MIDTERM PROJECT

POKER GAMES

Ngô Huy Anh - 19127095
Phùng Anh Khoa - 19127449
27/04/2020

I. Work Assignment

Timeline

1. **Deadline :** 28/04/2020
2. **Giai đoạn**
    1) Hiểu cách chơi Poker + Lên ý tưởng làm : 16/04/2020
    2) Code : 17/04/2020 -> 23/04/2020
    3) Ghép code – Thống nhất ý tưởng + Test : 23, 24, 25/04/2020
    4) Làm Menu : 26/04/2020
    5) Viết Report : 27/04/2020
    6) Hoàn chỉnh + Test + Demo GCC : 28/04/2020
    7) Nộp bài : 29/04/2020

Work Assignment

II. Report

Shuffle Cards

1. **Void shuffleCards(int deck[][])**
    - **Nhiệm vụ** : Tạo ra một bộ bài 52 lá bằng cách tạo ra 1 ma trận 4 * 13 chứa thứ tự của lá bài trong bộ bài.
    - **Vấn đề :** Làm sao để tạo ra được ma trận có 52 phần tử phân biệt ? Khi mà cách dùng srand() & rand() vẫn xảy ra trường hợp tạo ra những phần tử trùng lắp nhau
    - **Ý tưởng :**
        1) Tạo 1 mảng số tự nhiên có vị trí từ 1 đến 52 tượng trưng cho 52 lá bài
        2) Lần lượt phát sinh ngẫu nhiên 1 số k – Lấy phần tử ở vị trí k trong mảng – tượng trưng cho lá bài số k và lần lượt đưa

vào ma trận. Sau **đó xóa lá bài k trong mảng, dời những lá bài ở phía sau k lên và giảm độ dài mảng đi 1 đơn vị.**

⇨ k = rand() % n + 1 . Do k chỉ được tạo trong khoảng từ 0 đến n, nên khi xóa 1 phần tử và giảm n đi 1 đơn vị thì phần tử đó **sẽ không xuất hiện lại nữa !**

- **Hàm bổ trợ :**
  1) srand(), rand()
  2) Void deleteElement(int arr[], int position, int& n)
  3) Void shuffle(int card[], int n, int& r) (**Giải thích ở câu I.2)**

2. **Void printCardsShuffling(int deck[][], const char* suits[], const char* ranks[])**
   - **Nhiệm vụ** : Xuất ra màn hình vị trí 52 lá bài đã được xáo
   - **Vấn đề :** Sử dụng **const** char* suits[], **const** char* ranks[] thay cho char* suits[] và char* ranks[]
   - **Ý tưởng :**
     1) Tạo ra ma trận matrix **2 * 53** để chứa SUITS và RANKS của lá bài – Bởi vì ma trận deck có 52 lá bài, không có lá số 0 -> Vị trí cột 0 của matrix sẽ không được sử dụng.
   - **Hàm bổ trợ :**
     1) void CopyPositionCards(int deck[SUITS][RANKS], int matrix[2][53])

One player

1. **int** dealingForHand(int deck[SUITS][FACES])**
   - **Nhiệm vụ** : Phát bài cho 1 người chơi
   - **Vấn đề :** Không có
   - **Ý tưởng :**
     1) Cấp phát bộ nhớ cho ma trận result 5 * 2
     2) Tạo 1 biến number để kiểm soát số lá bài. Sau đó chạy vòng lặp Search từng lá bài rồi đưa vào ma trận result. Khi number == 5 thì trả về ma trận result
   - **Hàm bổ trợ :** Không

2. **Void printHand(int** hand, char* suits[], char* faces[])**
   - **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
   - **Vấn đề :** Không có
   - **Ý tưởng :**
     1) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài
   - **Hàm bổ trợ :** Không có

3. **int isFourOfAKind(int** hand)**
   - **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
   - **Vấn đề :** Không có
   - **Ý tưởng :**
   - **Hàm bổ trợ :** Không có

4. **int isFullHouse(int** hand)**
   - **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
   - **Vấn đề :** Không có
   - **Ý tưởng :**
     2) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài
   - **Hàm bổ trợ :** Không có

5. **int isFlush(int** hand)**
   - **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
   - **Vấn đề :** Không có
   - **Ý tưởng :**
     3) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài
   - **Hàm bổ trợ :** Không có

6. **Void printHand(int** hand, char* suits[], char* faces[])**
   - **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
   - **Vấn đề :** Không có
   - **Ý tưởng :**

4) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài

- **Hàm bổ trợ :** Không có

7. **Void printHand(int** hand, char* suits[], char* faces[])**
   - **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
   - **Vấn đề :** Không có
   - **Ý tưởng :**
     5) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài
   - **Hàm bổ trợ :** Không có

8. **Void printHand(int** hand, char* suits[], char* faces[])**
   - **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
   - **Vấn đề :** Không có
   - **Ý tưởng :**
     6) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài
   - **Hàm bổ trợ :** Không có

9. **Void printHand(int** hand, char* suits[], char* faces[])**
   - **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
   - **Vấn đề :** Không có
   - **Ý tưởng :**
     7) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài
   - **Hàm bổ trợ :** Không có

10. **Void printHand(int** hand, char* suits[], char* faces[])**
    - **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
    - **Vấn đề :** Không có
    - **Ý tưởng :**
      8) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài

- **Hàm bổ trợ :** Không có

11. Void printHand(int hand, char* suits[], char* faces[])**

- **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
- **Vấn đề :** Không có
- **Ý tưởng :**
    9) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài
- **Hàm bổ trợ :** Không có

Two player

1. Void printHand(int hand, char* suits[], char* faces[])**

- **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
- **Vấn đề :** Không có
- **Ý tưởng :**
    10) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài
- **Hàm bổ trợ :** Không có

2. Void printHand(int hand, char* suits[], char* faces[])**

- **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
- **Vấn đề :** Không có
- **Ý tưởng :**
    11) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài
- **Hàm bổ trợ :** Không có

3. Void printHand(int hand, char* suits[], char* faces[])**

- **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
- **Vấn đề :** Không có
- **Ý tưởng :**
    12) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài

- **Hàm bổ trợ :** Không có

4. **Void printHand(int** hand, char* suits[], char* faces[])**
   - **Nhiệm vụ** : Xuất ra 5 lá bài của người chơi
   - **Vấn đề :** Không có
   - **Ý tưởng :**
     13) Chạy vòng lặp rồi lần lượt xuất ra SUITS và RANKS của từng lá bài
   - **Hàm bổ trợ :** Không có

Dealer Side

***Ý tưởng:**

-Đầu tiên ta cho nhập n số người chơi người chia bài sẽ nằm ở vị trí n+1 vì theo luật bài người chia bài luôn được phát lá cuối cùng, sau đó ta nhập số lượt chơi s.

-Ở mỗi lượt ta sẽ xào lại bộ bài và phát lại bài cho từng người chơi. Rồi show bài của người chơi và người phát.

-Sau khi nhận bài người phát bài có thể đổi không, một, hai, ba lá trong bộ bài với 5 lá trên tay, có thể chọn lấy và thay thế bài ngẫu nhiên hoặc tự thay đổi.

- Sau khi người phát bài đổi bài xong thì ta sẽ show top người chơi cảu lượt đó. Rồi ta cộng điểm vào 1 mảng theo thứ tự (top i: + (n-i) điểm)

-Sau khi hết lượt cuối thì ta show điểm tổng sau s lượt của các người chơi và công bố người chiến thắng (người có điểm cao nhất)

-nhiệm vụ

-vấn đề

-ý tưởng

***Các Hàm:**

int*** dealingForHandsAndDealer(int deck[SUITS][RANKS], int n);

int* rankingHandsAndDealer(int*** hands, int n);

int* evaluatehandsAndDealer(int* ranked_hands, int s);

void congratulationDealer(int* ranked, int n);

void pointDealer(int* ranked, int n);

void sumDealer(int* ranked, int* rankedLast, int n);

Đều sử dụng ý tưởng từ các hàm trước đó nhưng thay vì chỉ phát, tính điểm, cộng điểm cho n người chơi thì các hàm này phát, tính điểm, cộng điểm cho n người chơi và người phát bài

void shuffle(int card[], int n, int& r);

-nhiệm vụ: tráo bộ bài sau mỗi lượt chơi.

-vấn đề: nếu ta sử dụng hàm srand(time) thì số sẽ luôn giống nhau sau các lượt tráo bài.

-ý tưởng: ta thêm 1 biến r để thay đổi giá trị mỗi lần sử dụng hàm shuffle nó sẽ làm thay đổi số và tránh trường hợp tráo bài giống nhau

int** drawCard(int deck[SUITS][RANKS], int** hand, int n);

-Nhiệm vụ: hỏi người phát có muốn đổi bài hay không, nếu đổi thì hỏi họ muốn đổi một, hai hay ba lá. Muốn đổi ngẫu nhiên hay tự đổi.

-ý tưởng: làm 1 mảng handafterdraw nhận 5 lá của người phát, 1 mảng x nhận số muốn thay đổi. ta thay đổi lá trong mảng handafterdraw dựa vào mảng x. sau khi đổi bài xong ta cho người phát nhận lại 5 lá.