プログラミング言語実験・C言語 第4回課題レポート

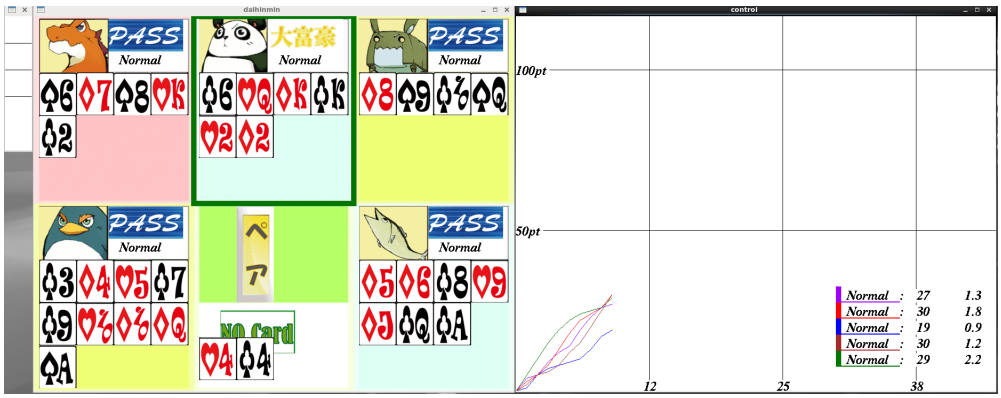
1510151 柳 裕太

2017年5月8日

1　課題7

1.1　スクリーンショット

図1　実際にペアが出された時のスクリーンショット



1.2　配列に対する操作

1.2.1　make_info_table()

第一引数 info_cards 配列を初期化したのち、各数字におけるカードの枚数 (第二引数 my_cards 配列 1-3 行目各列の合計) を、info_table 配列 4 行目に記録する。

1.2.2　search_low_pair()

まず提出するカードを示す第一引数 dst_cards 配列を初期化する。

その後、第二引数 info_table 配列 4 行目を for 文で要素が 2 以上 (=ペアが存在する) の部分が存在するまでループを回し、もしペアが存在するならば、該当する列の第三引数 my_cards 配列の各行要素を dst_cards 配列に代入し、1 を返すことで処理を終了する。

もしペアが存在しないならば、0 を返すことで処理を終了する。

1.2.3　select_cards_free()

まず、info_table 配列を定義した後、make_info_table 関数を呼び出し、info_table 配列と第二引数 my_cards 配列を渡すことで、ペアの情報を取得する。

その後、count_cards() 関数を呼び出し、第一引数 select_cards 配列を渡すことで、出すカードが決定されている状態であるかを調べる。もし出すカードが決まっていないのならば、search_low_pair() 関数を呼び出し、出せられるペアの中で最弱のものを提出カードに指定する。ここで search_low_pair() 関数にて出せられるペアが存在しないのならば、search_low_cards() 関数を呼び出し単騎カードを提出カードに指定する。

1.3 機能実装の決め手

1.3.1 make_info_table()

for 文により、my_cards 配列 0 3 行目の任意の列の合計を info_table 配列 4 行目の同じ列番号に代入することで、ペア情報の作成が可能となっている。

1.3.2 search_low_pair()

まず、for 文で info_table 配列 4 行目にてペアを出せられる数字があるかを 1 行目から調べ、i 列に存在した時点で break している。これにより、その直後に if(i¡=13) とすることで、ペアが存在しない = i が 14 (= ペアが存在すれば 14 に達する前に break されている) ケースは、ペアを指定するシーケンスをパスすることが可能となっている。

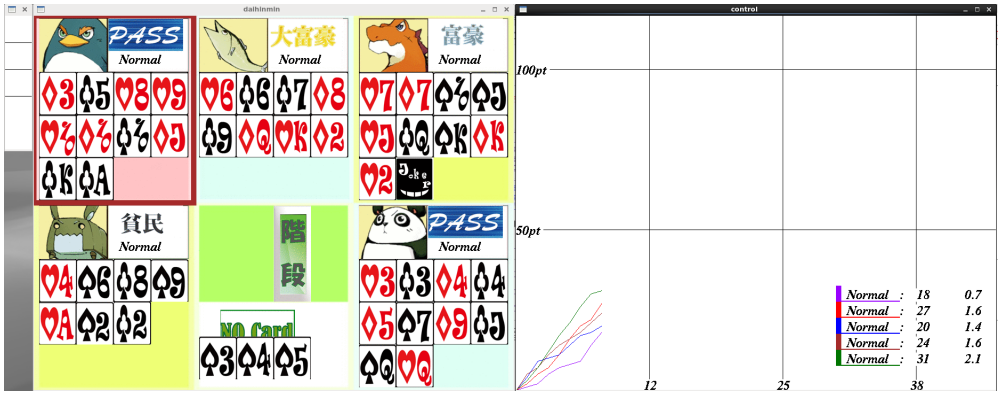
1.3.3 select_cards_free()

提出カードを指定するときに、search_low_pair() 関数或いは search_low_card() 関数を呼び出す前に、if(count_cards(select_cards)==0) を活用している。これにより、先に提出カードにペアが指定された後に、単騎提出カードに上書き指定されるのを防ぐことが可能となっている。

2 課題 8

2.1 スクリーンショット

図 2　実際に階段が出された時のスクリーンショット



2.2 配列に対する操作

2.2.1 make_info_table()

課題 7 に加え、for 文内に更に for 文をネストし、my_cards 配列各行における要素において、連続して次の列が 1 である (3 つ 1 が続く=階段を出せられる) 回数を数え、その結果を info_table の同一行・列に代入する。

2.2.2 search_low_sequence()

まず、提出するカードを指定する第一引数 dst_cards 配列を初期化する。その後、for 文を 2 重にネストし、第二引数 info_table 配列の各要素を調べ、3 以上=同一スートで連続して 3 枚以上のカードが存在する (=階段が出せられる) ならば、dst_cards 配列の該当する階段のエリアの部分に 1 を代入し、1 を返す。もし階段を出せられないならば、0 を返し処理を終了する。

2.2.3 select_cards_free()

課題 7 の記述において、if 文の後に search_low_pair() 関数を呼び出す部分の前に、また同じ if 文で提出カードが決定しているかを調べ、決まっていないならば search_low_sequence() 関数を呼び出し、階段が出せられるかどうかを調べている。

2.3 機能実装の決め手

2.3.1 make_info_table()

for 文を 2 重にネストし、更にその中に while 文をネストすることで、すべての要素を網羅的に調べることが可能となった。

2.3.2 search_low_sequence()

for 文を 2 重にネストする際、列の上限を 11 にした。これは、12 列目以降では、階段の成立はあり得ない (数字は 13 種類) ためである。

2.3.3 select_cards_free()

課題 7 で作成した「ペアがあるかをまず調べ、ないのならば単騎カードを出す」の流れの頭に、「階段があるかを調べる」ことで、提出カードのパターンの優先順位をより実戦に近づけることが可能となった。