コヨーテ(ボードゲーム)

コヨーテというボードゲームをPCソロプレイ用に改定したものです

あそびかた

自分以外のプレイヤーのカードを見て、その場のカードに書かれた数の合計を推理するゲームです。

プレイヤーは**自分のカードが見えない状態**でその場のカードの合計が少なくともいくつであるかを予想し宣言します。
次に続くプレイヤーも同様に、自分のカードが見えない状態で場のカードの合計を予想し宣言しますが、**前のプレイヤーが宣言した以上の数字**しか宣言することは出来ません。
もし前のプレイヤーが宣言した数がこの場のカードの合計より大きいと思った場合「**コヨーテ**」と宣言することができます。
「**コヨーテ**」と宣言した場合、場のカードをすべて開示し、**前のプレイヤーが宣言した数と場のカードの合計を比較**します。
もし前のプレイヤーが宣言した数のほうが大きければ「**コヨーテ**」は**成功**。場のカードより大きな数字を言ってしまった**前のプレイヤーのライフが一つ減ります**。
もし前のプレイヤーが宣言した数が場のカードの合計より小さければ「**コヨーテ**」は**失敗**。**「コヨーテ」を宣言したプレイヤーのライフが一つ減ります**。
これを繰り返し、ライフのなくなったプレイヤーが負けとなります。

詳細ルール

- 使用されたカードは捨て札となります。ただし**特殊カード シャッフル0**が出た際は次の場に行くときに捨て札と残った山札を混ぜ、プレイを再開します。
- **特殊カード x2**は**場の合計の数字を最後に2倍する**カードになります。
- **特殊カード MAX->0**は**場の一番大きな数字を0**にして計算します。

カード内訳

通常カード: (0/1/2/3/4/5/10/15/20/-5/-10)
特殊カード: (x2/MAX->0/シャッフル0)

実行方法

gccでコンパイル後、exeファイルを実行してください

```
gcc -o coyote ./coyote.c
./coyote.exe
```

もしくは

```
make coyote
./coyote
```

プログラムの仕様、および工夫点

オブジェクト形式マクロ

以下を使用

```
#define Players 4
#define CARDS 14
#define MAX_LENGTH 256
#define MAX_LINES 15
```

構造体:player

各プレイヤーの持つ情報を記録、保持するための構造体。
主にライフと持っているカードを保持

自作の関数

delay()関数

引数に単位がミリ秒となる数字を受けてその秒数処理を止める関数。
botの思考時間やプレイのしやすさを考慮し導入。
timeライブラリを使用。

culc_sum()関数

引数にカードのkeyの入った配列とその配列のサイズを受け取り、配列の合計値を計算する関数。
flagを複数使用することで複雑なx2やMAX->0のカードを含めた合計値が得られる。
配列を使用。

Player_Move()関数

引数に全プレイヤーの構造体と直前に宣言された数を受け付け、ゲームをプレイするプレイヤーの行動決定を出力する関数。
プレイヤーの入力に例外があっても処理できるように、以下のようなwhile文と標準ライブラリを用いた例外処理を実装している。

```
while(1){
    fflush(stdin); // 標準出力バッファをクリア

    printf("宣言する数字を入力してください:");
    fgets(input_str, sizeof(input_str), stdin);
```

```
        // 改行文字を取り除く
        size_t len = strlen(input_str);
        if (len > 0 && input_str[len - 1] == '\n') {
            input_str[len - 1] = '\0';
            len--;  // 改行文字を含まない長さに変更
        }

        char *endptr;
        input = strtol(input_str, &endptr, 10);

        if(len > 0 && input > prev){
            break;
        }
        else{
            printf("入力エラー 数値以外、もしくは前の宣言より小さい数字が入力されています。\n");
        }
    }
```

文字列、標準関数を使用。

Ai_Move()関数

botプレイヤーの行動決定のための関数。
基本的な行動としてはポインタを用いて自分の手札の予想を立て、それをもとにbotの行動決定を行うが、それに合わせて外部ファイルの読み込みを行い、botの思考を出力するためのメッセージテキストを用意している。
メッセージテキストは配列で管理し、状況に合わせた数字を指定することでその数字に合わせたメッセージテキストが表示されるようになっている。
ポインタ、配列を使用。

card_shaffule()関数

引数でカードのリストとそのサイズを受け付け、標準ライブラリのrand関数を用いてカードのリストをシャッフルする関数。
標準関数、配列を使用。

generate_list()関数

引数に空のカードのリストを受け付け、そのリストの中に、プログラム序盤で定義してあるカードの種類を保存する配列key[]とその種類ごとの枚数を管理する配列value[]からシャッフルされていないカードのkeyを山札として代入する関数。
for文中の変数を効果的に使用し、配列のインデックスを呼び出しやすく実装している。
配列を使用。

play_main()関数

プレイのための総合的な処理を行う関数。
flavor_text.txtからbotのメッセージを配列に代入し、山札を生成し、スタートプレイヤーをrand関数によって決めたのち、プレイヤーのライフが0になるまでwhile文が繰り返される。
このwhile文ではプレイヤーの手札の決定を行い、場のカードの合計を計算し、保存している。

続いていずれかのプレイヤーがコヨーテを行うまでのwhile文が繰り返される。
この中ではbotプレイヤー、あるいはプレイヤーがした行動決定を受け付け、その行動に従って条件分岐を行っている。　コヨーテが宣言されれば場のカードの合計と直前の宣言を比較してコヨーテが成功したかどうかを判定、そののちに適当なプレイヤーのライフを減らし、手札の決定に戻る。
この際、シャッフル0カードが手札に出ていた場合はflagで管理し、山札のシャッフルを行っている。
いずれかのプレイヤーのライフが0になればどのプレイヤーが勝利したかを示してゲーム終了となる。
ファイル入出力、文字列、配列、標準関数を使用。

既知の不具合、未完成部分

- botの難易度調整やプレイのアルゴリズムをもっと複雑にしたい。
- まれにbotの宣言が直前の宣言より小さく、もしくは直前の宣言と全く同じになる場合がある。原因不明

参考文献

1. 「新明解C言語入門編第2版」柴田望洋、SB Creative

Author

- 21T2166D 渡辺大樹