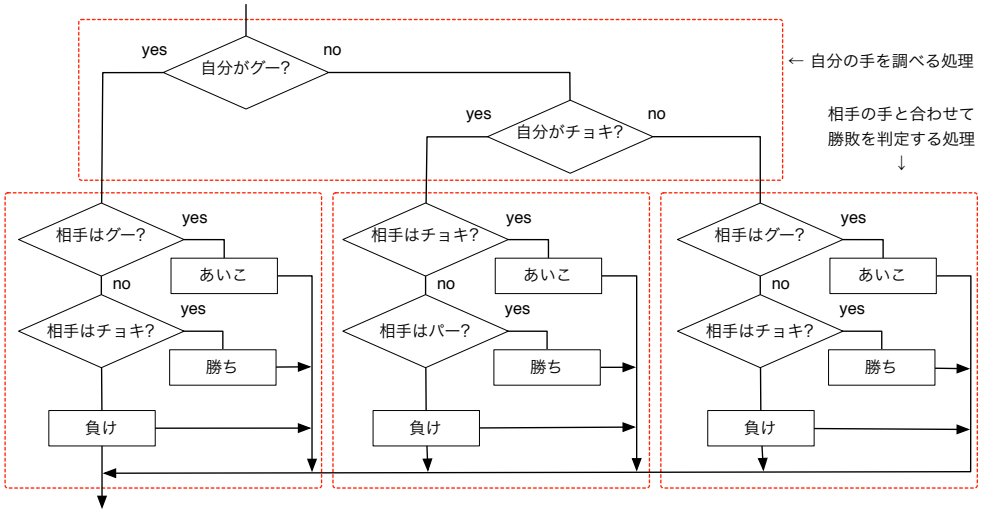
■ じゃんけんプログラム（解説）

幾つかのアルゴリズムがあり得る。問題の性質に注目すれば、より整理された良いコードになる。

□ 網羅的な場合分け

自分の手によってまず処理が三つに分岐し、その先で相手の手によって処理が三つに分岐する。
つまり双方の「手」に注目して記述するわけで、結果的にすべての場合（9 つ）に対応する処理を個別に書くことになる。

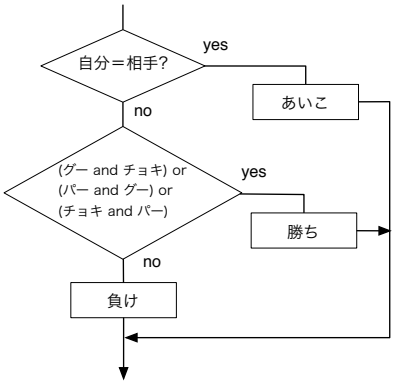


理屈は単純、コードは冗長。

□ 勝ち負けの状態（三種類）に分ける

逆に、あいこ、勝ち、それ以外（つまり負け）が成立する際の条件に注目すると記述量がかなり減る。まず「あいこ」については「自分と相手の手が同一」であることを確認すれば良い。
次に「勝ち」の判定は以下のように論理演算子（||, &&）を使って一つの if 文にまとめることができる。

```
if(   ( ( me == 1 ) && ( you == 2 ) )   // 自分がグーだった場合、
   || ( ( me == 2 ) && ( you == 3 ) )   // 自分がチョキだった場合、
   || ( ( me == 3 ) && ( you == 1 ) ) ) { // 自分がパーだった場合、
```



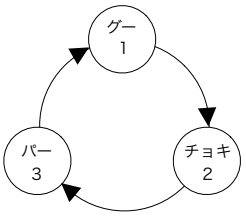
場合分けを減らしたぶんだけコードが整理されて単純に。使用している言語の記述能力も助けになった。

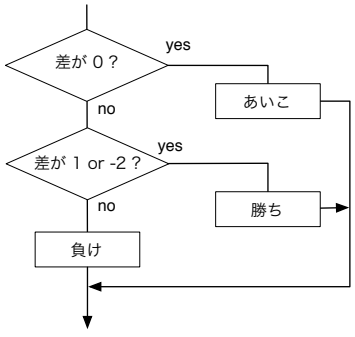
（論理演算子については「条件分岐と繰り返し」あるいは教科書 5.6 参照。）

□ 場合を直接的に判断せず、演算によって簡略化する

グーチョキパーがリング状の勝ち負け関係になっている（剰余系になっている）ことに注目すれば、記述はもっとシンプルになる。
つまり相手の手と自分の手の差分、つまり you - me がゼロなら「あいこ」、1 あるいは -2 なら「勝ち」、それ以外（実際には 2 あるいは-1）なら「負け」で良い。

```
diff = you – me;
if( ( diff == 1 ) || ( diff == -2 ) ) { // 勝ち
```





数の性質を利用して更に単純になった。

（もう少し剰余演算をうまく使うことでもっと単純にできる）

プログラムは動けば良いわけではない。どう書くか、が重要。
より良いアルゴリズムと、より良いコードの検討がプログラミングの本質。