\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Report: Author: F74055047張財實 <changchaishi@gmail.com>

Class: 乙班

Description: Understanding how essentially is the makefile for compiling a larger program. From this homework, we can master our call of function to another level.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

main.h:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>

#define NUM\_SUITS 4

#define NUM\_RANKS 13

#define NUM\_CARDS 5

#define PLAYER 4

#define RANK 0

#define SUIT 1

bool straight ,flush,three,four;

bool in\_hand[NUM\_RANKS][NUM\_SUITS];

int pairs;

int pattern\_point,rank\_point,suit\_point;

int pokercard[NUM\_CARDS][2];

int num\_in\_rank[NUM\_RANKS];

int num\_in\_suit[NUM\_SUITS];

int score[PLAYER];

int main();

main.c:

#include "main.h"

#include "analysis.h"

#include "winner.h"

int main(){

srand((unsigned)time(NULL));

int p;

for(p=0;p<PLAYER;p++){

dealcard(p);

analysis();

cardlevel();

player\_point(p);

printf("\n");

}

winner();

return 0;

}

analysis.h:

#include "main.h"

void dealcard(int);

void analysis();

void highcard();

void cardlevel();

int suitmax(void);

void player\_point(int);

analysis.c:

#include "analysis.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

dealcard:distribute different cards for each player

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void dealcard(int p){

printf("PLAYER %d: ",p+1);

int suit ,rank;

int i=NUM\_CARDS;

for(rank=0;rank<NUM\_RANKS;rank++)

num\_in\_rank[rank]=0;

for(suit=0;suit<NUM\_SUITS;suit++)

num\_in\_suit[suit]=0;

while(i>0){

suit=rand()%NUM\_SUITS;

rank=rand()%NUM\_RANKS;

if(in\_hand[rank][suit])

continue;

pokercard[i][RANK]=rank;

pokercard[i][SUIT]=suit;

num\_in\_rank[rank]++;

num\_in\_suit[suit]++;

switch(rank){

case 0:case 1:case 2:case 3:

case 4:case 5:case 6:case 7:

printf("%d",rank+2);break;

case 8:printf("T");break;

case 9:printf("J");break;

case 10:printf("Q");break;

case 11:printf("K");break;

case 12:printf("A");break;

}

switch(suit){

case 0:printf("C");break;

case 1:printf("D");break;

case 2:printf("H");break;

case 3:printf("S");break;

}

in\_hand[rank][suit]=true;

printf(" ");

i--;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

analysis:To obtain the pattern occcurences meanwhile

stores the rank and suit points in responding pattern

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void analysis(int p){

int num\_consec=0;

int rank,suit;

straight=false;

flush=false;

four=false;

three=false;

pairs=0;

/\*flush\*/

for (suit = 0; suit < NUM\_SUITS; suit++)

{

if(num\_in\_suit[suit]==NUM\_CARDS)

flush = true;

highcard();

}

/\*straight\*/

rank=0;

while(num\_in\_rank[rank]==0)rank++;

for (; rank<NUM\_RANKS&&num\_in\_rank[rank] > 0; rank++)

num\_consec++;

if(num\_consec==NUM\_CARDS){

straight=true;

rank\_point=rank-1;suit\_point=suitmax();

return;

}

/\*check 4,3,and pairs\*/

for (rank = 0; rank < NUM\_RANKS; rank++)

{

if(num\_in\_rank[rank]==4){four=true;rank\_point=rank;suit\_point=suitmax();}

if(num\_in\_rank[rank]==3){three=true;rank\_point=rank;suit\_point=suitmax();}

}

for (rank = 0; rank < NUM\_RANKS; rank++)

{

if(num\_in\_rank[rank]==2){

pairs++;

if(three)

return;

rank\_point=rank;

suit\_point=suitmax();

}

}

highcard();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cardlevel:this function is to obtain the pattern

point

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void cardlevel(void){

if(straight&&flush){pattern\_point=8;}

else if(four){pattern\_point=7;}

else if(three&&pairs==1){pattern\_point=6;}

else if(flush){pattern\_point=5;}

else if(straight){pattern\_point=4;}

else if(three){pattern\_point=3;}

else if(pairs==2){pattern\_point=2;}

else if(pairs==1){pattern\_point=1;}

else {pattern\_point=0;}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

suitmax:with same pattern and rank,the next we have to

compare the suit ,suitmax resolves it,and called by

analysis function when it meets the requairement.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int suitmax(void)

{

int i,max=0;

for(i=0;i<NUM\_CARDS;i++){

if(pokercard[i][RANK]==rank\_point){

if(pokercard[i][SUIT]>max)max=pokercard[i][SUIT];

}

}

return max;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

highcard:only occurs in flush or no-pattern,it is

called in analysis

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void highcard(void){

int i,max=0;

for(i=0;i<NUM\_CARDS;i++){

if(pokercard[i][RANK]>max)max=pokercard[i][RANK];

}

rank\_point=max;

suit\_point=suitmax();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

player point:evaluate the score for each player

pattern stands for 1000pts,rank for 10points,suit

for 1~4.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void player\_point(int p){

score[p]=1000\*pattern\_point+10\*rank\_point+suit\_point;

}

winner.h:

#include "main.h"

void winner(void);

winner.c:

#include "winner.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

winner:getting result and prints it,comparing the

scores will do the rest.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void winner(void){

int p,max=0,winner;

for(p=0;p<PLAYER;p++)

if(score[p]>max)max=score[p];

for(p=0;p<PLAYER;p++)

if(score[p]==max)winner=p;

printf("Winner: Player %d--",winner+1);

switch(score[winner]/1000){

case 8:printf("Straight flush");break;

case 7:printf("Four of a kind");break;

case 6:printf("Full house");break;

case 5:printf("Flush");break;

case 4:printf("Straight");break;

case 3:printf("Three of a kind");break;

case 2:printf("Two pairs");break;

case 1:printf("Pair");break;

case 0:printf("Highcard");break;

}

printf("\n");

}

makefile:

hw6:main.o analysis.o winner.o

gcc -o hw6 main.c analysis.c winner.c

main.o:main.c main.h

gcc -c main.c

analysis.o:analysis.c main.h analysis.h

gcc -c analysis.c

winner.o:winner.c main.h winner.h

gcc -c winner.c

Compilation:

make

Output:

PLAYER 1: 6C AH 3H TD JH

PLAYER 2: JC 7C 6D 5H TC

PLAYER 3: 2S TS 3D 2C KC

PLAYER 4: 4D 2H JS 3S AS

Winner: Player 3--Pair