() {
      int okane, shokken;

      shokken = 食券を買う(okane);
      調理場に発注(shokken); // 料理を発注
      受け取る(shokken);     // 料理を受け取る
      食べる(shokken);       // 料理を食べる
      食器を返す();          // 引数なし
      return(0);             // OSへの礼儀作法
}
```

(脚注) 本当は、食券を構造体で表現するべきでしょう -> [発展課題]構造体を使って書き直す

じゃんけんを関数化しよう

課題1000番台用のワークシートを再配布するので、それを使ってください

課題1110-1140: じゃんけんを関数化しよう

- 課題110,120,130,140の関数化は、課題190関数化の一部です。 次ページから課題190の関数化を進めますが、 不安な人は、課題110-140の関数化から始めてみてください

課題110-140の流れの復習

1. コンピュータの手を決める
2. キーボードから(ユーザ=jibun)の手を入力
3. じゃんけんの判定を計算
4. 出力

(脚注) みてわかるとおり、 1000番台の課題番号は単に「じゃんけんのオリジナル課題番号 + 1000」です:-)

[例題]課題1190: じゃんけんの関数化

課題190の「じゃんけん」(条件文なし、繰り返しなし)を例にします。

課題190の流れを復習すると、こうですね?

1. 乱数を初期化
2. コンピュータの手を決める
3. キーボードから(ユーザ=jibun)の手を入力
4. じゃんけんの判定を計算
5. 出力

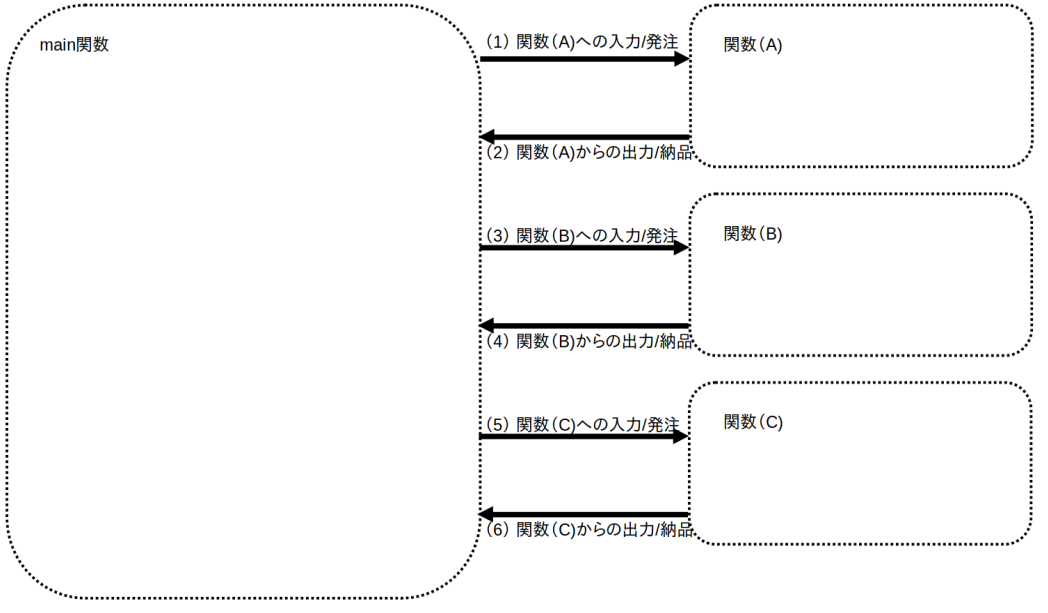
(脚注) 自信のある人は課題群をとばして課題1310(条件文と繰り返しあり版の関数化)へ進んでokです

課題1190: じゃんけん関数化の改造元,課題190

```c
#include <stdio.h>      // 最初の行に必ず書く呪文
#include <stdlib.h>     // 乱数(srand,rand)のため
#include <time.h>       // time()を使うため

int main () {           // 最後にreturn(0)つまり数字を返すからint
   int aite, jibun;     // 整数変数を宣言
   srand(time(NULL));   // 乱数の初期化,一回だけ実行


   aite = rand() % 3;   // コンピュータの手を乱数で決める
   scanf("%d", &jibun); // キーボードから入力した数字を変数jibunに代入

   kekka = (3 + jibun - aite) % 3;

   printf("jibun = %d, aite = %d, kekka = %d\n", jibun, aite, kekka);

   return(0);           // 礼儀作法(OSへstatus = 0を返す)
}
```

演習 課題1190: ワークシートを埋めてください

1. 関数(A)〜(C)を決めてください
2. (1)〜(6)で行き来するデータは何か？
   - 解答には「なし」もありえます (つまり引数なし、返り値なしの場合)
3. 次ページから答え合わせです
   - しばらくThinking Time



別途、このワークシート(束)を渡しますので、書きこんでみてください

課題1190: じゃんけんを関数化: 出力(1)

- 出力の本体は、この行です。(課題190の)この部分を関数outputに移します

```
printf("jibun = %d, aite = %d, kekka = %d\n", jibun, aite, kekka);
```

- 返り値は(mainへ納品するデータ/情報)?
  - 実はありません
  - 本来は「モニタへの出力が『成功したよ』『失敗したよ』を伝えるべき」ですが、 printfは失敗しないだろう〜とサボることが多いです
- 仮引数
  - 渡すべき情報つまり仮引数に必要な変数たちは何ですか?
  - printf行にある変数たちが必要ですね? これらをすべて書いてください
- では、次の関数宣言行を考えてください ... しんきんぐたいむ ...

```
返り値無し 関数名（仮引数）{
  printf("jibun = %d, aite = %d, kekka = %d\n", jibun, aite, kekka);
}
```

課題1190: じゃんけんを関数化: 出力(2)

- 出力の本体は、この行です。これを関数outputに移します

```
printf("jibun = %d, aite = %d, kekka = %d\n", jibun, aite, kekka);
```

- 返り値は無し、このときは void というキーワードを書きます。 コンパイラに「mainに返す情報は無い、心配すんな」とつたえてあげます。 伝えないとコンパイラが「まちがってるんじゃね？」と思って警告してくれることがあります
- 仮引数に必要な変数はprintf行で使う変数たち、つまりjibunとaiteそしてkekkaです。 これらをすべて書いてください
- 関数定義の解答は次のとおりです(仮引数の型も書くことも忘れずに)

```
void output (int jibun, int aite, int kekka) {
  printf("jibun = %d, aite = %d, kekka = %d\n", jibun, aite, kekka);
}
```

課題1190: じゃんけんを関数化: 出力(3)

- main側です。関数の定義は次のとおりでしたね?
  - 返り値なし ... `変数 = 関数` の `変数 =` 部分が無いということです
  - jibunとaiteそしてkekkaを渡します。 `()` の中に関数宣言と同じ順序で書いてください
- 解答: つまり

```
printf("jibun = %d, aite = %d, kekka = %d\n", jibun, aite, kekka);
```

を

```
output(jibun, aite, kekka);
```

に差し替えます。

(脚注) main側を先に関数宣言書いて、そのあと関数宣言を書く派? main側と関数宣言のどちらを先に書くか?は人によりますかね〜 書いていくうちに両方を修正していくことは普通だしなぁ。 main側が先がよいとかいう理屈はないかもしれない

課題1190: じゃんけんを関数化: 出力(4)

```c
#include <stdio.h>      // 最初の行に必ず書く呪文
#include <stdlib.h>     // 乱数(srand,rand)のため
#include <time.h>       // time()を使うため

int main () {           // 最後にreturn(0)つまり数字を返すからint
   int aite, jibun;     // 整数変数を宣言
   srand(time(NULL));   // 乱数の初期化,一回だけ実行


   aite = rand() % 3;   // コンピュータの手を乱数で決める
   scanf("%d", &jibun); // キーボードから入力した数字を変数jibunに代入

   kekka = (3 + jibun - aite) % 3;

   output(jibun, aite, kekka); // !!!関数になった!!!

   return(0);           // 礼儀作法(OSへstatus = 0を返す)
}
```

課題1190: じゃんけんを関数化: 入力(1)

- 同じように「入力部」(scanfの行)を関数inputにしてみましょう
- 返り値と渡す情報はなんでしょうね?
  - ヒント: 変数jibunの値は、このあとmainで使うよね?

... しんきんぐたいむ ...

```
int main () {                       // 最後にreturn(0)つまり数字を返すからint
   int aite, jibun;                 // 整数変数を宣言
   srand(time(NULL));               // 乱数の初期化,一回だけ実行

   aite = rand() % 3;               // コンピュータの手を乱数で決める
   変数 = 入力(引数);               // ???(new) <- ここは、どうなりますか？

   kekka = (3 + jibun - aite) % 3;
   output(jibun, aite, kekka);      // !!!関数になった!!!
   return(0);                       // 礼儀作法(OSへstatus = 0を返す)
}
```

課題1190: じゃんけんを関数化: 入力(2)

- 返り値
  - 関数の中で「キーボードから読みこむ動作」を呼び出します
  - キーボードから読み込んだ情報を返します
  - だから返り値はあります。型は整数(int)
- 引数
  - 細かな指示(発注書)は不要ですよね? -> mainから渡す情報はありません
  - 引数はありません
- では、考えて見てください ... Thinking Time

課題1190: じゃんけんを関数化: 入力(3)

- キーボードから読み込んだ情報(整数値,int)を返す
- 一方、mainから渡す情報はありません。引数がないときは void
- 解答(関数側)

```
int input (void) {
  int jibun;           // 変数の宣言                                  (new)
  scanf("%d", &jibun); // キーボードから入力した数字を変数jibunに代入 (コピペ)
  return jibun;        // 値を返す！                                  (new)
}
```

課題1190: じゃんけんを関数化: 入力(4)

- inputから「キーボードから読みこんだ情報(整数値,int)」が返ります
- この情報は、このあとmainの中でも使うので変数jibunに代入しておきます
- 一方、mainからinputに渡す情報はありません。 この場合、関数呼び出しは単に`()`とします(引数なし)

```
int main () {                          // 最後にreturn(0)つまり数字を返すからint
   int aite, jibun;                    // 整数変数を宣言
   srand(time(NULL));                  // 乱数の初期化,一回だけ実行

   aite  = rand() % 3;                 // コンピュータの手を乱数で決める
   jibun = input();                    // !!!関数になった!!! (new)

   kekka = (3 + jibun - aite) % 3;
   output(jibun, aite, kekka);         // !!!関数になった!!!
   return(0);                          // 礼儀作法(OSへstatus = 0を返す)
}
```

(脚注) 入力の関数化は単純な置き換え作業ではない例です

課題1190: じゃんけんを関数化: 判定+のこり2つ

- 次は判定の関数化です
  - まずは、ここまで
  - じゃんけん3点セットの関数化
- ここまで出来たなら、できれば残りの部分も関数化してみましょう (挑戦課題)
  - できあがると、だいぶ読みやすくなったことが実感できるはずです
  - 残りの部分(２つ)とは、ここです
    - コンピュータの手
    - (乱数の)初期化

課題1190: じゃんけんを関数化: 判定部分を関数化

- 関数 hantei を定義してください。まず考えること、次の2つは何ですか?
  - 入力 = 関数に渡す情報
  - 出力 = mainに返す情報
- 課題190の入力部分を hantei を使い、書き直してください
- main側を先に書き換えると、こう書き換えるでしょう。関数hanteiを考えてください

```
  kekka = hantei(引数); // 引数のところは何を書けばよいでしょうか?
```

```
// output,inputを書き直した段階のソースコード
aite  = rand() % 3;                // コンピュータの手を乱数で決める
jibun = input();                   // 関数化済

kekka = (3 + jibun - aite) % 3;    // 判定部分 <- 今ここ
output(jibun, aite, kekka);        // 関数化済
```

まずは、ここまでの3点セットが出来ればいいね

課題1190: じゃんけんを関数化: aiteの手を関数化

- コンピュータの手を決める部分を関数saikoro3を使い書きなおしてください
- main側を先に書き換えると、こうなるでしょう

```
  aite = saikoro3(引数);
```

  - 引数のところは何を書けばよいでしょうか?
  - 返り値の型は何でしょうか?
- 関数 saikoro3 (3面サイコロ)を定義してください。次の2つは何ですか?
  - 入力 = 関数に渡す情報
  - 出力 = mainに返す情報

解答は最終版の中に

課題1190: じゃんけんを関数化: 初期化部分を関数化

- (初期化といっても)乱数の初期化部分を関数`init`にしてください
  - 入力 = 関数に渡す情報
  - 出力 = mainに返す情報
- じゃんけんプログラムだと短いので意味が無い気もしますが、 もぐらたたきのinit関数は、そこそこ長いので、それの練習台だと思ってください
- main側を先に書き換えると、こう書くでしょう。関数initを考えてください

```
init(引数);  // 引数のところは何を書けばよいでしょうか?
```

(脚注1) 初期化はinitializationなので、たいていinit()という名前の関数にします

(脚注2) 変数宣言を関数に移しちゃダメだよ!?(な〜んでだ?)

課題1190: じゃんけん関数化 最終版: 宣言+関数定義(1)

```c
#include <stdio.h>  // 最初の行に必ず書く呪文
#include <stdlib.h> // 乱数(srand,rand)のため
#include <time.h>   // time()を使うため

void init (void) {
  srand(time(NULL)); // 乱数の初期化,一回だけ実行
}

int saikoro3 (void) {
  int me;
  me = rand() % 3;
  return me;
}
```

(脚注) この頁の関数定義２つがチャレンジ部分、最初できなくてもよい

課題1190: じゃんけん関数化 最終版: 関数定義(2)

```c
int input (void) {
  int jibun;
  scanf("%d", &jibun);
  return jibun;
}

int hantei(int jibun, int aite) {
  int kekka;
  kekka = (3 + jibun - aite) % 3;
  return kekka;
}

void output (int jibun, int aite, int kekka) {
  printf("jibun = %d, aite = %d, kekka = %d\n", jibun, aite, kekka);
}
```

(脚注) この頁が基本となるジャンケン3点セットです。 自力でこの内容を書けてほしい、それが目標

課題1190: じゃんけん関数化 最終版 main

```
int main () {                       // 最後にreturn(0)つまり数字を返すからint
  int aite, jibun, kekka;           // 整数変数を宣言

  init();                           // 乱数の初期化,一回だけ実行
  aite  = saikoro3();               // コンピュータの手を乱数で決める
  jibun = input();                  // キーボードから入力した数字を変数jibunに代入
  kekka = hantei(jibun, aite);      // 判定
  output(jibun, aite, kekka);       // 結果をモニタに出力

  return(0);                        // 礼儀作法(OSへstatus = 0を返す)
}
```

(脚注) 読みやすくなりましたよね? コメントが無くても、変数名と関数名だけ読めば、ほぼ分かるよね?

課題1210: 課題210を関数化してください

- 課題210(190のじゃんけん+条件文)は、判定結果をkachi,make,aikoと表示してくれる課題ですね。これを関数化してみてください
- 課題1190と同じように、意味のある単位で関数に分けていきます
- 考えるべきポイント
  - 条件文は、どこへ持っていくとよいでしょうか?

(脚注) 簡単だと思う人は、この課題をとばして課題310「条件文あり、繰り返しあり」の関数化へ進んでok

課題1310: 課題310を関数化してください

- 課題310は、3回くりかえして遊べる「じゃんけん」プログラムの一応の完成形です。これを関数化してください
- 考えるべきポイント
  - くりかえし文は、どこへ持っていくとよいでしょうか?
  - コードは課題1190, 1210と、ほとんど同じです。 異なるのは「どこからどこまでを繰り返すのか?」の部分だけです、 そこを考えてください。

(脚注) [発展] じゃんけんの発展課題も関数化してみましょう。 発展課題はノーヒント,がんばってください:-)

発展課題 14X0 シリーズ

じゃんけんの発展課題(400番台)も関数化してみてください

- 1410: ジャンケンを何回できるか?(回数の上限を)指定できるようにしてください
- 1420: ジャンケンの回数に上限をなくしてください。 ただし数字の9を入れたら終了です
  - つまり無限ループしていて9が入力されたときには終了するという意味
  - キーボードから入力される値は 0 1 2 9 のいずれかという想定
- 1430: 終了時に、何回勝ったか?も表示してください
- 1440: 終了時に、何勝、何敗、あいこが何回か?を表示してください
- 1450: 課題310をwhile文で書き直してみよう
- 1460: [お作法] キーボードから入力できる値を確認するように変更してください

付録: プロトタイプ宣言の書き方

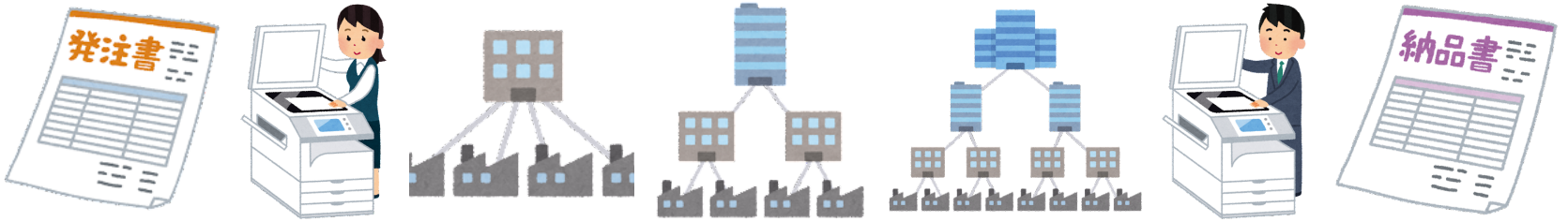
- ぜんぶ書いて動くことを確認してからコピペするといいぞ！？という話

モグラ叩きを関数化しよう

ジャンケンと同様です。注意点だけ簡単に説明します

モグラ叩きを関数化する際の検討事項

- 結論として**グローバル変数の整数配列moguraに仕様変更**しましょう。以下、解説
- ジャンケンと異なり、もぐらのいる位置情報(mogura 整数配列)を使っています
  - 課題690以降では、1次元も9、2次元では9x9=81個の要素がある配列を使いました
  - この配列は、かなり**大きなデータ**(構造)です
- 関数呼び出しとは？
  - main関数から値をコピーして(下請けの関数に引数として渡し)、 下請けの関数も処理結果をコピーし、 (返り値として)main関数に返します
  - (イメージ) 下請けには、 発注書とともに仕事に必要な資料(いまの場合mogura配列)をコピーして渡します。 下請けも処理結果をコピーしてmainに返します。 毎回、膨大な資料の山です。これは大変



グローバル変数

```
変数Gを宣言

返り値の型 関数1（引数）{
    変数Aを宣言

    変数Aを利用できる
    変数Gは利用できる
    関数2を呼び出す
    ... 省略 ...
}

返り値の型 関数2（引数）{
    変数Aは利用できない
    変数Gは利用できる
    ... 省略 ...
}
```

- 関数の中(`{ ～ }`で囲まれている部分)だけで、**変数は有効**です
  - 左の例では、変数Aが、これにあたります
  - main関数でも、その先で呼ばれている関数でも、すべて同じ挙動です
- 関数宣言の外側で変数宣言(左の例では変数G)すると、 場所に関わりなく変数(変数G)が使えるようになります。これが、いわゆる**グローバル変数**(or 大域変数)

(脚注) 関数2で使いたいデータを関数の引数として渡せば、**その値を利用できます**。 関数2の中で、関数1の変数Aを直接操作は出来ません。 関数2の引数で同じ名前の変数Aを割り当てることは出来るので、 見かけ上、関数2の中で変数Aを操作するコードは書けますが、 その操作結果は関数1の変数Aには反映されません (これが、下請けの関数を呼ぶ時にはコピーしてるんだよ！と説明する理由)

演習: グローバル変数(1)

```c
#include <stdio.h>

int mogura[10] = {};

int main () {
    int i = 0;

    i = 2;
    mogura[0] = 1;
    output();
}

void output () {
    printf("mogura[%d] = %d\n", i, mogura[i]);
}
```

- 問1: 間違いを指摘してください
- 問2: 間違いを修正してください

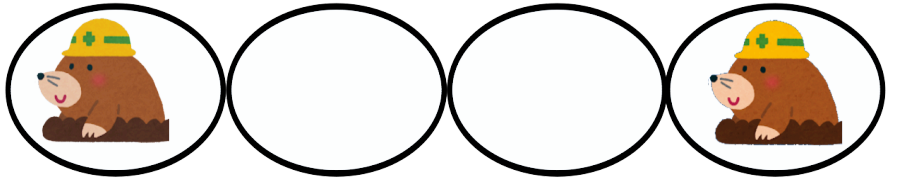
演習: グローバル変数(2)

```c
#include <stdio.h>

int main () {
    int mogura[10] = {};
    init();
    output(0);
    output(1);
}

void init () {
    int i;
    for (i = 0; i < 10; i++) { mogura[i] = i % 2;}
}

void output (int i) {
    printf("mogura[%d] = %d\n", i, mogura[i]);
}
```

- 問1: 間違いを指摘してください
- 問2: 間違いを修正してください

課題510 (再掲)

```
// (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);
}
```



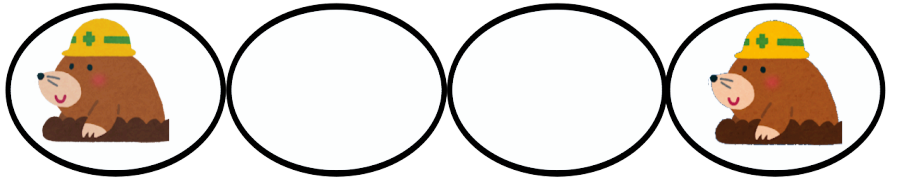
- 穴が4つあるとします。 穴の下にモグラが「いる」「いない」「いない」「いる」状態を表現してください
- 繰り返しを使わず、ベタに代入を4回、出力を4回書いてみましょうか

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1510

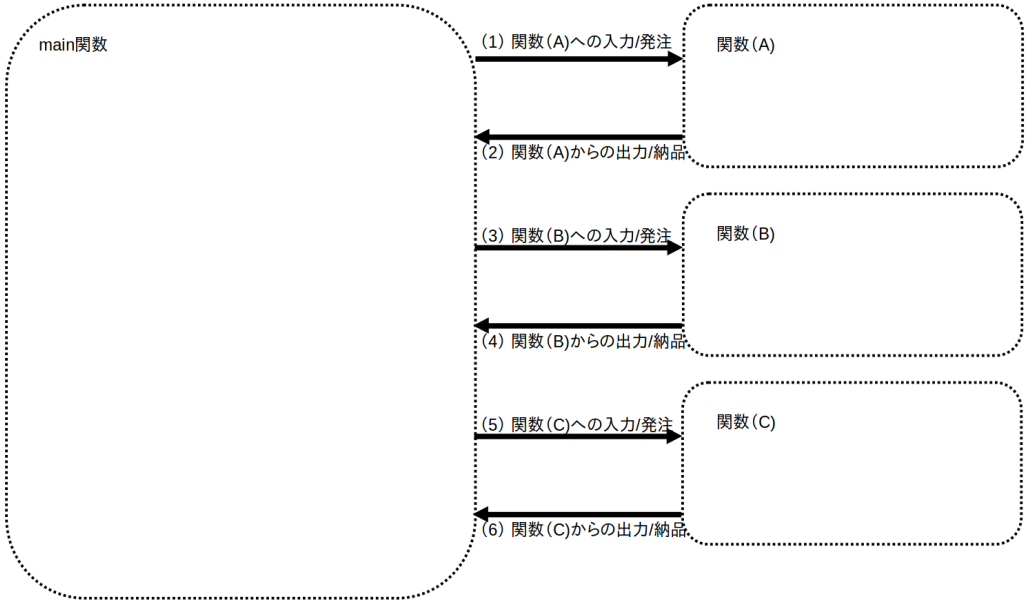
```
// (1) 最初の行に必ず書く呪文

// (2) 整数変数の配列 mogura を宣言

int main () {
  // (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);
}
```



- 課題510を関数化することを考えます
- 関数のワークシートに書きこんでみてください
  - (3)(4)に相当するinputとoutput関数を定義
  - 関数に渡す値
  - 関数から返す値

課題1510のワークシート

1. 関数(A)〜(C)を決めてください
2. (1)〜(6)で行き来するデータは何か？
    - 解答には「なし」もありえます (つまり引数なし、返り値なしの場合)



ジャンケン関数化の際に配布したワークシートと同じです

課題1610

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

  //（3）モグラの位置を乱数で決める

  // 繰り返し部分
     //（4）入力:キーボードからjibunに値を代入
     //（5）判定:もぐらが位置jibunにいるか?
     //（6）出力
}
```

- もう一つ一つやらなくても大丈夫でしょうから、 いくらか飛ばして、 課題610というモグラ叩きのコア部分の完成形を関数化してみましょう

発展課題

- あとは、のこりのモグラの課題を関数化してみましょう

番外編

ここは、もう番外編なのですが、夏休みに向けて少しだけ補足

ポインタ

- グローバル変数ではなく、main関数の中で整数配列moguraを宣言してください
- mogura配列を下請けの関数に渡すように改造してください

```
void main () {
    int mogura[10];
    init(mogura);
    ... 省略 ...
}

void init(int mogura[10]) {

    mogura[0] = 1;
    mogura[1] = 0;
    mogura[2] = 1;
    mogura[3] = 0;
}
```

構造体