ソフト系　C言語実習課題 0
九九の表の作成

v3.22

C言語の入門編として、九九の表を作成します。

- Step0 : 罫線、項目名なしで、9x9の九九の結果を表示します。
- Step1 : Step0に罫線を追加して、9x9の九九の結果を表示します。
- Step2 : Step1に項目名を付けて、10x10の大きさの九九の結果を表示します。
- Step3 : 縦(9)と横(9)の配列を用意し、その配列にそれぞれ1～9の数字を入れ、
  その配列 の内容を用いて、項目名付きで10x10の九九の結果を表示します。
- Step4 : Step3で作成した、縦と横の配列の内容を、それぞれ乱数を用いて入れ替えて、
  九九の表を作成します。
  - ・単に、1～9までの乱数を求めると、同じ数字が重複する場合がある。
  - ・そこで、予め1～9が入っている配列の内容を入れ替えることで、重複しない1～9の数字を求めることができる。
  - ・入れ替えは、新たに関数を作成して行います。
- Step5 : scanf()関数で、キーボードから2～9までの数字を入力して、入力された数字の
  大きさの表を作成する。0 が入力されるまで上記の処理を繰り返します。
  ただし、配列の大きさは縦(9)と横(9)のままで、画面表示だけを入力された数字
  の大きさの枠で表示する。
  - ・入力した数字に合わせて、横罫線の表示合わせる必要がある。
  - ・そのために、横罫線を表示する関数も新たに作成します。

C言語によるプログラミングのヒント

C言語の基礎 – 変数と変数の型、定数

- 変数(variable)とは、プログラム中で値が変更できるもの
  - 変数名は、英文字で始まる任意の英数字で構成(大文字と小文字は区別される)
    例: var, count, ptr ... ← 正しい変数名
    1var, var# ← 誤った変数名。1varは最初が数字、var#は#が含まれている
- 変数の種類として、次の10種類がある → 詳細は次ページを参照
  - char、short、int、longの整数型
  - unsigned char、unsigned short、unsigned int、unsigned longの符号なし整数型
  - float、doubleの実数型

例: int cnt ;　　char ch ;
(int: 型宣言, cnt: 変数名, char: 型宣言, ch: 変数名)

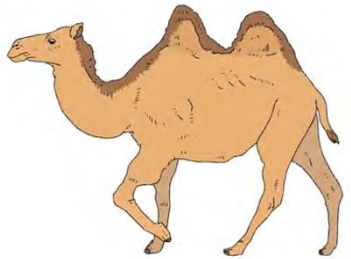
- キャスト(cast) - 型の変換
  - 変数や定数の前に括弧で型を書くことで、型変換が行われる。
  - 例: int cnt=0, *intPtr=&cnt;
    char *charPtr;
    charPtr = (char *) intPtr; // int型ポインタintPtrをchar型ポインタcharPtrに変換
    (キャスト(型変換))
- 定数(constant)とは、値の変更ができないもの
  #define NUMBER 12 ← 定数宣言 (#define文の最後は ; を付けない)

#define は、エディタの置換と同じ働きをする。
上記の例では、ソースコード中にある、NUMBER という文字が 12 に置き換わる。

名前の付け方のルールついて

- 英文字の関数名や変数名には、空白は使えない
  - そのため、いくつかのルール(作法)を定め可読性を良くしている
- キャメルケース
  - ローワーキャメルケース (LCC : Lower Camel Case)
    - 先頭の1文字を小文字で始める
    - 例: getStatus, printStatus, fileName
    - JavaやJavaScriptで使われている
  - アッパーキャメルケース (UCC : Upper Camel Case)
    - 先頭の1文字を大文字で始める
    - 例: GetStatus. PrintStatus, FileName
    - WindowsのAPI、C#、VBAなどで使われている

キャメル(Camel)は、「らくだ」のこと



- スネークケース
  - 英単語を「_」(アンダースコア)でつなげる表記
    - すべて小文字か大文字表記のときに見やすくなる
    - 例: get_status, print_status, file_name, W_LIST、MAX_SIZE

定数は、すべて大文字のスネークケースにする

- C言語では、いずれ付け方でもかまわないが、ソースコード全体で統一する
  - マイルールを決めると良いが、客先でのコーデングルールを優先すること

C言語で扱う変数の種類

分類	変数の型	ビット長	バイト数*1)	記憶できる数値の範囲	備考
整数型	char	8 bit	1	−128～+127	文字型
	short	16 bit	2	−32,768～+32,767	短整数型
	int	32 bit	4	−2,147,483,648～+2,147,483,647	処理系依存*2)
		16 bit	2	−32,768～+32,767	
	long	32 bit	4	−2,147,483,648～+2,147,483,647	長整数型
	unsigned char	8 bit	1	0～+255	符号なしchar型
	unsigned short	16 bit	2	0～+65,535	符号なしshort型
	unsigned int	32 bit	4	0～4,294,967,295	符号なしint型*2)
		16 bit	2	0～+65,535	
	unsigned long	32 bit	4	0～4,294,967,295	符号なしlong型
実数型	float	32 bit	4	-2^{+127}～-2^{-128}, $+2^{-128}$～$+2^{+127}$	単精度実数型
	double	64 bit	8	-2^{+1023}～-2^{-1024}, $+2^{-1024}$～$+2^{+1023}$	倍精度実数型

*1) バイト数は、sizeof(型)、sizeof(変数)で返される値 (sizeof()は型や変数のメモリサイズを返す演算子)。
ただし、引数で受け取った変数は、ポインタのサイズとなるためsizeof() 使用はできないため注意が必要。

*2) int型は、現在では32ビットが一般的だが、8ビットや16ビットのCPUで使用する場合、16ビットとして扱われることもある。正確な変数のサイズは、sizeof() 演算子で確認したほうが良い。

コンピュータが得意なこと – (4) 繰り返し(ループ)

- 繰り返しの代表が、for( )文

【初期値】
変数 cntを0にする

【条件】
cntが5より小さい
限り繰り返し

【増減数】
変数 cntを+1する

```
for ( cnt=0 ; cnt < 5 ; cnt++ ) {
      // この間が繰り返される
      printf("%d¥n", cnt);
}
```

初期値、条件、増減数
は、";"(セミコロン)で区切る

for文とwhile文の使い分け
- 決まった数の単純ループでは、for文を使用する。
  ・ループの中で変数(この例ではcnt)の増減をしない
- while文は、ループの中で変数の増減をする。

結果:

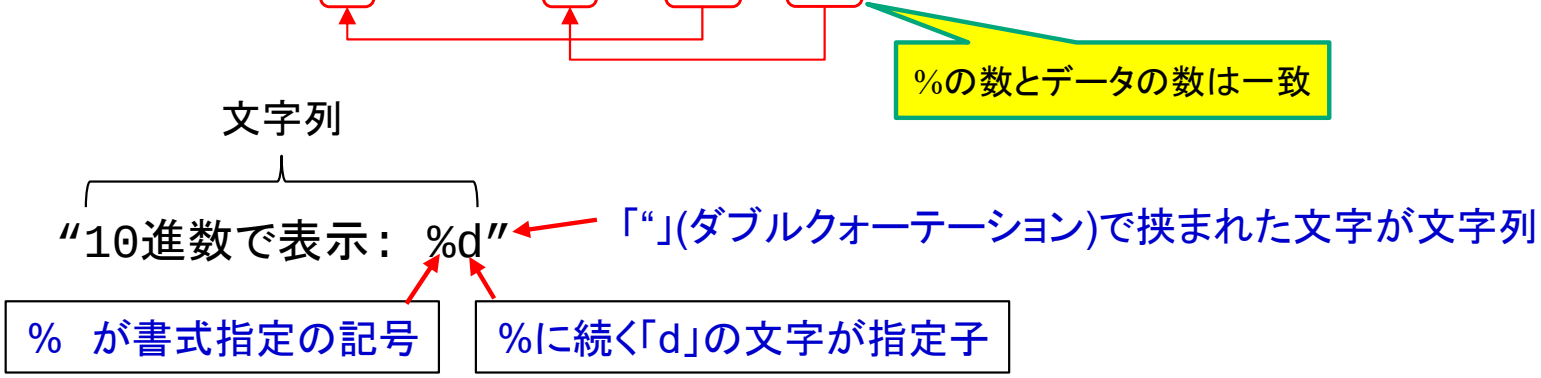
```
0
1
2
3
4
```

上記を実行すると、
変数 cntの値が、0～4まで表示される

C言語の書式(フォーマット指定)とは

- 書式とは、ある数字をどのように表示するかを指定するもの
  - EXCELの表示形式と同じ
- 書式指定は、文字列の中に、%(指定子)と次に続く英文字で指定する
  - %に続く英文字は小文字で指定(16進表示のxとXのみ小文字と大文字を区別する)
  - 同じ値でも、書式指定で色々な書式で表示される(変数 valに16進数で41が入っていた場合)、

```
printf("10進数で表示: %d" , val);  ➔「10進数で表示: 64」と表示
printf("16進数で表示: %x" , val);  ➔「16進数で表示: 41」と表示
printf("文字で表示: %c" , val);    ➔「文字で表示: A」と表示
Printf("文字=%c ,コード=%x", val, val); ➔「文字=A ,コード=41」と表示
```



- %と指定子の間に数字を入れることで数値の桁指定ができる。 重要

%d	➔12	可変桁で表示(数字桁数に関係なく、先頭から表示)
%3d	➔ 12	3桁表示。3桁に満たない場合は、スペースが挿入される
%03d	➔012	3桁で表示だが、3桁に満たない場合は、0が挿入される

C言語の書式一覧(フォーマット指定)

- 下表の中で、最初の4種類は、専門研修のC言語実習で使用します。

  %d (Dicimail:10進数) / %x (heXadecimal : 16進数) / %s (String: 文字列) / %c (Character: 文字)

指定子	対応する型	説明	使用例
%c	char	1文字を出力する	"%c"
%s	char *	文字列を出力する	"%8s", "%-10s"
%d	int, short	整数を10進で出力する	"%-2d","%03d"
%x	int, short ,unsigned int, unsigned short	整数を16進で出力する。%Xは、A～Fが大文字	"%04x"
%p	int, short ,unsigned int, unsigned short	ポインタとして16進数8桁で出力する(%08X と同じ)	"%p"
%u	unsigned int, unsigned short	符号なし整数を10進で出力する	"%2u","%02u"
%o	int, short ,unsigned int, unsigned short	整数を8進数で出力する	"%06o","%03o"
%f	float	実数を出力する	"%5.2f"
%e	float	実数を指数表示で出力する	"%5.3e"
%g	float	実数を最適な形式で出力する	"%g"
%ld	long	倍精度整数を10進で出力する	"%-10ld"
%lu	unsigned long	符号なし倍精度整数を10進で出力する	"%10lu"
%lo	long, unsigned long	倍精度整数を8進で出力する	"%12lo"
%lx	long, unsigned long	倍精度整数を16進で出力する	"%08lx"
%lf	double	倍精度実数を出力する	"%8.3lf"

専門研修で使用

C言語の基礎 – コメントについて

- コメントとは、プログラムを見やすく、わかりやすくするために入れる
  - コメントは、プログラムの動作には全く影響せず、コンパイル時には無視される
  - 適切なコメントは、可読性が良くなる

- C言語が登場した当初、コメントの表記は、 /* … */ で表していた

```
/* これはコメントです */ ← 1行のコメント
/*****************************
  これはコメントです
  このように、複数行にまたがることも可能
*****************************/
printf("%d¥n",a/*+b*/);   ← プログラム中の一部分をコメントにできる
```

/* コメント開始

*/ コメント終了

- その後、C++が登場し、「//」もコメントとして利用できるようになった

```
// これはコメントです
// *****************************
// これはコメントです
// このように、複数行にまたがっては使えない
// *****************************
printf("%d¥n",a+b);   // 「//」以降がコメントとなる
```

// でのコメントは、
その行の終わりまでが有効。
次の行には、影響しない。

可読性の良いプログラムにする

- 他人が読むことを前提にして書こう
  - 時間が経つと書いた本人もわからなくなる場合もある
  - 何だ! これは! と言われないように
- 美しく書く
  - 見やすく書くということ ➔ インデント(字下げ)を忘れずに入れる
  - マジックナンバー(プログラム中に直接書く数字など)は使わない
- 適切なコメントを入れる
  - コンパイラの場合、コメントの量は実行速度に影響しない
- **変数、定数の名前について**
  - 定数は大文字で、変数は小文字にする
  - 意味のある名前を用いる ➔ i , j, k, m, nなどの１文字変数名は使わない
    - 変数や定数は、ソースコードを見たとき、その役割を想像できる名前にする
    - 参考書やWebの例で登場する i , j, k, m, nは、プログラムの説明上使用している
    - 1文字変数は、あとで置換が難しい(最低でも2文字以上にする)

実際の課題

九九の表は、どんな言語でも書ける

- 九九の表はとても簡単な構造で、C言語の知識がなくても、どんな言語でも書けます。
- 実際、C#、Java、PHP、Python、Ruby、VBAで書いてみました。
- もし、C言語で九九の表ができなければ、得意な言語で書いてもかまいません。
- 環境として、C#はVisualStudioですが、それ以外はVScodeを使うと良いです。

- 九九の表で使用する要素としては、次の8種類になります。
  - Step1　for文
  - Step1　桁を揃えて(4桁)の整数の出力
  - Step3　1次元配列
  - Step4　乱数の生成
  - Step4　関数呼び出し(引数あり)
  - Step5　キーボードからの整数値の入力
  - Step5　永久ループ(while)とループからの脱出(break)
  - Step5　if文

- 各言語で、微妙に文法の違いますが、論理的な構造は同じです。

Step0：九九の表(罫線、項目名なし)

- Step0は、下記のように九九の答えのみを表示するプログラムを作成しなさい。

```
1  2  3  4  5  6  7  8  9
2  4  6  8 10 12 14 16 18
3  6  9 12 15 18 21 24 27
4  8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

ヒント: for文の2重ループの構成になります

Step1：九九の表に罫線を追加する

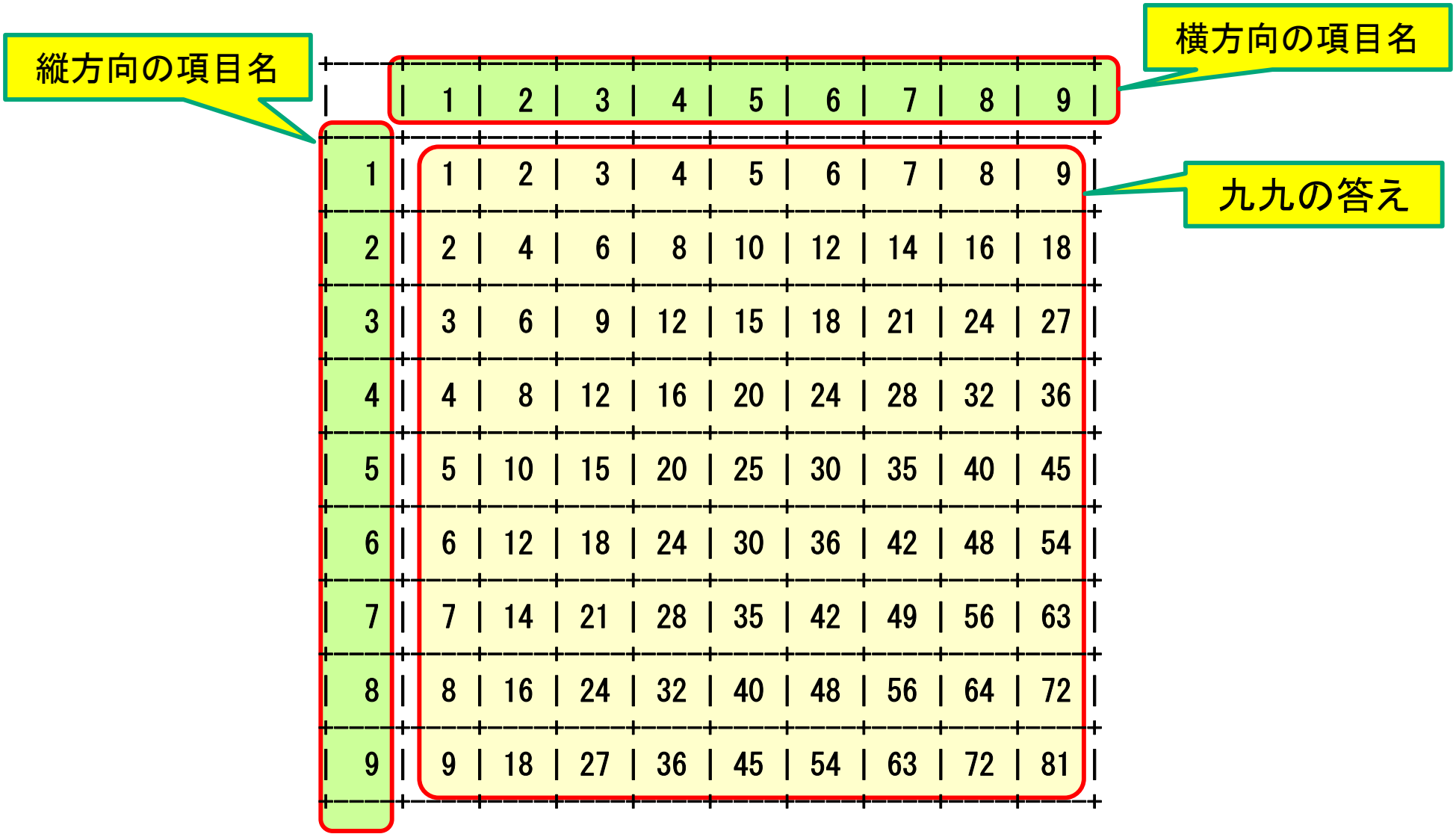
- Step1は、下記のように、Step0に罫線を追加して、九九の答えのみを表示するプログラムにしなさい。

```
+---+---+---+---+---+---+---+---+---+
|  1|  2|  3|  4|  5|  6|  7|  8|  9|
+---+---+---+---+---+---+---+---+---+
|  2|  4|  6|  8| 10| 12| 14| 16| 18|
+---+---+---+---+---+---+---+---+---+
|  3|  6|  9| 12| 15| 18| 21| 24| 27|
+---+---+---+---+---+---+---+---+---+
|  4|  8| 12| 16| 20| 24| 28| 32| 36|
+---+---+---+---+---+---+---+---+---+
|  5| 10| 15| 20| 25| 30| 35| 40| 45|
+---+---+---+---+---+---+---+---+---+
|  6| 12| 18| 24| 30| 36| 42| 48| 54|
+---+---+---+---+---+---+---+---+---+
|  7| 14| 21| 28| 35| 42| 49| 56| 63|
+---+---+---+---+---+---+---+---+---+
|  8| 16| 24| 32| 40| 48| 56| 64| 72|
+---+---+---+---+---+---+---+---+---+
|  9| 18| 27| 36| 45| 54| 63| 72| 81|
+---+---+---+---+---+---+---+---+---+
```

‘+’, ’-’, ’|’ の文字を組み合わせて罫線を表示する

Step2：九九の表に項目名(index)を追加する

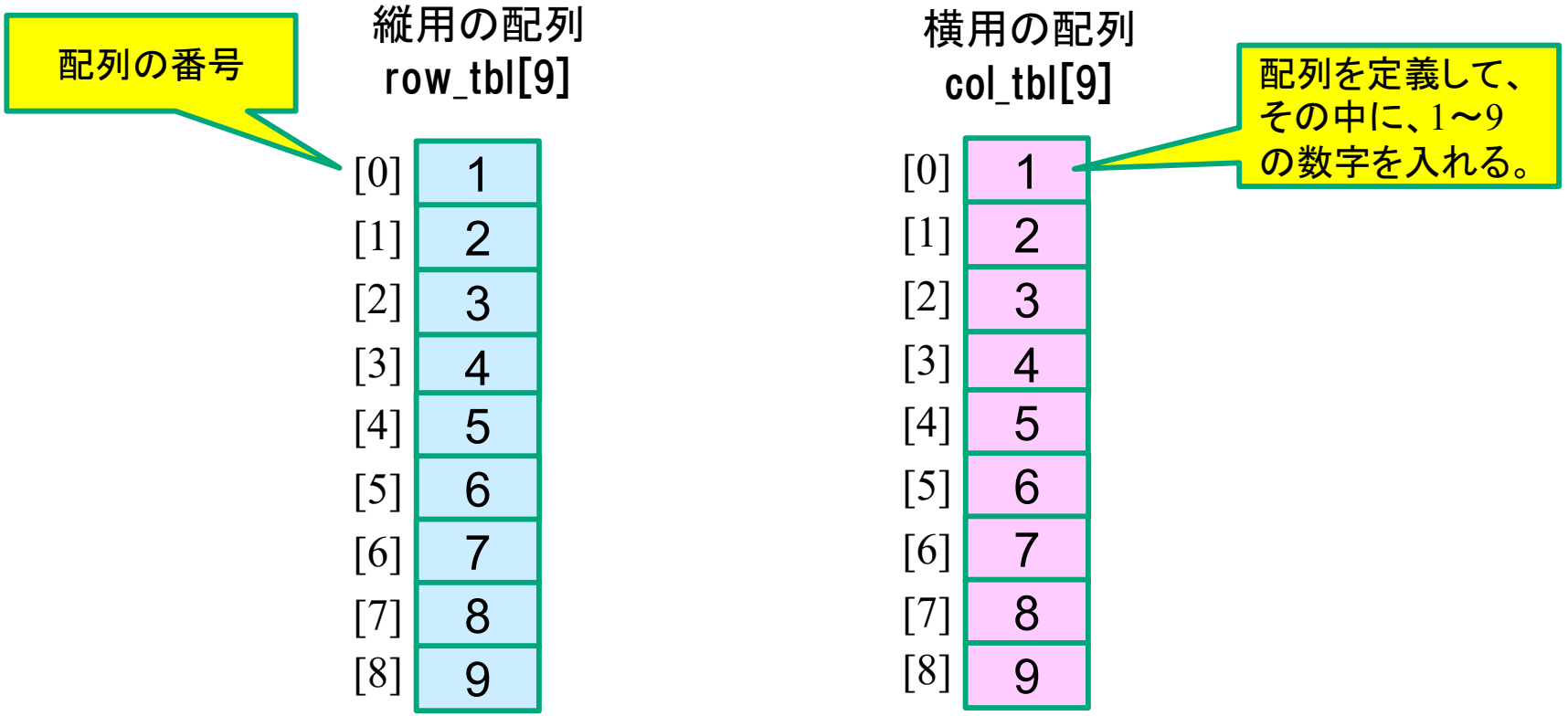
- Step2は、Step1に対して、下記のように、縦と横に項目名を表示するようにしなさい。



	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

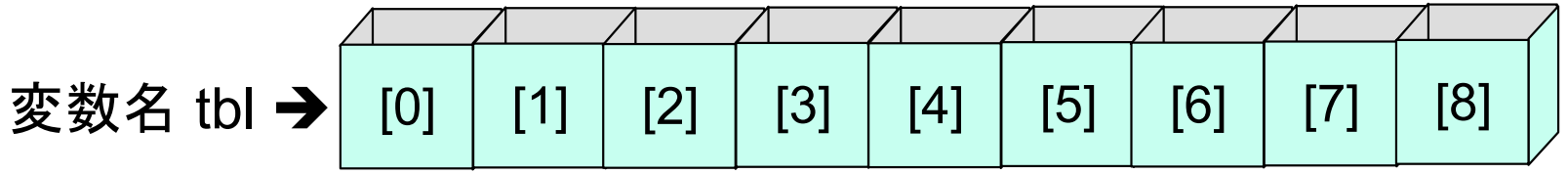
Step3：項目名の部分を配列にする

- Step3は、Step2に対して、縦用と横用に9要素の配列を用意し、その中に、1～9までの数字を入れて、その数字を用いて、Step2の表示を行うようにしなさい。



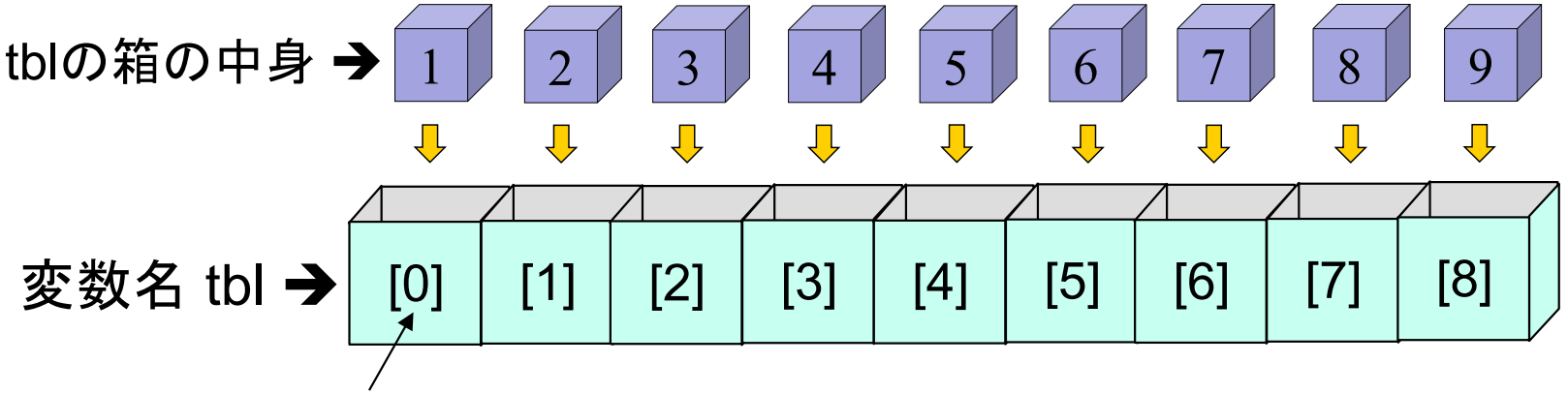
配列(array)とは、箱の集合体

char tbl[9]; // 配列変数の定義(char型のtblという名前の箱を9個定義)



配列(箱)の中にデータを入れる

char tbl[9] = { 1,2,3,4,5,6,7,8,9 }; // tblを定義し、1～9の数字で初期化

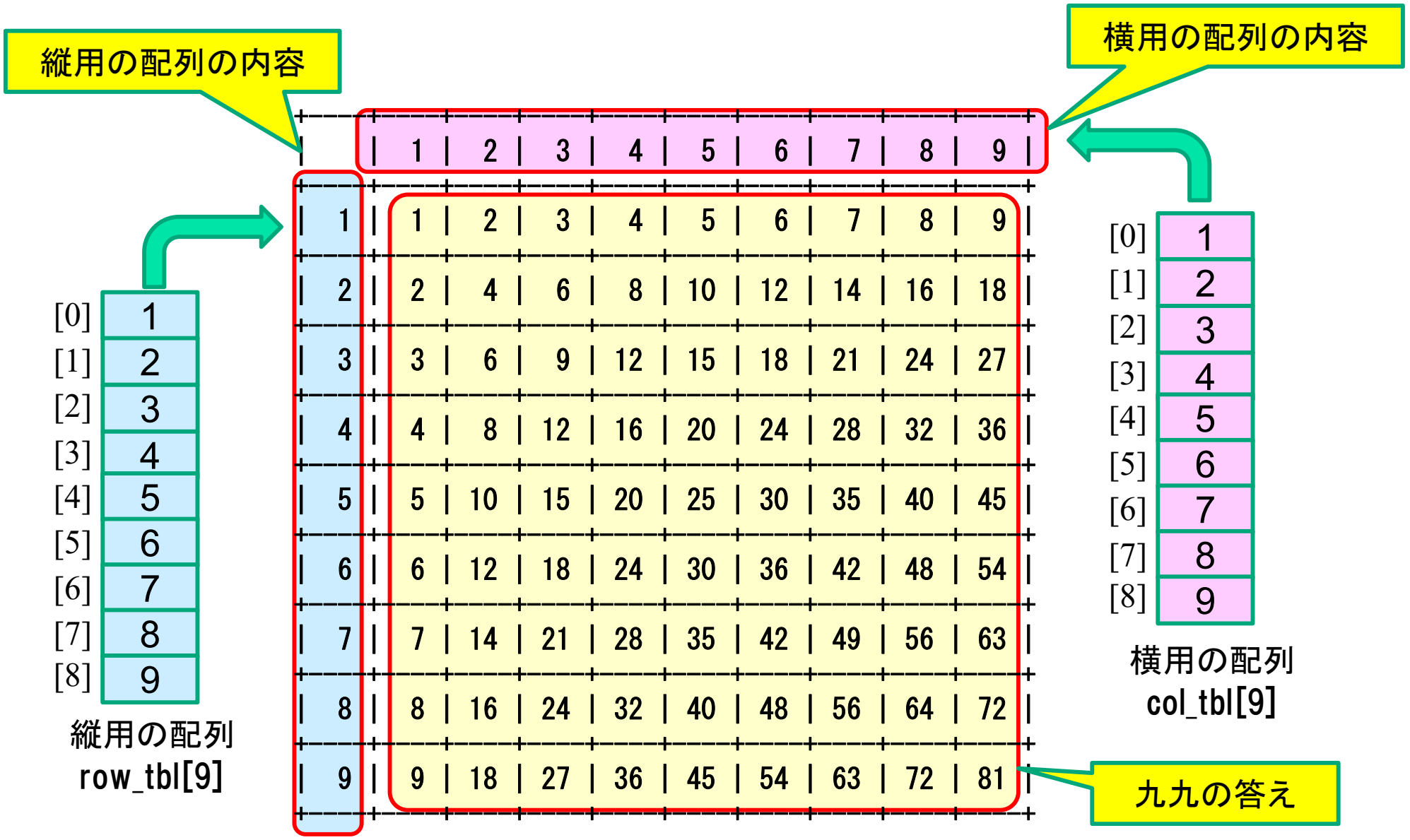


[ ]の中の数字は
tblの箱の番号

tblの箱の番号を指定して、その箱の中に入っている数値が実際の値
上記の例だと、tbl[0]は 1 、tbl[1]は 2、tbl[8]は 9 が入っている

Step3：配列の内容でStep2と同じ表示にする。

- 配列の内容を用いて、Step2と同じ表示にする。



Step4：配列の内容を乱数で入れ替える

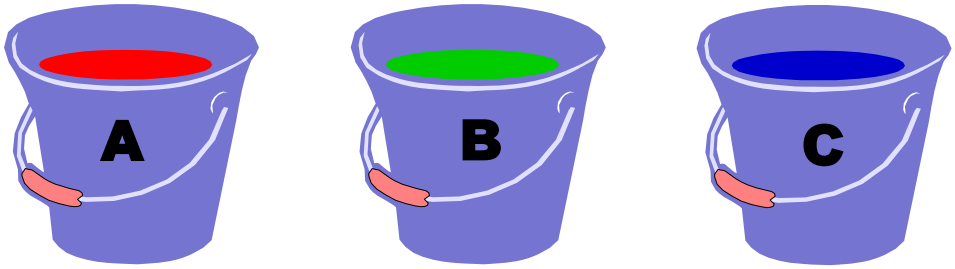
- Step3で、作成した配列の内容を、乱数を用いて入れ替える。
- 乱数で配列の中身の入れ替えは、shuffle( )という関数を作成して行う。

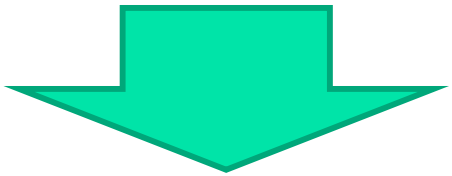
・乱数は、rand関数で得られる。 rand関数を使うには、#include <stdlib.h>が必要。
・単に、1～9まで乱数を求めると、同じ数字が重複する場合がある。
・そこで、予め1～9が入っている配列の内容を入れ替えることで、重複しない1～9の数字を求めることができる。
・また、rand関数は、何もしないで使用すると乱数のシード(種)が同じため、毎回同じパターンの乱数になる。
・そこで、最初に、srand((unsigned) time(NULL)); を実行して、乱数計算のシード(種)を変更する。
・これは、現在の時間を種として設定するものである。time関数を使うためには、#include <time.h>が必要。
・そして、rand()で求めた乱数から、9の余りを求めると、0～8の数字が得られる。
・その0～8の数字を使って、配列の内容を入れ替える。

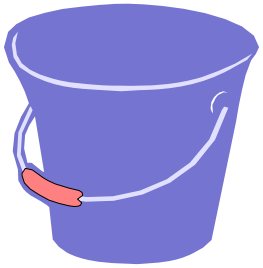
配列の入れ替えのヒント

- 下記ように、3つのバケツに色付きの水が入っている。
- そのバケツの水を入れ替えるにはどうしたら良いかを考えてみよう。

入れ替え前

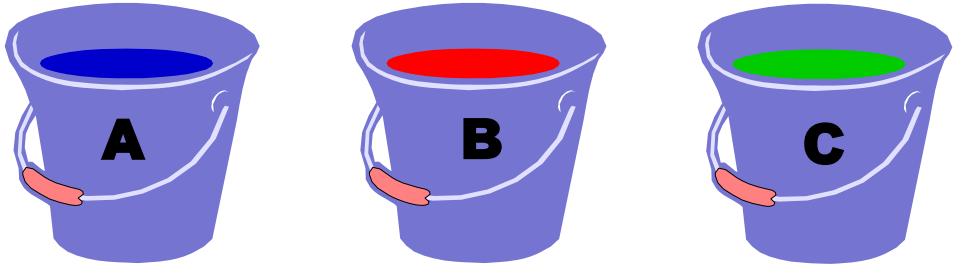






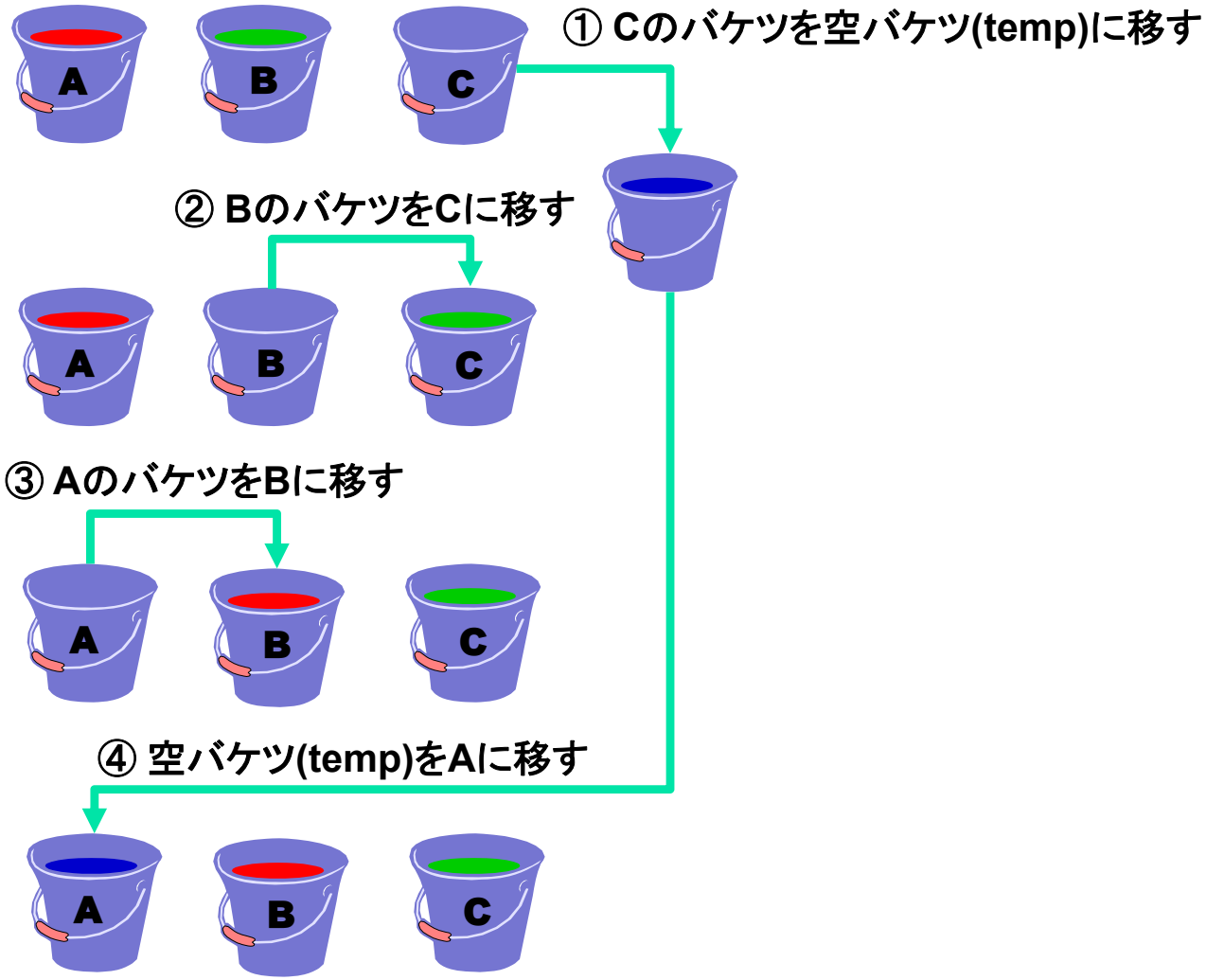
空のバケツが1個
あったら可能

入れ替え後

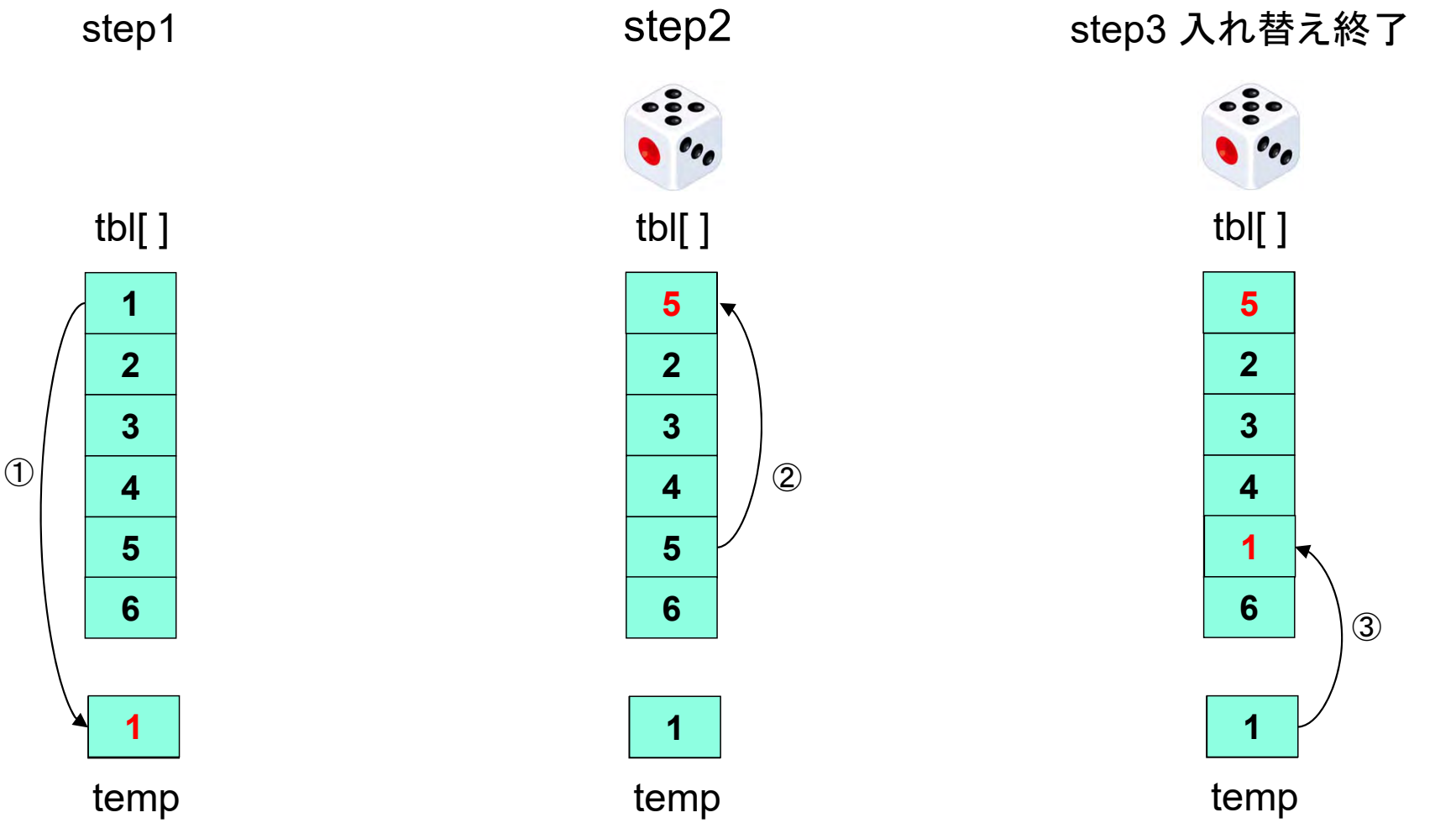


配列の入れ替えのヒント

- 空バケツをtempという名前にしたとすると、

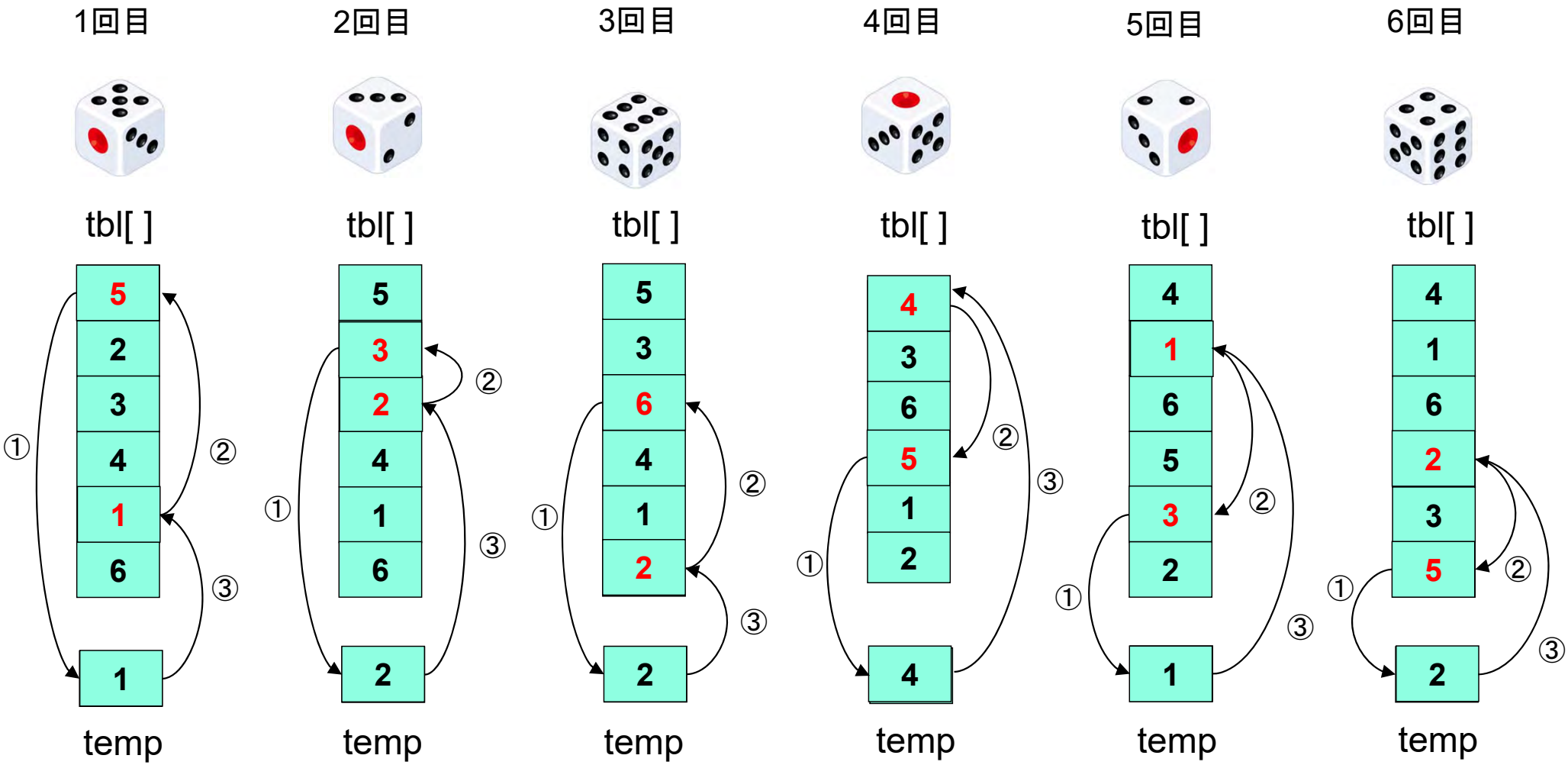


配列の入れ替えの動作



ここの例では、説明を分かりやすくするために、箱番号を1から始めていますが、実際のプログラムでは、箱番号は0から始まります。

実際のシャッフルの動き: 同様に繰り返す



必要な変数

```
int cnt;      // 処理をしている配列番号
int temp;     // 空バケツ
Int rndval;   // 乱数の値
```

新たに関数を作成する

関数の基本形と使い方

- 関数の4種類の基本形 ➔ 下記の4種類の基本形は覚えましょう

	呼び出し側	関数 (呼ばれる側)

ケース1
引数なし
戻り値なし
	funcA();	void funcA(void){
		/* 処理 */
		}

ケース2
引数なし
戻り値あり
	ret = funcB();	型宣言 funcB(void){
		/* 処理 */
		return 戻り値;
		}

ケース3
引数あり
戻り値なし
	funcC(引数1,引数2, …);	void funcC(型 引数1, 型 引数2, ….){
		/* 処理 */
		}

ケース4
引数あり
戻り値あり
	ret = funcD(引数1,引数2, …);	型宣言 funcD(型 引数1, 型 引数2, ….){
		/* 処理 */
		return 戻り値;
		}

重要

呼び出し側は、直接値か、変数名のみ

関数の型がvoid以外は、return と戻り値が必要

関数側は、型宣言が必要

・引数の名前は違っていても構わないが、順番は合わせなければならない。
・関数が引数を受け取らない場合は、関数の( )の中はvoidと記載
・関数が値を返さない(戻り値がない)場合は、関数名の先頭にvoidと記載

関数を作る(関数の定義)

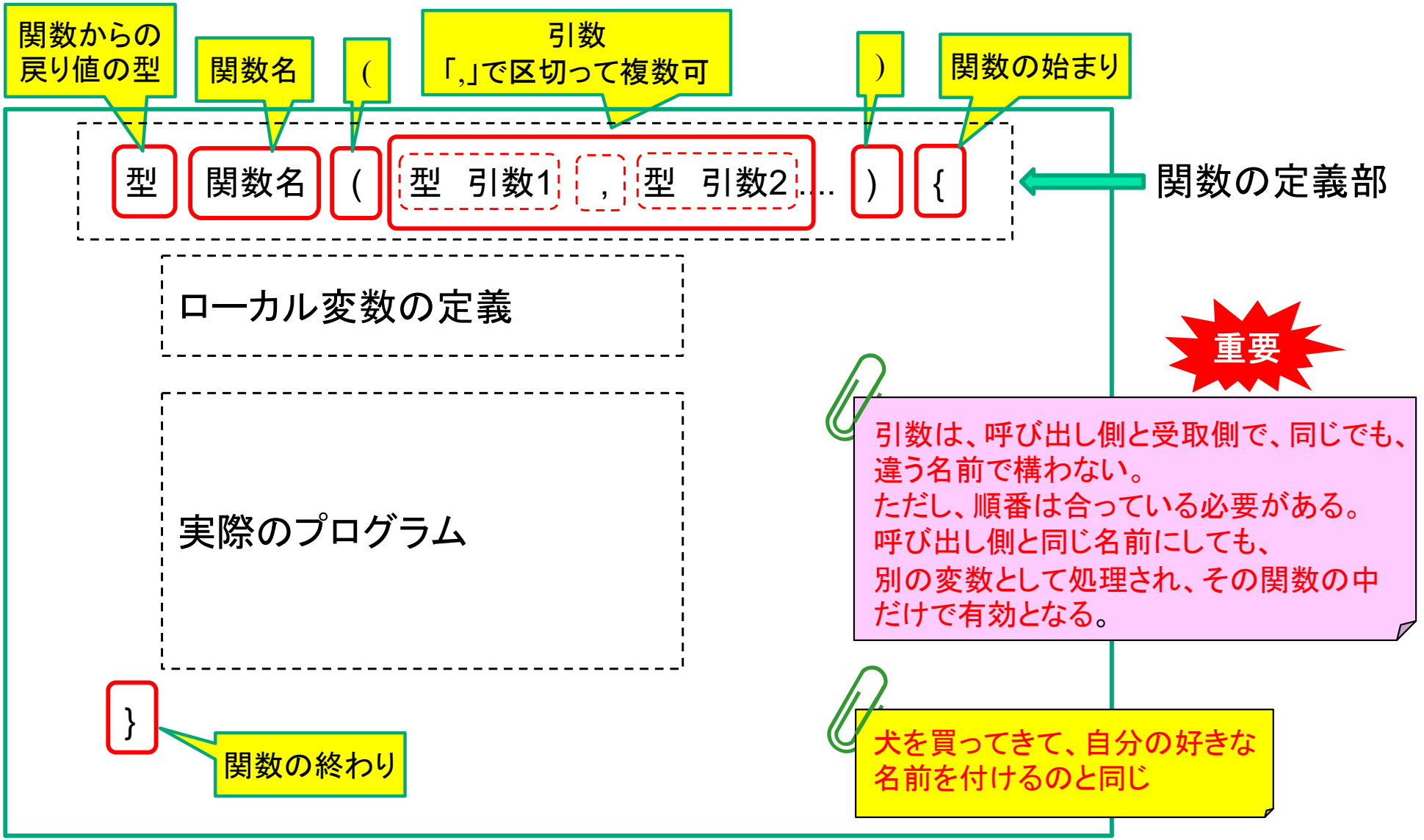
- 関数を作る時、その関数の責務(何をする関数)を明確する
  - 何をする関数か? ← 関数名も何をするか想像できる名前にする
  - 入力(引数)は何か?
  - 出力は何か?
  - 戻り値は何か? ← 戻り値によって 関数の型が決まる

【出力と戻り値の違い】

・出力とは、その関数を実行して書き換えされたり、ファイルや画面に出力されるもの
　引数に配列やポインタで受けて、配列の中やポインタの差す先を変更したりすることも含まれる
・戻り値とは、単純にその関数が呼び出し元に返す値。返す値は1つだけ

- 例:
  - 入力された数字を1から足した数を計算して返す関数を作る
  - 関数名は、`calcSum`
  - 引数は、`int` 型の数字
  - 出力はなし
  - 戻り値は、計算結果で `int` 型の数値 ← 関数の型は `int` 型

関数を作る(関数の構成)


関数からの
戻り値の型
関数名
(
引数
「,」で区切って複数可
)
関数の始まり
型
関数名
(
型　引数1
,
型　引数2
...
)
{
関数の定義部
ローカル変数の定義
実際のプログラム
}
関数の終わり
重要
引数は、呼び出し側と受取側で、同じでも、違う名前で構わない。
ただし、順番は合っている必要がある。
呼び出し側と同じ名前にしても、
別の変数として処理され、その関数の中だけで有効となる。
犬を買ってきて、自分の好きな名前を付けるのと同じ

関数を作る(実際の関数例)

```
#include <stdio.h>

// プロトタイプ宣言
int calcSum(int num);

int main(void) {

    int num=10;
    int sum;

    sum = calcSum(num);
    printf("sum=%d", sum);

    return 0;
}


// 入力された数字を1から足した数を計算して返す
int calcSum(int num) {

    int sum = 0;          // 合計値のクリア

    while (num > 0) {     // numが0より大きい限りループ
       sum += num;        // 加算する
       num--;             // numを減らす
    }
    return sum;           // 計算した合計値を返す
}
```

プロトタイプ宣言は、関数名の1行をコピーして、最後を；(セミロロン)にするだけ

ローカル変数の定義部

プログラム部

関数の呼ぶ

重要

C言語では、呼び出す関数が後ろ(前方参照)にあると、エラーとなる。そのため、使用する関数を前もって定義をしておく必要がある。このことをプロトタイプ宣言という。

ローカル変数の定義部

プログラム部

shuffle() 関数を作る

- shuffle( )関数:
  - 機能:　　引数で与えられた1次元配列の中を乱数で入れ替える。
  - 関数名:　shuffle
  - 引数:　　char型の1次元配列　🡐 char tbl[ ]
  - 出力:　　入れ替わった配列
  - 戻り値:　なし　🡐 戻り値がないので、関数の型はvoidとなる

- 上記を元にすると、関数定義は下記のようになる

```
void shuffle(char tbl[ ]){      🡐 1次元配列を引数として受け取る


}
```

- ローカル変数として、3個必要

```
int cnt;        // 処理をしている配列番号
int temp;       // 空バケツ
Int rndval;     // 乱数の値
```

Step4：乱数を使って九九の表を作成する

- Step4は、Step3の応用で、項目名が単に1～9でなく、乱数で作成した重複しない配列の数字を作って九九の問題を作成します。

乱数で作成した重複しない1～9の数字を使用する。

	8	2	7	4	5	6	9	3	1
6	48	12	42	24	30	36	54	18	6
2	16	4	14	8	10	12	18	6	2
5	40	10	35	20	25	30	45	15	5
4	32	8	28	16	20	24	36	12	4
9	72	18	63	36	45	54	81	27	9
7	56	14	49	28	35	42	63	21	7
8	64	16	56	32	40	48	72	24	8
1	8	2	7	4	5	6	9	3	1
3	24	6	21	12	15	18	27	9	3

乱数で作成した重複しない1～9の数字を使用する。

Step5：表示する枠数を入力して作成

- Step5はStep4の応用です。scanf()を使って数値を入力し、その数値の大きさの枠数で表示する。0が入力されるまで繰り返す。配列の大きさは縦(9)と横(9)のままで、画面表示だけを入力された数字の大きさの枠で表示する。

九九の表の枠数(2～9)を入力してください 0は終了:2

	2	1
1	2	1
2	4	2

0 が入力されるまで繰り返す。

九九の表の枠数(2～9)を入力してください 0は終了:4

	3	1	5	6
3	9	3	15	18
2	6	2	10	12
1	3	1	5	6
6	18	6	30	36

九九の表の枠数(2～9)を入力してください 0は終了:7

	5	3	1	6	2	4	7
3	15	9	3	18	6	12	21
6	30	18	6	36	12	24	42
1	5	3	1	6	2	4	7
2	10	6	2	12	4	8	14
7	35	21	7	42	14	28	49
4	20	12	4	24	8	16	28
5	25	15	5	30	10	20	35

横罫線の表示は、枠数に応じて表示にするため、関数にする
例: `void prt_line(char *lineStr, int num)`

キーボードからデータの入力 - scanf() 関数

- キーボードからデータの入力で、最も一般的な関数として scanf( )が使われる
  - scanf( )は、書式を指定して、文字や数値の入力ができる
  - VisualStudio2019以降は、戻り値のある関数で戻り値を使わない場合は、先頭に(void)を付けないと警告がでる

- 単純に10進数の数値入力

```
int intval;
printf("数字を入力してください:");
(void)scanf ("%d", &intvar);
```

scanf()の前に何を入力するかを表示させる(¥nは不要)

数値の場合、入力する変数の前に'&'を付ける

- 複数の数値を入力を一度に入力

```
int year, month, day;
printf("年/月/日を入力してください:");
(void) scanf ("%d/%d/%d", &year, &month,&day);
```

/で区切って、年/月/にを入力する

- 文字列の入力

```
char buf[256];
printf("ファイル名を入力してください:");
(void)scanf ("%s", buf);
```

文字列を読込み配列を定義

配列の場合、変数の前に'&'を付けない

scanf()を使うには、プログラムの先頭に #pragma warning(disable:4996) を入れないとエラーになります。
また、scanf()はキーボードからの入力だけではありません。ファイルや配列からも入力できます。
詳しくは、C言語の基礎の後半の「標準入出力とデータの入出力」で説明をしています。

Step5 のフローチャート