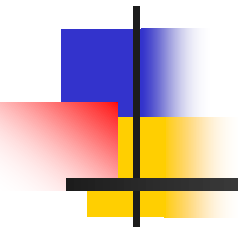
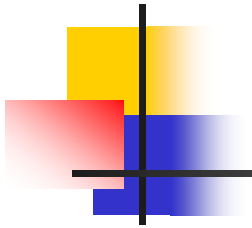


プログラミング言語実験

第3回：コンピュータ大貧民
（コンピュータ大貧民の実行）



コンピュータ大貧民とは

大貧民

1. トランプゲームのひとつ。
   色々な名前がある
   （大富豪、等）。
2. 地方ルールが沢山ある。
3. 日本で生まれたゲームだと言われており、海外では、
   ほとんどプレイされていない。
4. 配られた手札（カード）を、早く無くした人が勝ち。
5. 不完全情報ゲーム。
6. 大貧民をプレイするプログラムの大会が毎年開催。

コンピュータ大貧民大会

大貧民をプレイするプログラムを作る大会

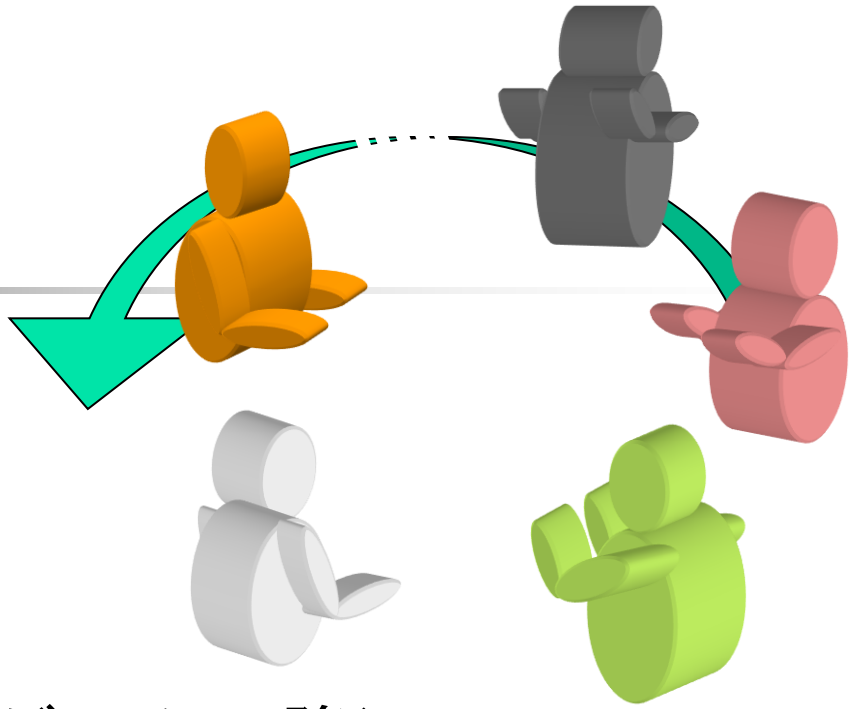
UEC コンピュータ大貧民大会

第4回UEC大貧民大会を、以下のように開催致します。

主催:	UEC(電気通信大学)
共催:	情報オリンピック日本委員会
協賛:	静岡県立大学経営情報学部
会場:	UEC(電気通信大学) 東京都調布市
日時:	2009年11月22日(日)
対象:	どなたでも御参加頂けます

大貧民大会で検索

ゲームの流れ



以下を，繰り返す．

① カードを全員に配る

② 大富豪は大貧民と，
富豪は貧民と，
カードを交換する（第2ゲーム以降）

③ 時計回りに，カードを出していく．

④ 早く手持ちのカードを無くした人から，
高い身分になっていく．

座っている順番は，一定試合ごとに変更

UECda-2007ルール

- ローカルルールばかりでは試合にならないので、大会用に制定されたルール。

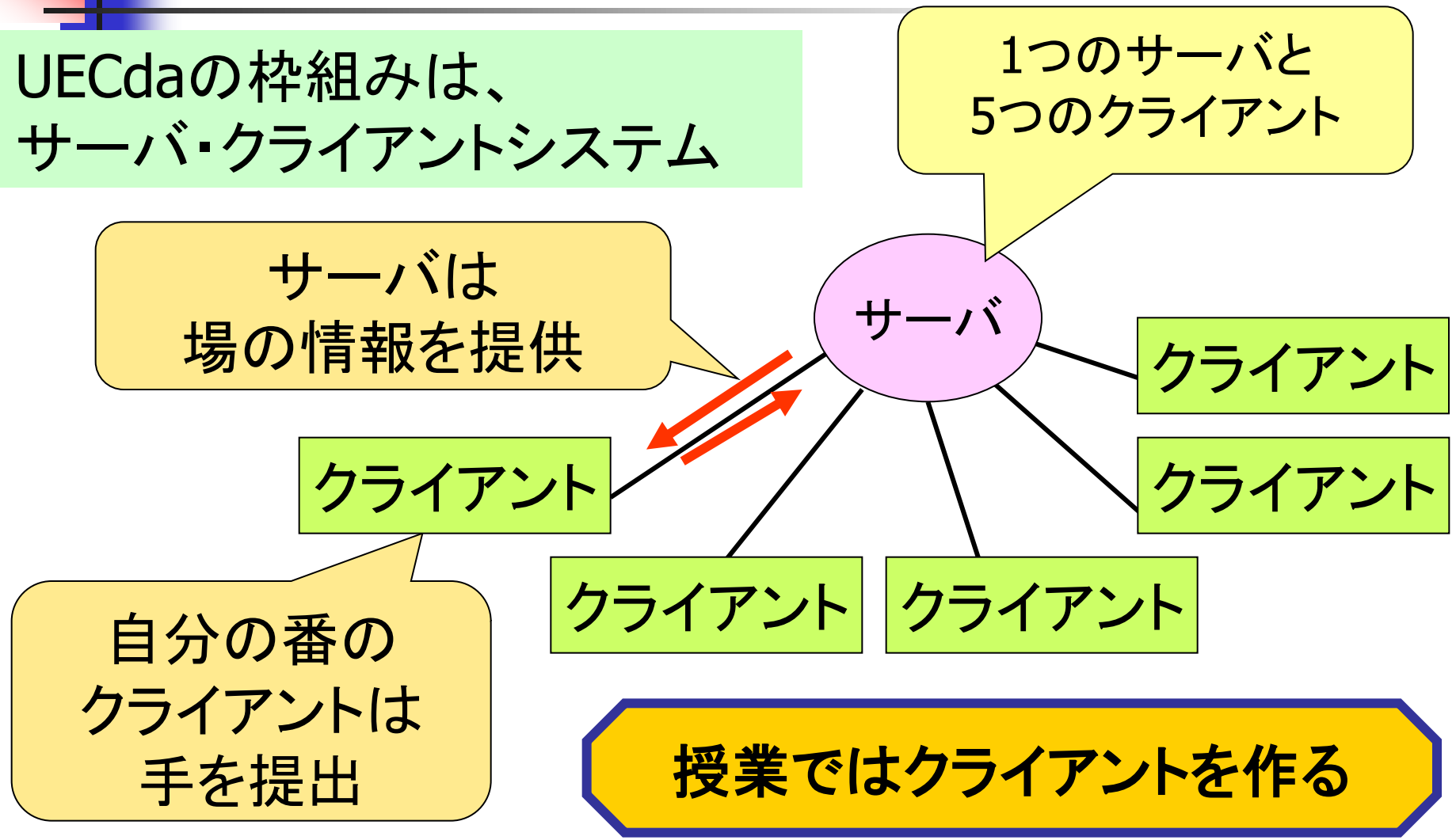
ルールの概要

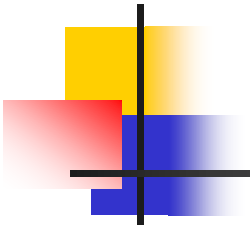
- 5人でおこなう
- 8切りあり
- 階段あり
- 11バックなし
- 席替えあり
- カード交換あり
- ジョーカーあり
- 大富豪は5点，富豪は4点，平民は3点，貧民は2点，大貧民は1点を獲得する．
- スペ3あり
- しばりあり

詳しくはWebサイトを見ましょう

システム構成

UECdaの枠組みは、
サーバ･クライアントシステム





コンピュータ大貧民の実行

ファイルの入手

- ターミナルを起動
- コンピュータ大貧民プログラム（アーカイブ）のコピー
  - cd ~
  - mkdir uecda
  - cp /usr/local/class/daihinmin/tndhm_devkit_c-20180826.tar.gz ~/uecda
- アーカイブの展開（解凍）
  - cd ~/uecda
  - tar xzvf tndhm_devkit_c-20180826.tar.gz

大貧民サーバのコンパイル

1. カレントディレクトリを変更する

```
cd  ~/uecda/tndhm_devkit_c-20180826/server
```

2. configureを実行する

```
./configure
```

3. make する（再コンパイル時は、直前に「make clean」を実行すること）

```
make
```

大貧民サーバの実行

4. カレントディレクトリを変更する

```
cd  ~/uecda/tndhm_devkit_c-20180826/server/src
```

5. 実行

```
./tndhms  -p ポート番号
```

ここで、ポート番号は以下のルールで指定
50000+入学年度(西暦)下1桁×1000
+学籍番号下3桁

6. 無事起動した場合、3つのウィンドウが開く。

標準クライアントのコンパイル

1. 大貧民サーバとは別のターミナルを起動
2. カレントディレクトリを変更する

```
cd  ~/uecda/tndhm_devkit_c-20180826/client
```

3. configureを実行する

```
./configure
```

4. make する

```
make
```

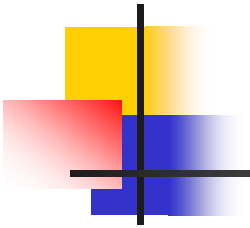
標準クライアントの実行

5. 実行する(標準クライアントの起動)

```
./client  -p ポート番号  &
```

ここで、ポート番号は、「大貧民サーバの実行」の手順5.で指定した番号を入力する

6. 対戦には5つのクライアントが必要なので、上記の手順5.を全部で5回行う。クライアントが5台起動すると、ゲームが開始される



教育用クライアントのソースコード解読

教育用クライアントのファイル入手

- 新しくターミナルを開く
- 教育用クライアントのアーカイブをコピーする

```
cp /usr/local/class/daihinmin/tndhmc-0.03.tar.gz  ~/uecda
```

- アーカイブを展開

```
cd  ~/uecda
tar xzvf tndhmc-0.03.tar.gz
```

- 各自で開発するクライアントは、~/uecda/tndhmc-0.03/src で作成する

教育用クライアントのコンパイル

1. カレントディレクトリを変更する

```
cd  ~/uecda/tndhmc-0.03
```

2. configureを実行する

```
./configure
```

3. make する（Cのソースファイルを修正し、再度コンパイルを行うときは、直前に「make clean」を実行する）

```
make
```

教育用クライアントの実行

4. カレントディレクトリを変更する

```
cd  ~/uecda/tndhmc-0.03/src
```

5. 実行する(教育用クライアントの起動)

```
./client  -p ポート番号 &
```

ここで、ポート番号は、「大貧民サーバの実行」の手順5.で指定した番号を入力する

6. 対戦には5つのクライアントが必要なので、上記の手順5.を全部で5回行う

教育用クライアントのソースの概要

- 教育用に、標準クライアントからサブセットを作成
  - 最低限の関数のみ実装
  - 大域変数を廃止
- ファイル構成
  - client.c　　：　全体の流れ
  - select_cards.c　：　提出カード選択
  - common.c　　：　共通関数
  - daihinmin.c　　：　基本関数
  - connection.c　　：　通信関係

教育用クライアントのソースの実装

- 詳細は、ソースを参照。
- コメント等は不完全
- 基本的な配列操作（common.c）
  - 指定より弱いカードの削除。指定したスートの削除。
  - and、or、copy、diff、not、count_cards
  - テーブルのコピー、テーブルの初期化
- 基本的なカード提出アルゴリズム (select_cards.c)
  - 通常時に単騎（１枚出し）でのみ提出
  - 革命時や縛り、ペアや階段では何もしない

カードの表現例1（手持ちの札）

- 詳しくは、UECdaのWebサイト参照

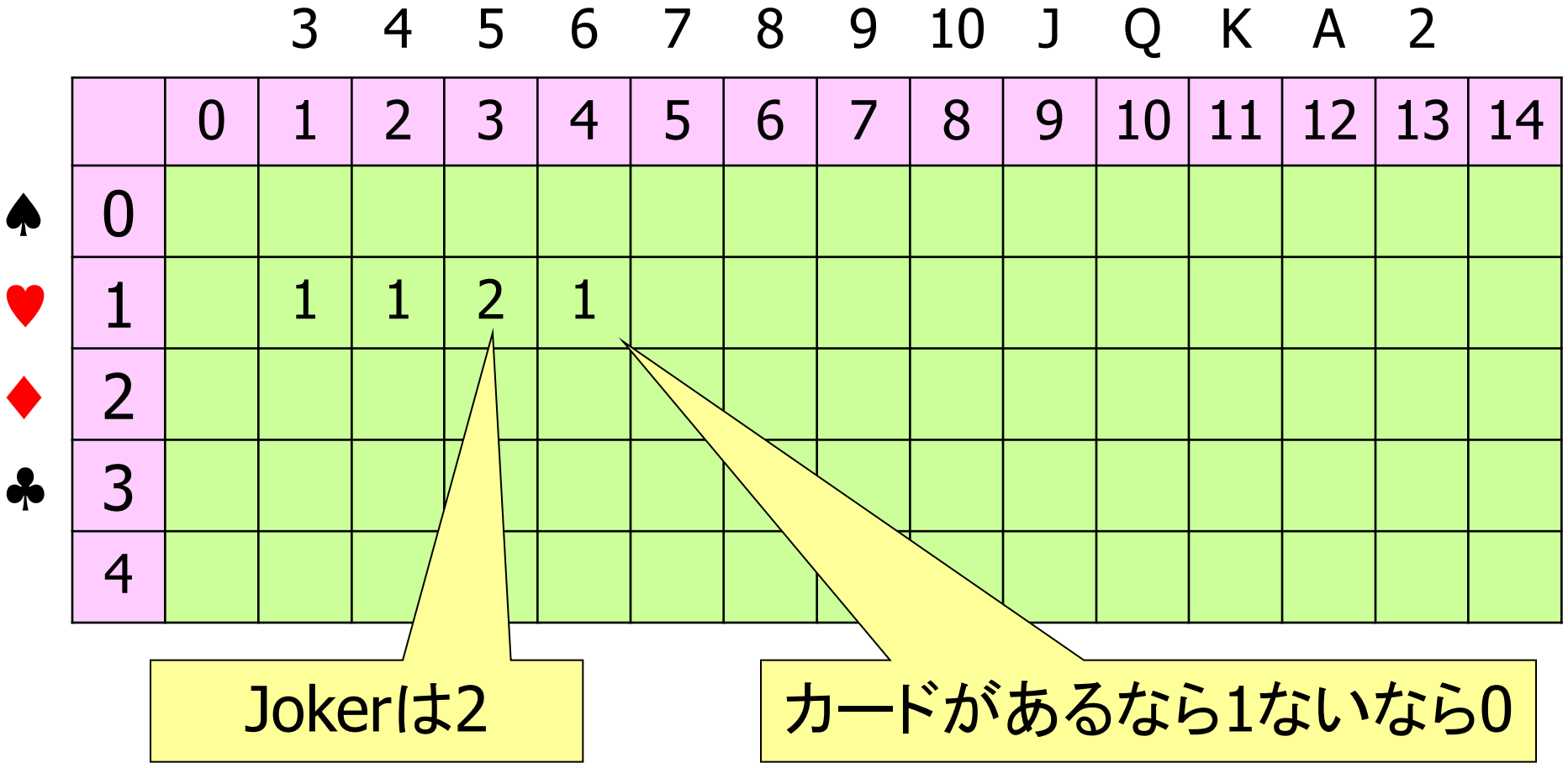
	3	4	5	6	7	8	9	10	J	Q	K	A	2
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
♠	0														
♥	1			1	1	1				1	1		1		
♦	2							1						1	
♣	3							1			1				
	4		2												

Jokerは2

カードがあるなら1ないなら0

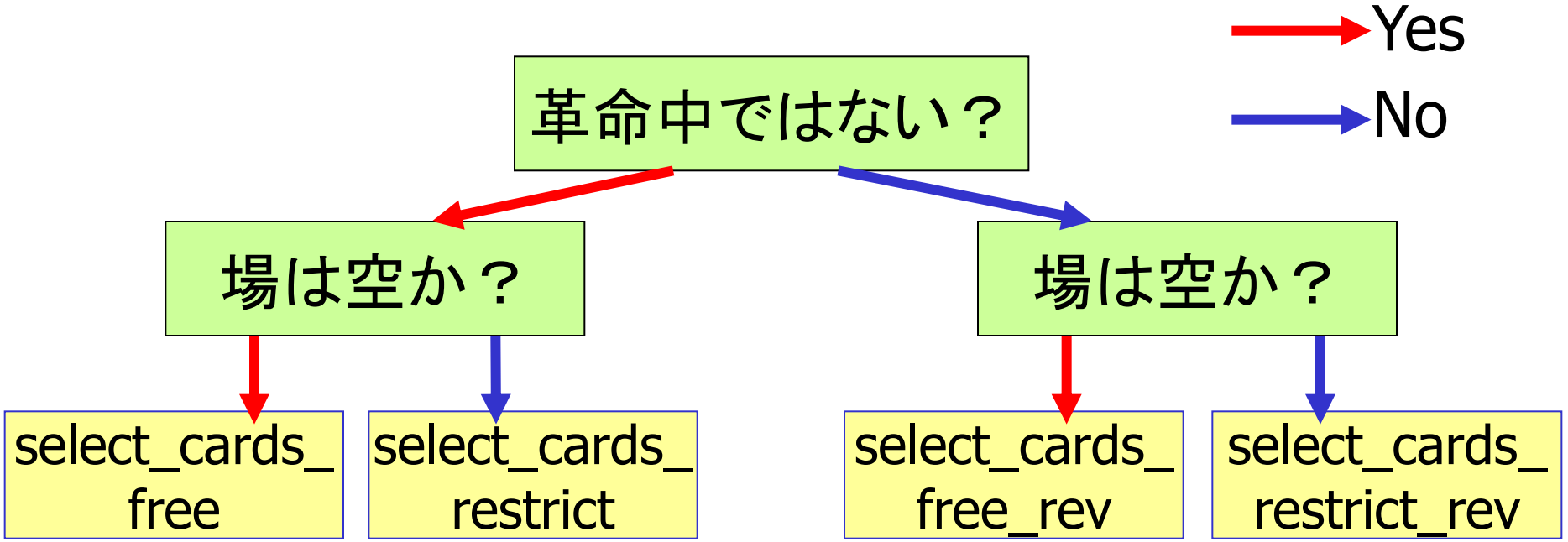
カードの表現例2（場札）

- 場に♥3、♥4、Joker、♥6 の階段が出た



フローチャート(全体)
関数 select_submit_cards

- 引数:
  - out_cards　　:提出するカードを設定する配列
  - my_cards　　:自分の手持ちのカード情報
  - *field_status:場の状態情報



関数 select_submit_cards

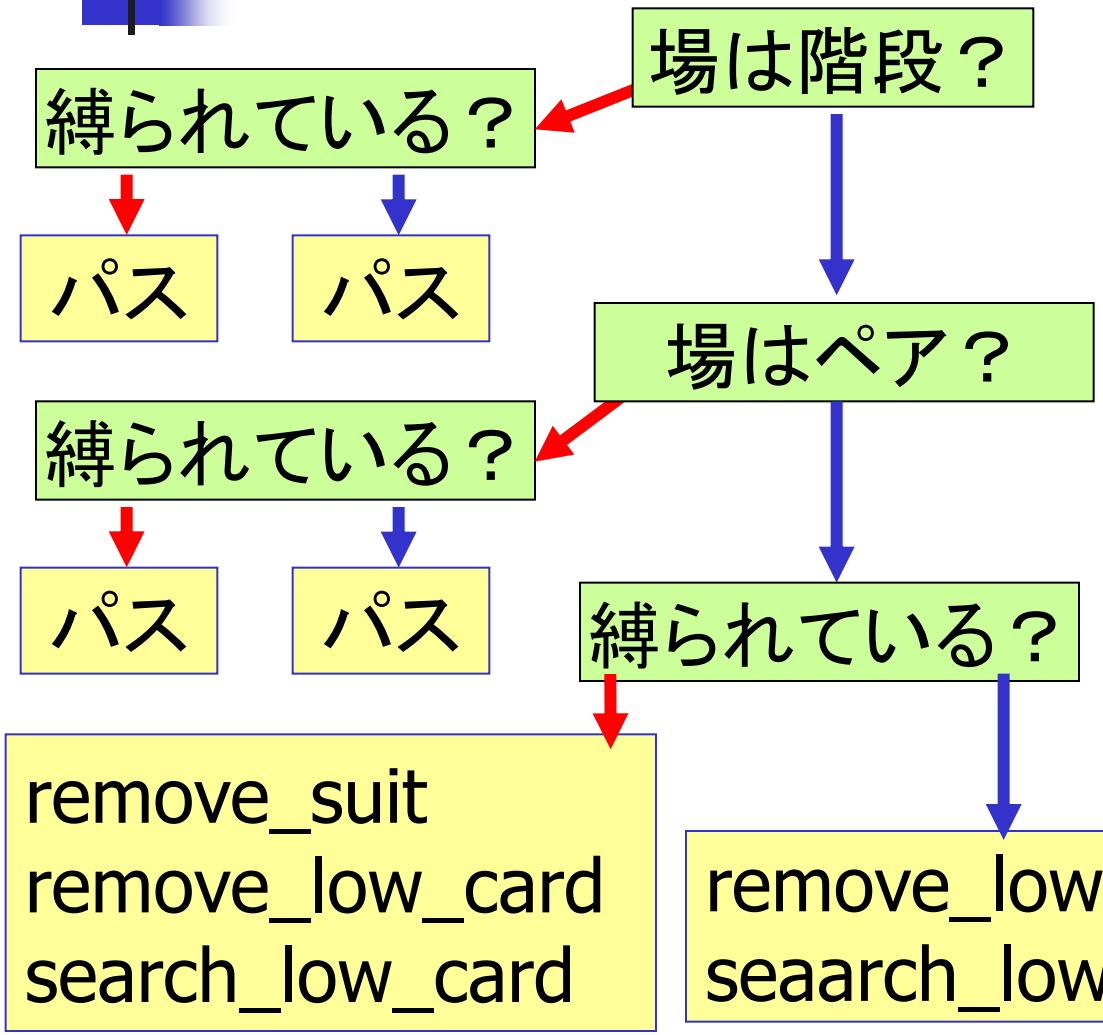
革命中でないか？

場は空か？

```
if(field_status->rev==0){
    if(field_status->is_no_card==1)
        select_cards_free(select_cards, my_cards, field_status);
    else
        select_cards_resrict(select_cards,my_cards, field_status);
else
    if(field_status->is_no_card==1){
        select_cards_free_rev(略);
    else
        select_cards_restrict_rev(略);
}
```

場は空か？

フローチャート （通常時、場札あり）
関数 select_cards_restrict



remove_suit
　指定スートの
　カード削除
remove_low_card
　指定値より弱い
　カード削除
search_low_card
　一番弱いカード
　探索

単騎のみ提出

フロー (他の関数)

- select_cards_free　　/* 通常時、場札なし */
  - search_low_solo　　/* 一番弱いカードを探索 */
- select_cards_free_rev　/* 革命時、場札なし */
  - パス
- select_cards_restrict_rev　/* 革命時、場札あり */
  - パス

remove_low_cardが行うこと(1)

- 指定値以下の弱い（あるいは強い）カードを削除
  void remove_low_card(int cards[8][15], int num, int rev)
- 呼出し例（場札が♠5のとき）：remove_low_card(cards, 3, 0)

		3	4	5	6	7	8	9	10	J	Q	K	A	2
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
♠	0														
♥	1		1	1	1				1	1		1			
♦	2						1							1	
♣	3						1			1					
	4	2													

ここの値を0に

remove_low_cardが行うこと(2)

```
void remove_low_card(int cards[8][15],int num,int rev){
  int i,j;

  if(rev==0){  /* 革命中でない時 */
    for(i=0;i<=num;i++)
      for(j=0;j<=3;j++)
        cards[j][i]=0;
  }else{       /* 革命中の時 */
    for(i=14;i>=num;i--)
      for(j=0;j<=3;j++)
        cards[j][i]=0;
  }
}
```

値

強いのを削除か
弱いのを削除か

構造体 state と変数 field_status

- 場の状況等を保存するための構造体と変数
- get_field_state_from_own_cardsと get_field_state_from_field_cardsで更新（client.c にすでに書いてある）

- 構造体のメンバ（すべて整数型）
  - ord, suit[5], quantity, is_sequence
  - is_rev, is_lock, is_no_card
  - player_quantity[5], player_rank[5], seat[5];
  - have_joker

場のカード情報

場の状況

自分がJoker持ちか

プレイヤの状況