Lesson 3
じゃんけんゲーム

本 Lesson では《じゃんけんゲーム》を作成します。最初は単純なものを作り、少しずつ手を加えて機能を追加していきます。

この Lesson で学ぶおもなこと

- switch 文
- 条件演算子・条件式
- 文字コード
- 漢字コード
  （JIS / シフトJIS / EUC）
- 漢字を含んだ文字列
- ワイド文字
- ポインタによる文字列の走査
- 文字列の配列
  （２次元配列 vs ポインタの配列）
- 関数
- 識別子の有効範囲

- wchar_t 型
- isprint 関数
- CHAR_BIT
- CHAR_MAX

3-1 じゃんけんゲーム

本 Lesson では、コンピュータとプレーヤとが対戦する《じゃんけんゲーム》を作成します。

基本設計

まずは大まかな設計をしましょう。プログラムの流れを次のようにします。

1. コンピュータの手を決定する。
2. 「じゃんけんポン」と表示して、プレーヤが手を入力する。
3. 勝敗の判断を行い、その結果を表示する。
4. 続行するかどうかをたずね、プレーヤが希望すれば1に戻る。

次に、各ステップを少し詳しく設計していきます。

1　コンピュータの手を乱数で決定します（具体的な値は2で設計します）。

もしプレーヤの手を読み込んだ後だと、コンピュータが勝つように作為するというズルも可能です（p.83 で検討します）から、プレーヤの手を読み込む2よりも前に行います。

2　プレーヤに **"グー"**、**"チョキ"**、**"パー"** と入力させると、グーやパーの長音記号をマイナス記号に間違えるといった**タイプミス**が起こるかもしれません。

次のように表示して選んでもらうことにします。

```
じゃんけんポン　…　(0) グー　(1) チョキ　(2) パー：
```

グー、チョキ、パーの手を、順に `0`, `1`, `2` に対応させます（**Fig.3-1**）。

プレーヤの手とコンピュータの手を同じ値にすると、一貫性が保てるため好都合です。これで、1の設計で未解決だった手の値も決定です。

3　コンピュータとプレーヤの手から、勝敗を判断します。

両者が出す手の組合せは、3 × 3 の 9 通りあります。その組合せと勝敗の関係をまとめたのが **Fig.3-2** です。

● **Fig.3-1**　手とその値

	グー	チョキ	パー
グー	引分け	勝ち	負け
チョキ	負け	引分け	勝ち
パー	勝ち	負け	引分け

コンピュータの手

プレーヤの手

※ プレーヤ側の勝敗を表す。

● **Fig.3-2**　手と勝敗

プレーヤとコンピュータの手を *user* と *comp* とすると、勝敗は次のように判定できます（**Fig.3-3**）。

a 引分け

user と *comp* の値が等しければ引分けであることは自明です。*user* - *comp* の値は 0 となります。

b プレーヤの勝ち

図に示すように、0, 1, 2, 0, 1, 2, … という循環において、矢印の**始点**側が勝ちで、**終点**側が負けです。

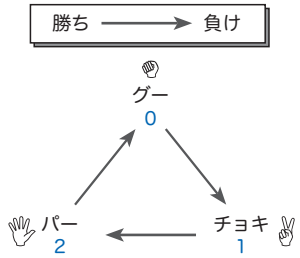
プレーヤが始点、コンピュータが終点となる組合せが、プレーヤの勝ちです。このとき、*user* - *comp* の値は -1 または 2 となります。

c プレーヤの負け

プレーヤが終点、コンピュータが始点となる組合せが、プレーヤの負けです。このとき、*user* - *comp* の値は -2 または 1 となります。

a 引き分け

user	*comp*
0	0
1	1
2	2



b プレーヤの勝ち

user	*comp*
0	1
1	2
2	0

c プレーヤの負け

user	*comp*
0	2
1	0
2	1

● **Fig.3-3**　勝敗の判定

三つの判定は、共通の式（*user* - *comp* + 3）% 3 で行えます。この値が 0 であれば引分け、1 であればプレーヤの負け、2 であればプレーヤの勝ちです。

4　これについて詳しい説明の必要はないでしょう。

swtich 文

ここまでの設計をもとに作成したプログラムを **List 3-1** に示します。

コンピュータの手の表示と判定結果の表示は、switch 文によって行っています（網かけ部）。

switch 文の構文は、右のとおりです。

switch文の構文

```
switch（式）文
```

Lesson 3

List 3-1

```
/*
    じゃんけんゲーム（その１）
*/

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

int main(void)
{
    int user;                   /* プレーヤの手 */
    int comp;                   /* コンピュータの手 */
    int judge;                  /* 勝敗 */
    int retry;                  /* もう一度？ */

    srand(time(NULL));              /* 乱数の種を初期化 */

    printf("じゃんけんゲーム開始!!\n");

    do {
        comp = rand() % 3;      /* コンピュータの手（0～2）を乱数で生成 */

        printf("\n\aじゃんけんポン … (0)グー (1)チョキ (2)パー：");
        scanf("%d", &user);     /* プレーヤの手を読み込む */

        printf("私は");                         /* コンピュータの手を表示 */
        switch (comp) {
         case 0: printf("グー");    break;
         case 1: printf("チョキ");  break;
         case 2: printf("パー");    break;
        }
        printf("です。\n");

        judge = (user - comp + 3) % 3;      /* 勝敗を判定 */

        switch (judge) {
         case 0: puts("引き分けです。");         break;
         case 1: puts("あなたの負けです。");     break;
         case 2: puts("あなたの勝ちです。");     break;
        }

        printf("もう一度しますか … (0)いいえ (1)はい：");
        scanf("%d", &retry);
    } while (retry == 1);

    return (0);
}
```

プログラムの流れは、**case** に続く値が**式**を評価した結果と一致する**ラベル**へと移ります。

ただし、一致するラベルがない場合、**default** ラベルがあればそこに移動し、なければ **switch** 文を抜け出します。

ラベルに飛んだ後は、**break** 文に出会うまで書かれている文を順次実行します。**Fig.3-4** のプログラムと、その流れを示した図を見比べれば理解できるでしょう。

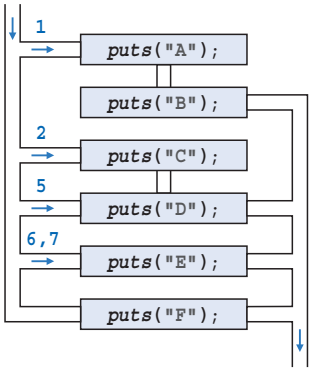
実行例

```
じゃんけんゲーム開始!!

♪じゃんけんポン … (0)グー (1)チョキ (2)パー：2↵
私はグーです。
あなたの勝ちです。
もう一度しますか … (0)いいえ (1)はい：1↵

♪じゃんけんポン … (0)グー (1)チョキ (2)パー：1↵
私はチョキです。
引き分けです。
もう一度しますか … (0)いいえ (1)はい：0↵
```

```
switch (sw) {
 case 1  : puts("A");
           puts("B");  break;
 case 2  : puts("C");
 case 5  : puts("D");  break;
 case 6  :
 case 7  : puts("E");  break;
 default : puts("F");  break;
}
```



● **Fig.3-4**　switch文によるプログラムの流れの分岐

▶ 以下に示す **if** 文と **switch** 文（いずれも同じ動作をします）を比較しましょう。

先頭三つの **if** は *va* の値を、最後の **if** は *vb* の値を調べています。変数 *vc* に 80 が代入されるのは、*va* が 1，2，3 のいずれでもなく、かつ *vb* が 4 であるときです。

連続した **if** 文において、分岐のための比較対象となるのは、必ずしも単一の式であるとは限りません。最後の判定を、**if** (*va* == 4) と読み違える人や、**if** (*va* == 4) の書き間違いではないかと勘ぐる人もいるでしょう。

その点、**switch** 文のほうは、全体の見通しがよいため、プログラムを読む人が、そのような疑念を抱くことも少なくなります。

```
if (va == 1)
    vc = 10;
else if (va == 2)
    vc = 20;
else if (va == 3)
    vc = 50;
else if (vb == 4)
    vc = 80;
```

```
/* 左のif文を書き直したもの */
switch (va) {
 case 1  : vc = 10;  break;
 case 2  : vc = 20;  break;
 case 3  : vc = 50;  break;
 default : if (vb == 4) vc = 80;
}
```