【練習課題】ブラックジャックを作ろう！　―解答例―

■練習 1 の解答例 （復習⇒第 9 章）

```
struct tr{
    int suit;
    int value;
    int disp;
    char mark;
} card[52],deal[21],play[21],tmp;
```

カード情報を格納するという目標は、配列でも達成できます。でも、情報のコピーのしやすさなどから、今回は構造体を使ってみました。「配列で作ってみたいぞ」という方は、ぜひ配列でチャレンジしてみてください。

最低限、スーツ（スペード、ハートなどのマーク）とカードが持つ値を格納できればいいと思います。今回は、将来ほかのゲームを作るときにも使えるよう、4 つのメンバを作ってみました。

- int suit; —— スーツを数字（1…スペード、2…ハート、3…ダイヤ、4…クラブ）で入れてみます。
- int value; —— カードの値を入れます。「絵札は 10」ルールのゲームなどが多いため、実際の値を入れるのに使ってみます。
- int disp; —— カードの値を入れます。こちらは表示される名目上の値（11 とか 13 とか）にしてみます。
- char mark; —— スーツをマークで入れます。♠マークなどは、VisualC++では使いにくいので、今回はスペード（S）など、スーツの頭文字を入れてみます。

構造体型 struct tr から、実際に作った構造体は card、deal、play、tmp の 4 つです。

card は実際のカードに見立てて使います。ジョーカーは使わないので、52 個の要素を持つ配列にしました。

deal と play はそれぞれ、ディーラーとプレイヤーが受け取ったカードを格納します。ブラックジャックで 10 枚も 20 枚もカードを受け取ることはありませんが、要素数は 21 にしています。tmp はカードをシャッフルするときの一時記憶領域です。

■練習 2 の解答例 （復習⇒第 9 章）

たくさんの方法を考えることができますが、一例を挙げます。

```
for (int i=0; i<=12; i++) {
    card[i].suit=1;
    card[i].value=i+1;
    card[i].disp=i+1;
    card[i].mark='S';
}
```

```
for (int i=13; i<=25; i++) {
    card[i].suit=2;
    card[i].value=i-12;
    card[i].disp=i-12;
    card[i].mark='H';
}

for (int i=26; i<=38; i++) {
    card[i].suit=3;
    card[i].value=i-25;
    card[i].disp=i-25;
    card[i].mark='D';
}

for (int i=39; i<=51; i++) {
    card[i].suit=4;
    card[i].value=i-38;
    card[i].disp=i-38;
    card[i].mark='C';
}
```

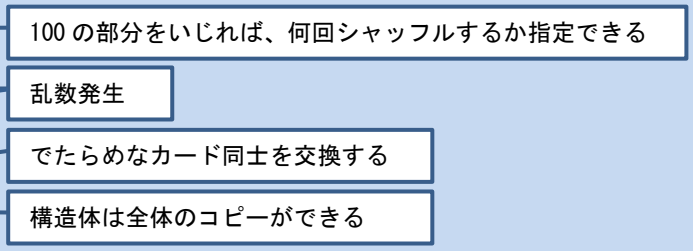
■練習 3 の解答例 （復習⇒第 5 章、第 9 章）

```
for (int i=1; i<=100; i++) {

    s1=rand()%52;
    s2=rand()%52;

    tmp=card[s1];
    card[s1]=card[s2];
    card[s2]=tmp;

}
```



■練習 4 の解答例 （復習⇒第 7 章）

```
void disp(int a,struct tr b[21]){

    for(int i=0;i<a;i++) printf("%c%d  ",b[i].mark,b[i].disp);
    printf("¥n");
}
```

1 枚目から a 枚目（プレイヤーが持っている手札の数）まで、プレイヤー（ディーラー）の手札情報を表示します。

■練習 5 の解答例 （復習⇒第 7 章、第 8 章）

```
void num(int *n){
```

```
    if (*n>10) *n=10;
    if (*n==1) *n=11;

    return;

}
```

ふつうに値渡しの関数でも作れますが、今回はポインタを使ってみました。そのため、戻り値を使っていません。

■練習 6 の解答例

```
    int cont=1;        ①
    int deal_s=0;      ②
    int play_s=0;      ③
    int c_d=0;         ④
    int c_p=0;         ⑤
    int c_t=0;         ⑥
    int playflag =1;   ⑦
    int dealflag =1;   ⑧
    int win=0;         ⑨
    int p_key=0;       ⑩
```

① ゲームを続けるかどうかを代入します。1 なら続行、0 なら終了です。
② ディーラーの札の合計点を代入します。
③ プレイヤーの札の合計点を代入します。
④ ディーラーが引いた札の数を代入します。
⑤ プレイヤーが引いた札の数を代入します。
⑥ カードの山(デッキ)が何枚目までめくられたかを代入します。
⑦ プレイヤーがまだ札を引くのかどうかを代入します。1 ならまだ引く、0 ならもう引きません。
⑧ ディーラーがまだ札を引くのかどうかを代入します。1 ならまだ引く、0 ならもう引きません
⑨ 勝ち負けの状態を代入します。0…未決着、1…ディーラーの勝ち、2…プレイヤーの勝ち、3…引き分け、です。
⑩ プレイヤーのキー入力を受け付けて、押されたキーを代入します。

削れる変数もありますし、もっと追加した方がよりプログラミングが楽になる変数もあるでしょう。ご自分なりの「必要な変数」を是非、考えてみてください。

■練習 7 の解答例 （復習⇒第 7 章、第 8 章）

```
int hantei(int p,int d){

    int win;

    if((p>d && p<22) || (d>21 && p<22)){
        printf("%s の勝ちです¥n",ps);
        win = 2;
```

```
    }
    if((d>p && d<22) || (p>21)){
        printf("%s の勝ちです¥n",ds);
        win = 1;
    }
    if(d==p || (d>21 && p>21)){
        printf("引き分けです¥n");
        win = 3;
    }
    return win;
}
```

■練習 8 の解答例 （復習⇒第 5 章）

```
for (int i=0;i<2;i++){ ―――― ①
    play[c_p]=card[c_t++]; ―――― ②
    num(&play[c_p].value); ―――― ③
    play_s = play_s + play[c_p++].value; ―――― ④

    deal[c_d]=card[c_t++]; ―――― ⑤
    num(&deal[c_d].value);
    deal_s = deal_s + deal[c_d++].value;
}
```

① 2 枚ずつなので、2 回ループさせています。上半分がプレーヤー、下半分がディーラーの処理です。

② デッキの c_t 枚目からカードを引いて、プレイヤーの手札の c_p 枚目としています。デッキからは c_t 枚目のカードを引いたので、引いた後に c_t に＋1 していることに注目してください。次に引く人は、次のカードをめくるわけです。後置演算なので、引いた後に＋1 しています。

③ カードは無事引けましたが、まだ油断できません。絵札（11、12、13）は 10 に数え直さなきゃいけませんし、A は 11 にしなきゃなりません。そこで、今引いた c_p 枚目の値をさっき作った num 関数に渡して、処理してもらいます。

④ 確定した c_p 枚目の値を、プレイヤーの合計点へ加算しています。それが終わったら c_p に＋1 して、次のカードをもらう準備をします。後置演算ですよね。

⑤ プレイヤーと処理の考え方はまったく一緒です。

■練習 9 の解答例 （復習⇒第 4 章）

```
        if (play_s >21) {
            for(int i=0; i<c_p; i++){
                if (play[i].value==11) {
                    play[i].value=1;
                    play_s = play_s-10;
                }
            }
        }
```

合計点が 21 を超えちゃった！

手持ちのカードに、

まだ 11 として数えている A がないかを探して、

あったら 1 として数えなおす

合計点からも 10 引いておく

これはプレイヤー側の例です。ディーラー側も同じ考え方で作れます。この辺は作り方のバリエーションがたくさん思いつくところなので、ほかの方法を考えることにも挑戦してみてください。

■練習 10 の解答例(完成版「makeBJ.cpp」)

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

struct tr{
int suit;
int value;
int disp;
char mark;
}card[52],deal[21],play[21],tmp;

char *ps="プレイヤー";
char *ds="ディーラー";

void disp(int a,struct tr b[21]){

    for(int i=0;i<a;i++) printf("%c%d  ",b[i].mark,b[i].disp);
    printf("¥n");
}

void num(int *n){

    if (*n>10) *n=10;
    if (*n==1) *n=11;

    return;

}

int hantei(int p,int d){

    int win;

    if((p>d && p<22) || (d>21 && p<22)){
        printf("%s の勝ちです¥n",ps);
        win = 2;
    }
    if((d>p && d<22) || (p>21)){
        printf("%s の勝ちです¥n",ds);
        win = 1;
    }
    if(d==p || (d>21 && p>21)){
        printf("引き分けです¥n");
        win = 3;
    }
    return win;
}
```

```
int main(void){

    srand(time(NULL));

    int s1=0,s2=0;
    int cont=1;

    do{

    int deal_s=0;
    int play_s=0;
    int c_d=0;
    int c_p=0;
    int c_t=0;
    int playflag =1;
    int dealflag =1;
    int win=0;
    int p_key=0;

for (int i=0; i<=12; i++) {
    card[i].suit=1;
    card[i].value=i+1;
    card[i].disp=i+1;
    card[i].mark='S';
}

for (int i=13; i<=25; i++) {
    card[i].suit=2;
    card[i].value=i-12;
    card[i].disp=i-12;
    card[i].mark='H';
}

for (int i=26; i<=38; i++) {
    card[i].suit=3;
    card[i].value=i-25;
    card[i].disp=i-25;
    card[i].mark='D';
}

for (int i=39; i<=51; i++) {
    card[i].suit=4;
    card[i].value=i-38;
    card[i].disp=i-38;
    card[i].mark='C';
}

for (int i=1; i<=100; i++) {

    s1=rand()%52;
    s2=rand()%52;

    tmp=card[s1];
    card[s1]=card[s2];
    card[s2]=tmp;

}
```

```
printf("カードを配ります¥n");

for (int i=0;i<2;i++){

    play[c_p]=card[c_t++];
    num(&play[c_p].value);
    play_s = play_s + play[c_p++].value;
    deal[c_d]=card[c_t++];
    num(&deal[c_d].value);
    deal_s = deal_s + deal[c_d++].value;
}

printf("%s の手札¥n",ps);
disp(c_p,play);

printf("%s の手札¥n",ds);
disp(c_d,deal);

while(1){

    if (playflag){
        printf("%s のターン¥n",ps);
        printf("  もう１枚カードを引きますか？  Yes=1, No=0¥n");
        scanf("%d",&p_key);

        if (!p_key) {
            playflag = 0;
        }
        else
        {
        play[c_p]=card[c_t++];
            num(&play[c_p].value);
            play_s = play_s + play[c_p++].value;

        if (play_s >21) {
            for(int i=0; i<c_p; i++){
                if (play[i].value==11) {
                    play[i].value=1;
                    play_s = play_s-10;
                }
            }
        }

        printf("%s はカードを引きました¥n",ps);
        printf("%s の手札¥n",ps);
        disp(c_p,play);
        }

        if(play_s > 21) {
            playflag=0;
            dealflag=0;
        }
    }

    if (playflag == 0 && dealflag ==0) win=hantei(play_s,deal_s);
    if (win) break;

    if (dealflag){
```

```
        printf("%s のターン¥n",ds);
        if (deal_s>=17) dealflag = 0;
        if (deal_s <17) {
            deal[c_d]=card[c_t++];
            num(&deal[c_d].value);
            deal_s = deal_s + deal[c_d++].value;

            if (deal_s >21) {
                for(int i=0; i<c_d; i++){
                    if (deal[i].value==11) {
                        deal[i].value=1;
                        deal_s = deal_s - 10;
                    }
                }
            }

            printf("%s はカードを引きました¥n",ds);
            printf("%s の手札¥n",ds);
            disp(c_d,deal);
        }

    if (playflag == 0 && dealflag ==0) win=hantei(play_s,deal_s);
    if (win) break;
    }
    }

    printf("  もう１回やりますか？  Yes=1, No=0¥n");
    scanf("%d",&p_key);
    if(!p_key)cont=0;

}while(cont);

}
```