CSED101 Programming & Problem Solving
Fall, 2018
Programming Assignment #2

무은재 새내기학부
20180038
박형규
POVIS ID : hyeongkyu

Honor Code : 나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.

1. 프로그램 개요

1.1 목적

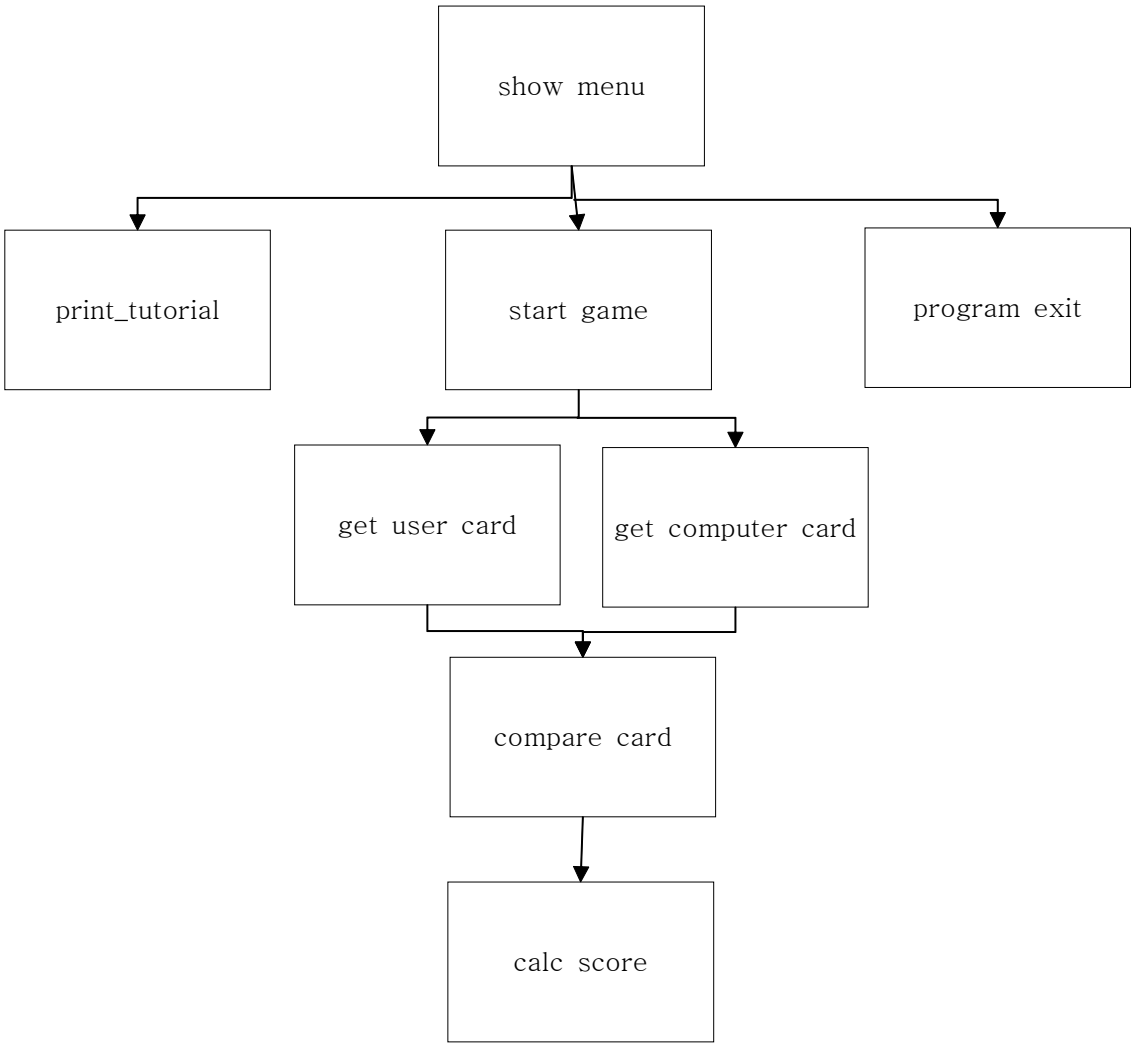
이번 프로그램의 목적은 if 함수와 같은 조건문, for, while, do while과 같은 반복문, 그리고 여러 가지 사용자 정의 함수와 여러 가지 라이브러리 함수 사용법을 익히고 이 함수들을 활용하여 제시된 문제를 해결하는 데에 있다.

1.2 전체 구조

main 함수에서 show_menu함수를 호출하여 프로그램이 시작된다. show_menu함수에서는 1, 2, 3번을 선택 할 수 있게 하여 1번을 누르면 print_tutorial 함수가 호출되어 게임 방법을 설명해주고, 2번을 누르면 start_game함수를 호출하여 게임을 시작할 수 있도록 하고, 3번을 누르면 게임을 종료 할 수 있도록 한다.

2번을 눌러 게임이 시작되었으면 처음에 황제패와 노예패를 선택할 수 있다. 각각 선택한 패에 따라서 카드 5장을 가지고 게임이 시작되게 되는데, 5장은 시민카드 4장과 각각 선택한 패의 카드 1장(노예패를 선택하면 노예카드 1장, 황제패를 선택하면 황제카드 1장)을 가지게 된다. 그 후에는 get_user_card 함수를 호출하여 카드 5장 중 사용자가 어떤 카드를 선택할지 고르게 된다. 그 후에는 get_computer_card 함수를 호출하여 컴퓨터가 임의로 카드 한 장을 고르고 compare_card 함수를 호출하여 나의 카드와 컴퓨터 카드를 비교한다. 이 때 승부는 황제>시민>노예>황제 순으로 결정되며, 서로 시민 카드를 뽑았을 때에는 무승부로 같은 회차에서 시민 카드 한 장을 소비한 후에 게임을 다시 진행하게 된다. 승부가 나면 calc_score함수를 호출하여 나와 컴퓨터의 점수를 계산한 후 만약 나와 컴퓨터의 점수중 하나라도 2000점이 넘으면 게임을 종료하게 되고, 서로 2000점 미만의 점수를 가지고 있으면 다음 게임을 진행 할 것인지 사용자의 입력에 따라 결정하게 된다.

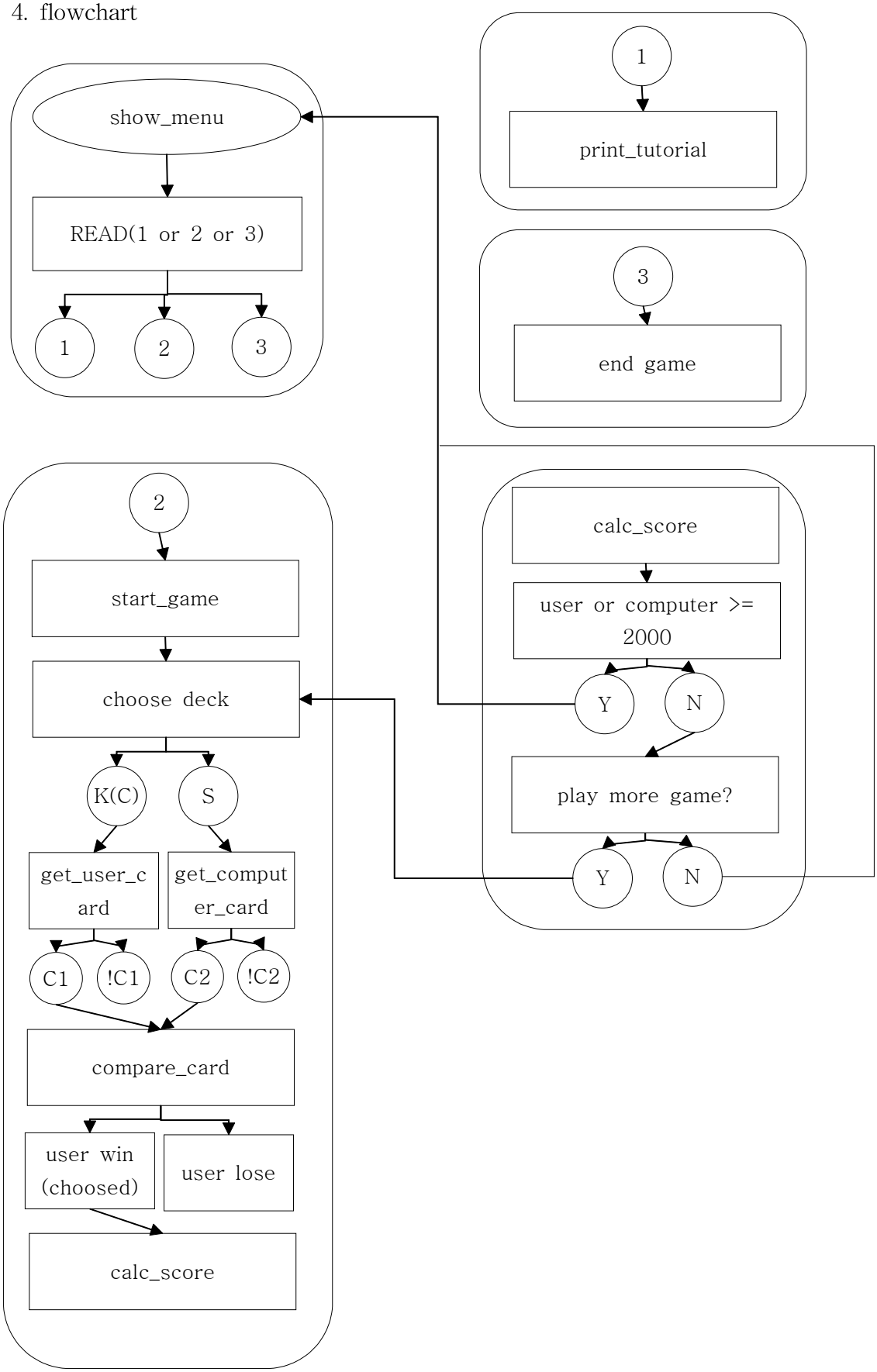
2. 전체 구조도



3. pseudocode

Algorithm E-Card
1 print show_menu
2. if 1; print_tutorial, if 2; start_game; if 3, end game
3. 2; start_game; print deck menu(start_game); choose deck (0 -> King or 1 -> Slave)
4. print card menu(get_user_card); choose card (1 -> 4 Citizen card, 0 -> 1 King/Slave card)
5 get_computer_card —> rand() % 2 = 0(king / slave) or 1(citizen)
6 compare_card; print result -> King > Citizen > Slave > King
7 calc_score; print score result; if user or computer score over 2000, end game; else choose continue or end game

4. flowchart


show_menu
READ(1 or 2 or 3)
1
2
3
1
print_tutorial
3
end game
2
start_game
choose deck
K(C)
S
get_user_card
get_computer_card
C1
!C1
C2
!C2
compare_card
user win (choosed)
user lose
calc_score
calc_score
user or computer >= 2000
Y
N
play more game?
Y
N

5. 프로그램 코드

Header file 포함

```
#include <stdio.h>
#include <stdlib.h>
#include <Windows.h>
#include <time.h>
```

Main 함수

```
int main()
{
show_menu();

return 0;
}
```

main 함수에는 콘솔 창이 시작되면 가장 처음에 보여야 할 화면인 menu화면을 띄우기 위해 show_menu 함수를 호출하였다.

show_menu의 print부분

```
int show_menu()
{
int num;
printf("********** E - Card **********₩n");
printf("            1. 게임 설명  ₩n");
printf("            2. 게임 시작  ₩n");
printf("            3. 게임 종료  ₩n");
printf("**********************************₩n");
printf("");
scanf("%d", &num);
getchar();
```

콘솔창을 띄웠을 때 가장 먼저 보일 menu창을 printf 함수를 이용하여 제작하였다. 또한 scanf 함수를 이용하여 사용자가 1,2,3중 한 가지 숫자를 입력하여 num 변수에 저장되게 하여 이용할 프로그램을 선택하도록 하였다.

show_menu의 swith-case 함수 부분

```
switch (num)
{
case 1: system("cls");
print_tutorial();
break;
case 2: start_game();
break;
case 3: printf("프로그램을 종료합니다...“);
exit(0);
default: printf("올바른 메뉴를 선택하세요...“);
Sleep(1000);
system("cls");
show_menu();
}
```

사용자가 입력한 num값이 1이면 print_tutorial 함수를 호출하여 게임 방법을 설명하게 하였고 2이면 start_game 함수를 호출하여 게임을 시작하도록 하였으며, 3이면 프로그램이 종료되도록 하였다. 또한 1,2,3 이외의 값을 입력하면 “올바른 메뉴를 선택하세요...”라는 문구가 뜬 후 1초뒤에 화면이 사라지고 다시 메뉴 창이 뜨도록 하였다.

print_tutorial

```
void print_tutorial()
{
char c;
printf("************ E - Card 게임 설명**************************************₩n");
printf("₩n)
printf("본 게임은 도박묵시록 카이지의 E - Card 게임을 기반으로 한다.            ₩n");
printf("₩n");
printf("1. 두 플레이어는 황제패와 노예패 중 하나의 패를 각각 선택한다.          ₩n");
printf("    황제패 = 황제 카드 1장 + 시민 카드 4장                              ₩n");
printf("    노예패 = 노예 카드 1장 + 시민 카드 4장                              ₩n");
printf("                                                                        ₩n");
printf("2. 두 사용자는 매 턴마다 한 장의 카드를 안보이게 내밀고, 함께 뒤집는다.  ₩n");
printf("                                                                        ₩n");
printf("3. 아래의 카드 간 상성 관계에 따라 승패를 결정한다.                      ₩n");
printf("    단, 두 플레이어가 시민 카드를 냈다면, 무승부로 처리한다.             ₩n");
printf("                                                                        ₩n");
printf("4. 내밀었던 카드는 소모되고 승패가 결정될 때까지 2로 돌아가 반복한다.   ₩n");
printf("                                                                        ₩n");
printf("상성관계 : 황제 > 시민 > 노예 > 황제                                     ₩n");
printf("                                                                        ₩n");
printf("************************************************************************₩n");
printf("메뉴로 돌아가려면 Enter키를 입력하세요...“);

scanf("%c", &c);
system("cls");
show_menu();
```

사용자가 num으로 1을 입력하였을 때 나타나는 게임 설명이다.

start_game의 변수 지정 부분

```
int select;
int i, j;
int ms = 0;
int cs = 0;
i = 1;
```

select는 황제패를 골랐는지, 노예패를 골랐는지 구분해주는 패이며(0이면 황제패, 1이면 노예패) I는 게임 횟수를 나타내는 변수이고, j는 한 게임 횟수에서 반복되는 횟수를 나타내는 변수이다. 또한 ms는 나의 점수를 나타내고, cs는 컴퓨터의 점수를 나타낸다.

start_game의 while문 안에서의 변수 지정 부분과 print 부분

```
while(1)
{
int a, b, c, d, e, f, g, h, k;
int m = 0;
int n = 0;
system("cls");
printf("[게임 횟수 : %d, 컴퓨터 %d점, 나 : %d점]  ₩n", i, cs, ms);
printf("                                        ₩n");
printf("[카드 패 선택]                          ₩n");
printf("**************************************₩n");
printf("                                        ₩n");
printf("0. 황제패 (황제 1장, 시민 4장)          ₩n");
printf("1. 노예패 (노예 1장, 시민 4장)          ₩n");
printf("                                        ₩n");
printf("**************************************₩n");
printf("입력 : ");
scanf("%d", &select);
getchar();
```

a, b, c, d, e, f, g, h, k는 각각 아래의 switch-case 문에서 쓰일 변수를 나타내며, 나의 점수와 컴퓨터의 점수의 초기값은 각각 0이므로 초기값을 0으로 설정해준다. 또한 마지막에 scanf 함수를 통하여 select 값을 입력받아 황제패를 고를지, 노예패를 고를지 입력받는다.

start_game while문 중 switch-case 함수중 case 0

```
switch (select)
{
case 0: for (j = 1; j < 6; j++)
{
if (j == 1)
{
int Eis_selected = 0;
int Sis_selected = 0;
system("cls");
a = get_user_card(select, i, j, &Eis_selected, &Sis_selected, &ms, &cs);
b = get_computer_card();
m = compare_card(select, a, b, &ms, &cs);
if (m == 0)
{
continue;
}
else
{
break;
}
}
else
{
int Eis_selected;
int Sis_selected = 0;
system("cls");
g = get_user_card(select, i, j, &Eis_selected, &Sis_selected, &ms, &cs);
h = get_computer_card();
n = compare_card(select, g, h, &ms, &cs);
if (n == 0)
{
continue;
}
else
{
break;
}
}
}
```

먼저 case 0 문 안에서 j를 변수로 가지는 for함수를 정의한다 이는 결과가 무승부가 나올 때를 대비하여 만든 함수로 한 사람의 카드가 총 5장이기 때문에 최대 5번 반복하는 것으로 한다. 또한 Eis_selected는 이전에 황제가 선택되었는지를 나타내는 변수이고(선택되었으면 1, 선택되지 않았으면 0) Sis_selected는 이전에 노예가 선택되었는지를 나타내는 변수이다. 먼저 j=1일 경우에는 한 회차에서 처음 실행되

는 경우이므로 황제와 노예가 선택되었을 리가 없다. 따라서 Eis_selected와 Sis_selected를 모두 0으로 지정한다. 그 뒤, get_user_card 함수를 호출하여 사용자가 내밀 카드를 고르고, get_computer_card를 호출하여 컴퓨터가 내밀 카드를 호출한다. 그 뒤 사용자가 내민 카드와 컴퓨터가 내민 카드를 비교하는 compare_card를 호출하게 된다. 이 때, compare_card 함수는 int형 함수로 정수 값을 return 받게 되는데 0을 return 받으면 무승부라는 뜻으로 continue를 통해 j=2로 넘어가게 되고, 0 이외의 값을 return 받으면 승부가 났다는 뜻으로 다음 회차(i++)로 넘어가게 된다. 만약 1회차에서 무승부가 발생하게 된다면 j=2로 넘어가게 되는데 이 때에도 앞에서와 같이 get_user_card, get_computer_card, compare_card를 통해 승부를 내게 된다. 이와 같은 process를 최대 5번까지 반복하면 무조건 승부가 나고, 승부가 났을 경우 I++을 통하여 다음 회차로 넘어가게 된다.

switch-case 문에서 case 1의 경우는 노예 팩을 선택하여 진행하는 경우인데 위의 경우에서 황제가 노예로만 바뀌었을 뿐이므로, 생략한다.

get_user_card 의 콘솔창에 print 되는 부분

```
int king, city;
printf("[게임횟수 : %d  ,컴퓨터 : %d점, 나 : %d점]  ₩n", i, *cs, *ms);
printf("                                    ₩n");
printf("************************************₩n");
show_table(j, select, &Eis_selected, &Sis_selected);
printf("************************************₩n");
printf("                                    ₩n");
printf("[카드선택]                          ₩n");
printf("******************                  ₩n");
printf("                                    ₩n");
if (*Eis_selected == 1)
{
king = 0;
city = 6 - j;
}
else
{
king = 1;
city = 5 - j;
}
printf("0. 황제 (%d장)                      ₩n", king);
printf("1. 시민 (%d장)                      ₩n", city);
printf("                                    ₩n");
printf("******************                  ₩n");
if (j != 5)
{
printf("입력 : ");
scanf("%d", &numb);
getchar();
}
else
{
printf("입력 : ", numb = 0);
}
if (numb == 0)
{
*Eis_selected = 1;
}
else
{
if (king == 0)
{
*Eis_selected = 1;
}
else if (king == 1)
{
*Eis_selected = 0;
}
```

get_user_card에서 현재 게임 횟수, 나와 컴퓨터의 점수를 표시하고, show_table 함수를 호출하여 현재 남아있는 카드 개수와 종류를 보여준다. 또한 [카드 선택] 란에서도 카드 종류와 종류에 따라 남아있는 카드 개수를 보여준다. 또한 아래의 if문은 사용자가 0을 입력하면 황제 카드가 사용되었다는 뜻이므로 Eis_selected의 값을 1로 지정해주고, 사용자가 1을 입력하였을 경우에는 아직 황제 카드가 사용되지 않았다는 뜻이므로 Eis_selected에 0의 값을 저장한다. 또한 4번 연속 시민카드를 선택하여 무승부가 4번 연속으로 나왔을 때는 5번째에 낼 수 있는 카드가 0번 카드 1장밖에 없기 때문에 자동으로 0번이 선택되고, 승부가 결정 되도록 하였다.

아래의 else문은 get_user_card에서 위에 나왔던 내용을 황제 카드에서 노예 카드로만 바꾸면 되는 내용이므로 생략한다.

get_computer_card

```
int seed;
seed = time(NULL);
srand(seed);
int card = rand() % 2;

return card;
```

컴퓨터가 선택할 수 있는 카드의 종류는 두 종류 뿐이므로 rand함수를 이용하여 컴퓨터가 카드를 선택하게 한 것이다.

compare_card중 사용자가 황제 댁을 골랐을 경우 카드 제시 / 결과 제시

```
if (a == 0)
{
if (x == 0)
{
printf("* 나의 카드     : 황제₩n");
}
else
{
printf("* 나의 카드     : 시민₩n");
}
if (y == 0)
{
printf("* 컴퓨터의 카드 : 노예₩n");
}
else
{
printf("* 컴퓨터의 카드 : 시민₩n");
}
if (x == 0 && y == 0)
{
printf("* 결과 : 나의 패배          ₩n");
result= 2;
}
else if (x == 0 && y == 1)
{
printf("* 결과 : 나의 승리          ₩n");
result= 1;
}
else if (x == 1 && y == 0)
{
printf("* 결과 : 나의 승리          ₩n");
result= 1;
}
else // 0
{
printf("* 결과 : 무승부          ₩n");
result= 0;
}
```

compare_card 함수에서는 start_game함수로부터 황제 팩/ 노예 팩의 선택 여부, 팩 안에서의 사용자의 카드 선택 여부, 팩 안에서의 컴퓨터의 카드 선택 여부를 매개변수로 가지고 온다. 위의 상황은 황제 팩을 선택하였을 경우인데, 사용자가 선택한 카드와 컴퓨터가 선택한 카드를 비교하여 결과를 내는 것이다. 이 때, 결과값 반환을 위하여 무승부일 경우에는 0, 사용자가 승리할 경우에는 1, 컴퓨터가 승리할 경우에는 2를 반환하도록 하였다.

compare_card중 사용자가 황제 댁을 골랐을 경우 calc_score 호출

```
if (a == 0)
{
if (x == 1 && y == 1)
{
printf("다음 턴 (Enter키를 입력하세요...)“);
char c;
scanf("%c", &c);
system("cls");
}
else if (x == 0 && y == 1)
{
calc_score(a, x, y, ms, cs);
}
else if (x == 1 && y == 0)
{
calc_score(a, x, y, ms, cs);
}
else
{
calc_score(a, x, y, ms, cs);
}
}
```

if 함수를 통하여 무승부일 때는 Enter 키를 입력하면 다음 턴으로 넘어가도록 하였고, 무승부가 아닌 경우에는 calc_score 함수를 호출하여 점수를 계산하도록 하였다.

calc_score함수 (사용자가 황제 댁을 골랐을 경우)

```
char q;
printf("[현재 점수]₩n");
printf("**********₩n");
printf("₩n");
if (select == 0)
{
if (a == 0 && b == 1)
{
*ms = *ms + 200;
*cs = *cs - 300;
}
else if (a == 1 && b == 0)
{
*ms = *ms + 200;
*cs = *cs - 300;
}
else if (a == 0 && b == 0)
{
*ms = *ms - 500;
*cs = *cs + 700;
}
else
{
*ms = *ms
*cs = *cs
}
}
printf("* 나의 점수   : %d₩n", *ms);
printf("* 컴퓨터의 점수 : %d₩n", *cs);
printf("₩n");
printf("**********₩n");
printf("₩n");
```

사용자와 컴퓨터의 승 패 결과에 따라서 사용자와 컴퓨터의 점수를 조정한다. 이때, 사용자와 컴퓨터의 점수값은 포인터 변수로 받아와서 역참조 연산자를 이용하여 점수값을 계산한다.

calc_score에서 마지막 단계

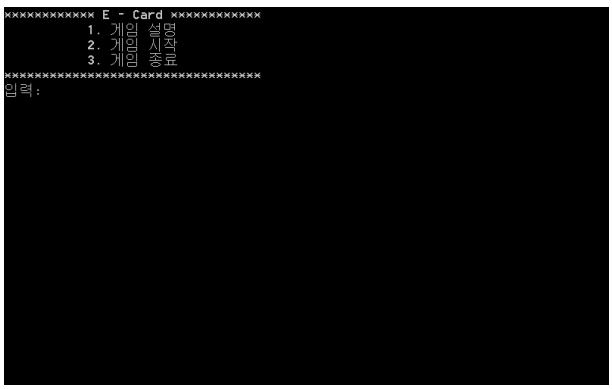
```
if (*ms >= 2000)
{
printf("[최종 결과]₩n");
printf("**********");
printf("₩n");
printf("* 당신의 승리!");
printf("₩n");
printf("**********");
printf("₩n");
printf("Enter키를 입력하세요...“);
char c;
scanf("%c", &c);
system("cls");
show_menu();
}
else if (*cs >= 2000) // 2000
{
printf("[최종 결과]₩n");
printf("**********");
printf("₩n");
printf("* 당신의 패배!");
printf("₩n");
printf("**********");
printf("₩n");
printf("Enter키를 입력하세요...);
char c;
scanf("%c", &c);
system("cls");
show_menu();
}
else
{
printf("게임을 계속하시겠습니까? (y/n): ");
scanf("%c", &q);
getchar();
if (q == 'y' || q == 'Y')
{
system("cls");
}
else
{
system("cls");
show_menu();
}
}
```

if 함수를 이용하여 사용자나 컴퓨터의 점수중 하나라도 2000점을 초과하게 된다면 게임이 종료되고, 처음 화면인 menu 화면으로 넘어가게 된다. 또한 사용자나 컴퓨터의 점수 중 어느 것도 2000점을 넘지 않으면 게임을 계속할지 여부를 y/n을 사용자에게 입력받아 정하여, y(Y)를 입력받으면 다음 회차로 넘어가서 게임을 계속 하게 되고, n(N)을 입력받으면 메뉴 창으로 이동하게 된다.

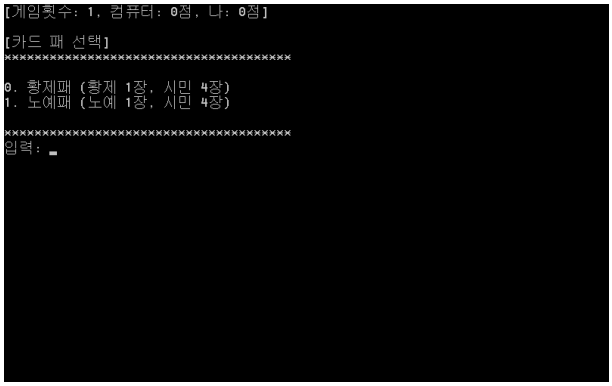
6. 프로그램 실행 방법

처음 메뉴에서 1,2,3 중의 값을 입력한다. 1을 입력하면 게임 설명을 볼 수 있고, 2를 입력하면 게임을 시작 할 수 있으며, 3을 입력하면 게임이 종료된다. 게임을 시작하면 황제패(0)와 노예패(1) 중 하나를 고를 수 있다. 패를 고르면 패 안에 포함된 5장의 카드 중 어떤 카드를 내밀지를 선택 할 수 있다. 카드를 내밀면 내가 고른 카드와 컴퓨터가 고른 카드를 비교하여 결과가 나오고, 결과에 따라 게임을 계속할 지의 여부를 선택할 수 있다. 이 때, 게임을 계속하다가 사용자나 컴퓨터의 점수중 하나라도 2000점을 넘어가면 게임이 끝나게 되고 처음 메뉴 화면으로 넘어가게 된다.

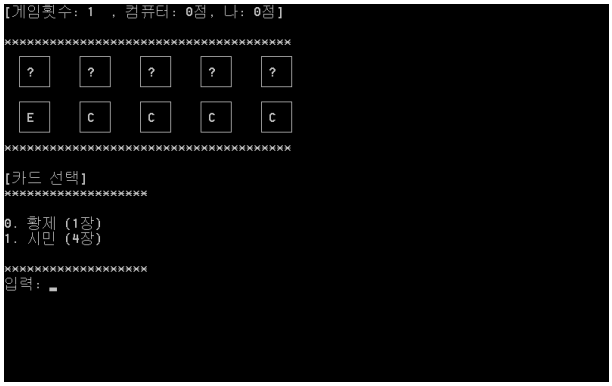
7. 프로그램 실행 예제



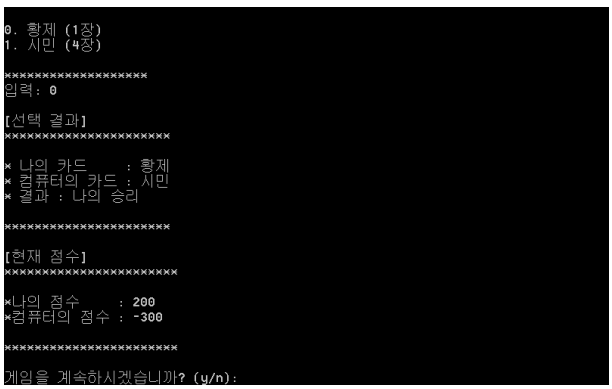
처음 프로그램을 실행하였을 때의 메뉴 화면이다.



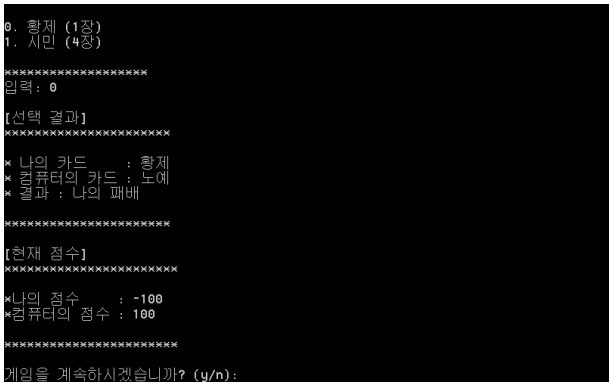
게임을 시작하였을 때, 황제패와 노예패를 고를 수 있는 화면이다.



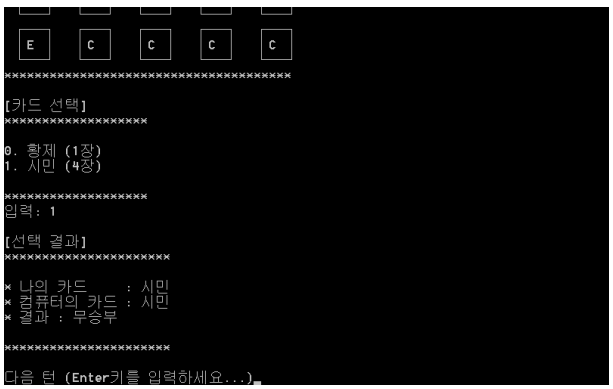
패를 고른 후에 패 안에서 카드를 고르는 화면이다.



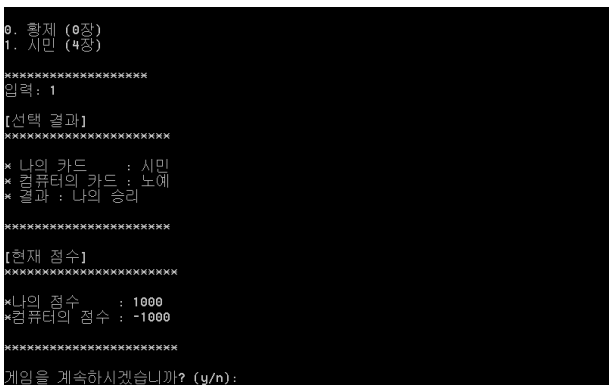
황제카드와 시민카드가 나왔을 때 황제카드가 이기는 장면이다.



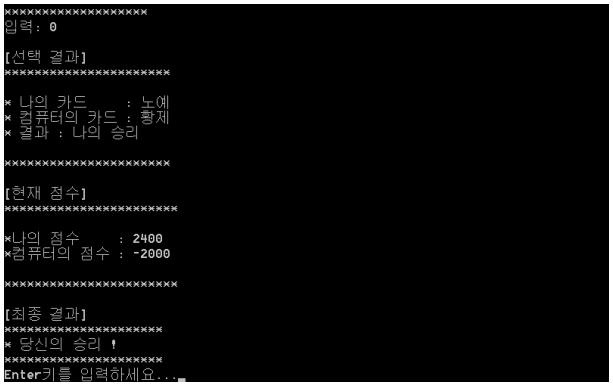
황제카드와 노예카드가 나왔을 때 노예카드가 이기는 장면이다.



시민카드와 시민카드가 나왔을 때 무승부 되는 장면이다.



시민카드와 노예카드가 나왔을 때 시민카드가 이기는 장면이다.



나의 점수가 2000점을 넘어서 게임이 종료되고 최종 결과가 제시된 모습이다.

8. 토론 및 결론

이번 어싸인을 하면서 사용자 함수를 사용하고, 반복문, 조건문 등을 사용하는 법을 제대로 익힐 수 있었던 것 같다. 카드를 만들때는 오직 조건문만을 사용하여 카드를 모두 그려 낸 것이 아니라 조건문과 반복문을 함께 사용하여 상황에 맞게 카드 개수를 알아서 설정하고 카드를 콘솔창에 내보낼 수 있게 한 것이 반복문을 사용하는 데 조금이나마 더 익숙해지는데 도움이 되었다. 또한 처음 main 함수부터 마지막 calc_score 함수까지 유기적으로 연결시켜서 flow chart에 나와 있는 대로 값을 반환해가며 함수를 유기적으로 연동하는 점이 흥미로웠다. 처음 어싸인을 보았을 때는 무엇부터 손대야 할지 막막했지만 처음에 이 문제를 어떻게 해결해야 할지 생각하고, structure chart와 flow chart, 그리고 pseudocode를 그려보면서 이 문제의 구조를 이해하였더니 처음에 생각하였던 것만큼 막막한 문제라고는 생각되지 않았다. 이번 어싸인을 통해 이해한 함수들의 유기적 연관 구조, 반복문과 조건문의 사용 방법등을 이용하여 다음 어싸인에서는 새로 배운 내용들과 함께 이 내용들도 조금 더 숙달되어 능숙히 사용하는 내 모습을 상상하며 보고서를 끝마친다.