1　課題 5

1.1　サーバーのみ実行時

1.2　クライアント実行時

2　課題 6

2.1　common.c

2.1.1　copy_table()

■何の処理？

第二引数の配列の任意の値を第一引数の配列へコピーする処理。

■配列への処理内容

第二引数の配列の x 行 y 列の値を第一引数の同じ場所へ代入する

■機能実現への決め手

for 文を 2 重にネストし、dst_cards[i][j]=src_cards[i][j] とすることで実現が可能となった。

2.1.2　clear_table()

■何の処理？

引数として受け取ったカードテーブルを初期化する。

■配列への処理内容

配列のすべての値を 0 にする。

■機能実現への決め手

for 文を 2 重にネストし、cards[i][j]=0 とすることで実現が可能となった。

2.1.3　copy_cards()

■何の処理？

第二引数の配列のカード情報を第一引数の配列へコピーする処理。

■配列への処理内容

第二引数の配列の x 行 y 列 (x ≦ 5) の値を第一引数の同じ場所へ代入する

■機能実現への決め手

基本的には copy_table() と処理内容は同じである。1 点のみ違いがあり、for 文でネストする際に i の制限を 8 ではなく 5 とすることで、カード情報のみ指定して初期化することが可能となった。

2.1.4　clear_cards()

■何の処理？

引数で受け取ったカードテーブルのカード情報の部分のみ初期化する。

■配列への処理内容

カードテーブルの第 5 行までに限り (列は任意)、配列の値を 0 にする。

■機能実現への決め手

基本的には clear_table() と処理内容は同じである。1 点のみ違いがあり、for 文でネストする際に i の制限を 8 ではなく 5 とすることで、カード情報のみ指定して初期化することが可能となった。

2.1.5 diff_cards()

■何の処理？

ゲーム開始前の貧民・大貧民→大富豪・富豪のカード譲渡において、引数で指定された (search_low_card() にて指定された貧民・大貧民持ち札で最も強い 1-2 枚の) カードを貧民・大貧民の持ち札から削除する処理。

■配列への処理内容

第一引数の自分の持ち札を表す cards1 と、事前に search_low_card() にて譲渡=削除すべきカードを cards2[x][y]=1 とされた配列を第二引数として受け取った。cards1 配列において、cards2[x][y] にて 1 とされた場所を 0 にする、つまり cards1[x][y] に 0 を代入すること。

■機能実現への決め手

for 文を 2 重にネストすることで 2 次元配列 cards1[][] 全体を探索し、if 文として 1 と指定された場所に限り、cards1 の同一の行・列部分に 0 を代入することで実現可能となった。

2.1.6 or_cards()

■何の処理？

ゲーム開始前の大富豪・富豪→貧民・大貧民のカード譲渡において、引数で指定された (search_low_card() にて指定された貧民・大貧民持ち札で最も強い 1-2 枚の) カードを富豪・大富豪の持ち札へ追加する処理。

■配列への処理内容

第一引数の自分の持ち札を表す cards1 と、事前に search_low_card() にて譲渡=削除すべきカードを cards2[x][y]=1 とされた配列を第二引数として受け取った。cards1 配列において、cards2[x][y] にて 1 とされた場所を 1 にする、つまり cards1[x][y] に 1 を代入すること。

■機能実現への決め手

for 文を 2 重にネストすることで 2 次元配列 cards1[][] 全体を探索し、if 文として 1 以上に指定された場所に限り、cards1 の同一の行・列部分に 1 を代入することで実現可能となった。

2.1.7 and_cards()

■何の処理？

第一引数の cards1 配列において、cards2 においても 1 として指定された場所以外をすべて 0 にする処理。

■配列への処理内容

cards1[][] において、cards1 と cards2 が同じ場所の配列の値がともに 1 である場合に限り、cards1 の該当値を 1 のままとする。両方共 1 でないのならば、cards1 の該当値は 0 が代入される。

■機能実現への決め手

for 文を 2 重にネストすることで 2 次元配列 cards1[][] 全体を探索し、if 文として cards1[i][j] と cards2[i][j] がともに 1 である場合を&&で指定し、その場合は cards[j][i] に 1 を代入し、それ以外の場合は else として cards[j][i] に 0 を代入することで、実現可能となった。

2.1.8 not_cards()

■何の処理？

カードの有無を配列の値を 0 と 1 とで入れ替えることで反転する処理。

■配列への処理内容

配列の任意の場所の値を 0 ⇄ 1 とで反転している。

■機能実現への決め手

for 文を 2 重にネストすることで 2 次元配列 cards[][] 全体を探索し、cards[j][i]=1 ならば 0 を、0 ならば 1 を cards[j][i] に代入することで、実現可能となった。

2.1.9 get_field_state_from_field_cards()

■何の処理？

場に出ているカードの情報から、現在の場の状況を把握する処理。

■配列への処理内容

初期段階判定　cards[i][j] が 1 である場所を捜し、存在しているカードのスートを field_status の suit[i] に記録する。すべてのスートを調べた (i=4) ならば j をインクリメントし、i を 0 に戻す。

階段判定　cards[i][j] において、cards[i][J+1] も場に出ている=1 である場合、階段が発動していると判定し、field_status の is_sequence に 1 を代入する。

枚数組判定　階段が不成立の場合 (field_status=0)、まず同一数値における別スートのカードの有無を調べ、存在するならば枚数をカウントする count をインクリメントし、suit[i] に 1 を代入する。存在しなければ suit[i] を代入する。

革命有無判定　階段が不成立で j が 0 または 14 のとき、革命が発動中ならば、field_status の order に 14 を代入し、力関係が逆転していることを伝える。革命中でないのならば、order には 0 が代入される。j が 0 でも 14 でもない時は、j の値が order に代入される。

階段処理判定　階段が成立中の場合、出されたカードの最大値 (j)+1 セットの枚数 (count) が 15 を超えず、かつ cards[i][j+count] が 0 でない (場に出ている状態である) 間、while として count をインクリメントする。その後、革命中ならば field_status の order には j+count-1 を代入し、革命中でなければ j をそのまま代入する。最後に、field_status の suit[i] に 1 を代入する。

場カード有無　上述のコードによって判定された count を field_status の quantity に代入し、更に quantity の数値が 0 でないのならば、field_status の is_no_card に 0 を代入する。

■機能実現への決め手

初期段階判定　while で j を先に固定し、その後 i を 0 から 4 までインクリメントしている。スートよりも数字の値の方が重要性が高いため、先に固定することでカード位置の正確な探索が可能になった。
階段判定　もし場に同じスートで数字が隣接するカードも出ている場合を if-else で判別することで可能となった。
枚数組判定　j の値は変えずに、i のみ 0 から 4 までインクリメントして cards[i][j] の値を調べることにより、同一数値で何枚カードが存在するか調べることが可能になった。
革命有無判定　field_status に is_rev という革命有無を記す変数が搭載され、その値の有無によって if-else を分けることにより、可能なものとなった。
階段処理判定　先にスートを固定し、隣接するカードがなくなるまで while 文を回すことにより、セットで出された階段の枚数の判定が可能になった。また、革命の有無によって次に出すべき階段のセットの判定を変えることにより、正確に次に出すべきカードを判定することが可能になった。
場カード有無　count の値を quantity に移植、さらに quantity が 0 を超えることを if 文でキャッチしその場合 is_no_card に 0 を代入することで新場なのか否かを判別できるようになった。

2.1.10　get_field_state_from_own_cards()

■何の処理？

■配列への処理内容

■機能実現への決め手

2.1.11　remove_low_card()

■何の処理？

■配列への処理内容

■機能実現への決め手

2.1.12　remove_suit()

■何の処理？

■配列への処理内容

■機能実現への決め手

2.1.13 count_cards()

■何の処理？

■配列への処理内容

■機能実現への決め手

2.2 select_cards.c

2.2.1 select_change_cards()

■何の処理？

■配列への処理内容

■機能実現への決め手

2.2.2 select_submit_cards()

■何の処理？

■配列への処理内容

■機能実現への決め手

2.2.3 select_cards_restrict()

■何の処理？

■配列への処理内容

■機能実現への決め手

2.2.4 select_cards_free()

■何の処理？

■配列への処理内容

■機能実現への決め手