Computer Program Design 2015 Fall

Final Assignment - Simple Poker game

201122037 김태환

# Program source list

소스 코드 별 자세한 기능은 아래에 설명되어 있습니다.

* main.c // main 함수가 있어서 실제로 컴퓨터와 포커를 하는 기능이 구현되어 있음
* card.c // 포커를 하기 위해 사용되는 다양한 function이 구현되어 있음
* card.h // 카드 족보나 패를 정의하는 enum 상수, player, card 구조체, 함수 정의
* Makefile // 프로그램 빌드를 위한 Makefile

# 실행 방법

1. Codeblocks -> Press F9 or Build -> Build and run click
2. bash shell (UNIX, Mac, Linux) -> 아래에 설명
3. Windows terminal -> 프로젝트 폴더로 가면 Debug\ 디렉토리에 실행 파일이 존재함.

## Terminal 실행법

tkim@tkim-laptop:~/computer\_design\_final$ make

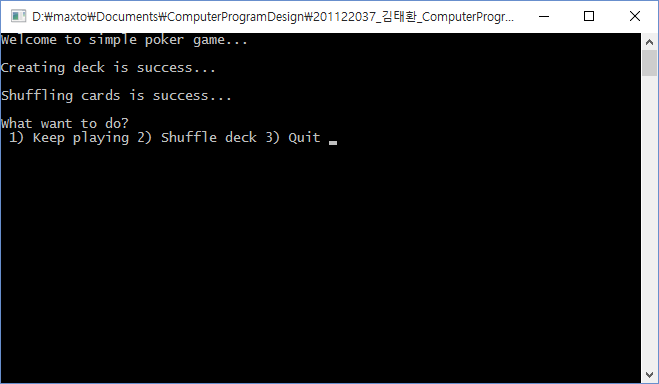
gcc -c main.c

gcc -o main main.o card.o

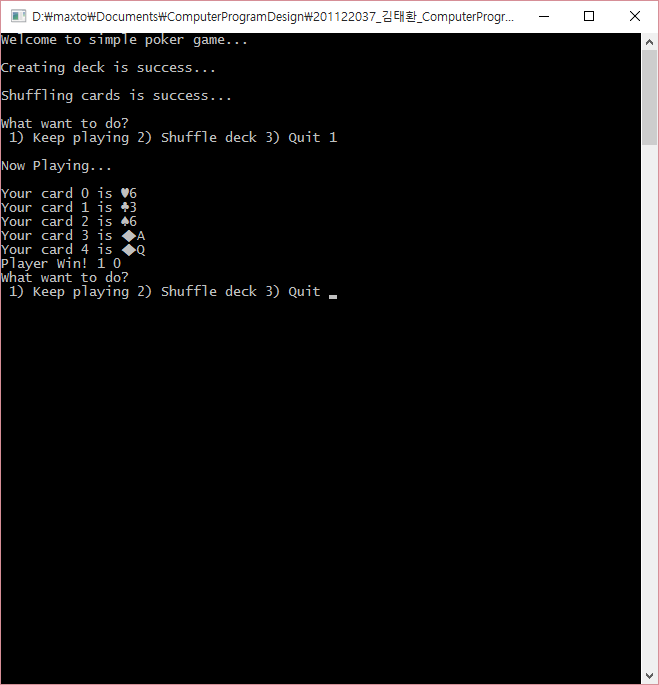
tkim@tkim-laptop:~/computer\_design\_final$ ./main

# 결과 화면

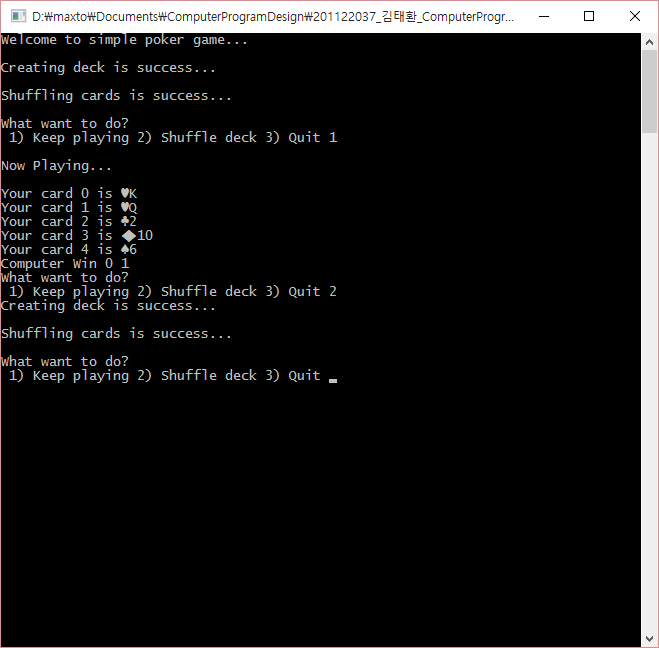
실행 시



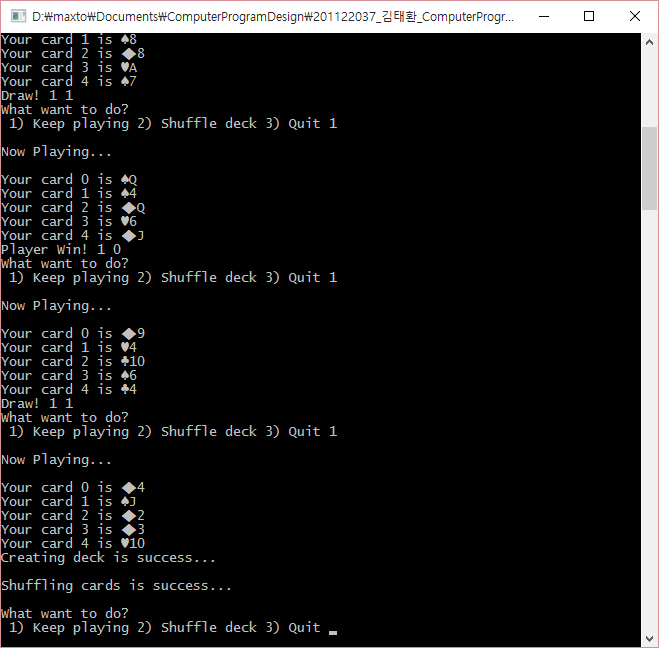
플레이 중 (1번)



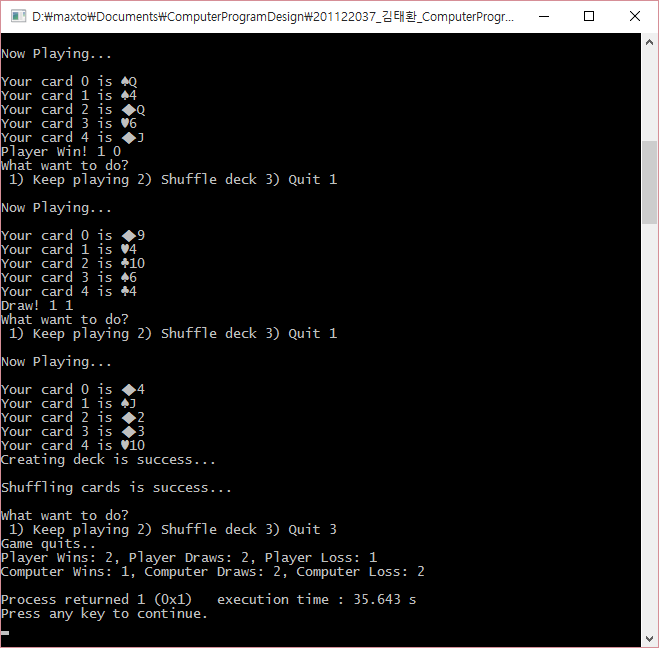
Deck 섞음 (2번)



카드를 다 써서 새로 Deck을 만들고 섞음



게임 종료 시 전적 출력 (3번)



# 프로그램 알고리즘

## main.c

1. 플레이어(player)와 컴퓨터(computer)를 초기화하고 카드 덱을 만든다.
2. 사용자에게 input을 받는다. (choice 변수)
3. choice의 값이 3이 아닌 동안 아래를 계속 수행한다.choice가 1이면 4~ 6, 2면 7을 수행한다. 3을 입력하면 8을 수행하고 프로그램을 종료한다.
4. player와 compute에게 card를 다섯 장씩 번갈아 가며 배분한다. 카드 배열에서 카드를 얻어오기 위해 get\_card( )라는 함수를 사용했다. 또한 현재 플레이어가 가진 카드를 출력하기 위해 info()라는 함수를 사용했다.
5. 현재 카드 배열의 어느 지점을 플레이어들이 가지고 있는지 확인하기 위해 curIdx라는 변수가 존재한다. 카드를 50장 이상 사용하면 자동으로 카드 배열을 초기화하도록 했다.
6. decideWinner() 에 player와 computer에 카드를 매개변수로 넣어 승패를 가린다. 함수의 반환값이 1이면 플레이어의 승리, 2면 컴퓨터의 승리, 0이면 무승부이다. 결과를 플레이어와 컴퓨터의 승패기록 변수에 저장한다.
7. 카드 덱을 초기화한 후 섞는다.
8. 플레이어와 컴퓨터의 전적을 출력하고 프로그램을 종료한다.

## card.h

헤더파일로써 프로그렘에서 사용되는 각종 열거형 상수들과 함수 정의가 담겨있다. 어떤 역할을 하는 struct 및 function인지 comment가 달려 있고 아래에서 설명할 예정이므로 여기서의 설명은 생략한다.

## card.c

main.c 에서 사용한 함수들의 함수 본문이 담겨있는 파일이다.

### void info(CARD card);

// get\_suit() 와 get\_rank()를 이용해 무늬와 숫자를 출력하는 함수이다.

### char\* get\_rank(RANK rank); char\* get\_suit(SUIT suit);

// 무늬와 숫자를 매개변수로 받아 switch case 문을 이용해 그 무늬와 숫자를 string으로 반환하는 함수다. 올바른 무늬나 숫자가 들어오지 않았다면 “E”를 반환하도록 되어 있다. 또한 사용된 카드라면 “USED”를 반환한다.

### CARD get\_card(CARD\* decks, int idx);

// 카드 배열과 얻고 싶은 인덱스를 매개변수로 받아 그 카드를 반환하는 함수이다. 생각해보면 필요가 없는 함수지만, 현재 사용된 카드의 rank와 suit를 열거형 상수 USED로 바꿔주는 역할을 한다. 또한 카드를 다 사용했다면 EMPTY 상수를 반환해서 현재 카드 덱에 카드가 없다는 사실을 알 수 있게 해준다.

### int decideWinner(CARD\* p, CARD\* c);

// 플레이어1,2의 카드 배열(5장)을 받아 실제로 포커 족보를 판정해주고 승패를 가려주는 함수이다. 플레이어 1이 이겼으면 1, 2가 이겼으면 2, 무승부라면 0을 반환한다.

먼저 플레이어와 컴퓨터에 족보에 해당하는 변수를 초기화한다. 그리고 플러시나 스트레이트를 가리기 위한 flag 변수들을 선언해준다.

다음은 플러시인지 알아내기 위해서 각 플레이어의 카드를 card\_quicksort() 라는 함수를 이용해 무늬 별로 정렬한다. 그 다음 for문으로 첫 번째 카드의 무늬와 나머지 카드의 무늬가 같은지 비교하여 맞으면 isPlushP은 1로 유지될 것이다.

역시 스트레이트인지 알아내기 위해 이번엔 숫자 별로 카드를 정렬한다. 스트레이트는 카드들이 각각 차이가 1인 등차수열을 이루는 족보이다. 따라서 등차수열의 일반항을 이용해 for문으로 스트레이트인지 판정했다.

다음은 나머지 족보를 판정하기 위해 각각의 족보에 해당하는 모든 경우를 if else문으로 구현했다. 그리고 그 결과를 각각 player 및 computer 변수에 넣는다. 좀더 효율적인 방법이 없을까 고민해봤지만, 결국 if문이 많아지는 걸 피할 수 없을 것 같아서 현재는 이 방법이 제일 효율적이라고 생각한다. 이제 그렇게 얻어진 player와 computer 변수의 크기를 비교하여 큰 쪽을 이겼다고 판정한다.

### void card\_quicksort(CARD\* cards, int size, int criteria);

// 카드들을 퀵소트로 정렬하는 함수이다. 알고리즘 교과서를 참고하여 퀵소트를 구현했고, 사용할 일이 많고 이미 카드가 정렬 되 있는 경우는 거의 없으므로 효율적일 것이라고 생각했다.

### CARD\* create\_deck(void);

// 카드 덱을 만들어서 반환하는 함수로 나머지 연산을 이용해 각 카드들이 고유한 숫자와 무늬를 가지도록 구현했다.

### void shuffle\_card(CARD cards[]);

// 만들어진 카드 덱을 섞는 함수로 rand() % DECK\_SIZE를 통해 숫자가 0~51까지만 나오게 통제하고, 그렇게 얻어진 index를 이용해 각 카드를 swap 하는 식으로 구현했다.