Final Project

40947071S 許証淵

包含檔案:

all.h
document.pdf
document.docx
game.c
game.o
main.c
Makefile
final Project.ppt

Structure 結構:

Each card has a structure.

```
typedef struct structureOfCard
{
    int32_t                     cardNumber;
    char*                       cardName;
    char*                       cardDescription;
    int32_t                     cardCost;
    bool                        cardHasProduct;
    int32_t                     cardScore;
    struct structureOfCard*     linkedListNext;
    int32_t                     subcard; //紀錄分數用的
} __attribute__((packed)) sCard;
```

Also, each "player" has a structure

```
typedef struct
{
    sCard*      cardOnHand;
    sCard       cardOnTable[13];
    int32_t     number_cardOnHand;
    int32_t     number_cardOnTable;
    int32_t     pointsReceive;

}__attribute__((packed)) sPlayer;
```

Functions (Before the Game Starts):

```
void            printIntroduction();
//版面
int32_t         showSelectionList();
//選擇開始or離開
int32_t         getPlayerNumber();
//選擇玩家數量
void            printErrorMessage(int32_t messageNumber);
//print出錯誤訊息
void            initialize_Player(sPlayer *a, int32_t  amountOfPlayers);
//玩家初始化
void            initialize_Deck();
//卡牌初始化設定
```

```
void        initialize_Card(int32_t cardAmount, int32_t cardNumber, char* cardName, char* cardDescription, int32_t cardCost, int32_t cardPoint);
//卡牌初始化
void        shuffle(int32_t amount_InitializedCard);
//洗牌
void        cardDistribute(sPlayer *player, int32_t numberOfPlayers, int32_t governor);
//發牌
void        draw(sPlayer* player, int32_t governor, int32_t amountOfCard);
//抽牌
```

Functions (During the Game):

```
void round_start(sPlayer* player,int32_t numberOfPlayers,int32_t governor);
//遊戲開始；透過 governor 找出是誰先需要有動作
void showCardsOnTable(sPlayer* player,int32_t numberOfPlayers);
//透過參數『玩家數量』把桌上的牌印出來
void discard(sPlayer* player, int32_t numberOfPlayer, sCard* Card);
//透過 card(實際上是一個節點) 來丟牌
void cardOnHands(sPlayer* player, int32_t numberOfPlayer);
//印出手牌
void cardOnHands_part(sPlayer* player, int32_t numberOfPlayer, sCard* startCar
d); //印出部分手牌
void displayCardDescription(sPlayer* player, int32_t amountOfPlayers, int32_t
numberOfPlayer);
//印出手中卡牌敘述
void display_TableCardDescription(sPlayer* player, int32_t amountOfPlayers, in
t32_t numberOfPlayer);
//印出桌上卡牌敘述
```

```
sCard* find_handcard(sPlayer* player, int32_t numberOfPlayer, int32_t card_id)
; //透過 id 找出特定卡牌
bool discardFromHand(sPlayer *player, int32_t amountOfPlayers, int32_t numberO
fPlayer, int32_t amountOfDiscardCard);
//透過最後一項參數找出要丟的數量，並搭配節點找出其 ID 與 discard function 搭配
int32_t chooseProfession(sPlayer* player,int32_t numberOfPlayers,int32_t profe
ssionOfPlayer);
//選擇職業，透過節點把選到的職業代回去
void chapel_MakeDecision(sPlayer* player, int32_t amountOfPlayer, int32_t numb
erOfPlayer, int32_t tableCardIndex);
//如果有教堂時，發動教堂功能，把卡排放到教堂下
bool office_MakeDecision(sPlayer* player, int32_t amountOfPlayer, int32_t numb
erOfPlayer, int32_t tableCardIndex);
//如果有辦公樓，則發動辦公樓的功能
void produce_product(sPlayer* player, int32_t numberOfPlayer, int32_t tableCar
dIndex);
//丟掉貨物，透過節點與桌上卡牌的 index 於功能結束時紀錄新的狀態。
bool game_end(sPlayer* player, int32_t amountOfPlayers);
//判別 main function 裡的無窮迴圈是否需要結束遊戲
```

Function(職業設定):

```
void pro_Builder(sPlayer* player,int32_t amountOfPlayer,int32_t professionOfPl
ayer,int32_t numberOfPlayers);
void build(sPlayer* player, int32_t amountOfPlayer,int32_t  professionOfPlayer
,int32_t numberOfPlayer);
//pro_的意思是職業，當 pro_被呼叫時，表示啟動該職業並根據玩家選擇決定是否在此
function 呼叫該職業之特權。
//建築師的選擇與特權

void pro_Councillor(sPlayer* player,int32_t amountOfPlayer,int32_t professionO
fPlayer,int32_t numberOfPlayers);
void councillor(sPlayer* player, int32_t amountOfPlayer, int32_t professionOfP
layer, int32_t numberOfPlayer);
//議員的選擇與特權。 原理同上。
```

```
void pro_Producer(sPlayer* player,int32_t amountOfPlayer,int32_t professionOfP
layer,int32_t numberOfPlayers);
//製造商的選擇與特權。

void pro_Prospector(sPlayer* player,int32_t amountOfPlayer,int32_t professionO
fPlayer,int32_t numberOfPlayers);
//礦工的選擇與特權。

void pro_Trader(sPlayer* player,int32_t amountOfPlayer,int32_t professionOfPla
yer,int32_t numberOfPlayers);
//貿易商的選擇與特權。
```

Function(機器人與雜項):

```
bool bot_decision(int32_t chance);
//透過 chance 比大小回傳要不要出牌
int32_t findCard(sPlayer* player, int32_t numberOfPlayer, int32_t cardID);
//透過 ID 找出是否存在在手牌或桌上
```

Function(卡牌效果):

所有的 function 都會先判斷是否是玩家，若非玩家則有機器人模式，隨機決定是否出牌。

```
void office(sPlayer* player, int32_t amountOfPlayers, int32_t numberOfPlayer,
int32_t tableCardIndex);
void bank(sPlayer* player, int32_t amountOfPlayers, int32_t numberOfPlayer, in
t32_t tableCardIndex);
void chapel(sPlayer* player, int32_t amountOfPlayers, int32_t numberOfPlayer,
int32_t tableCardIndex);
//處理將牌放到教堂底下的行動，還有紀錄分數
void carpenter(sPlayer* player, int32_t amountOfPlayers, int32_t numberOfPlaye
r, sCard* card);
//判別是否發動木工坊的條件，如果符合就多抽一張牌
void smithy(sPlayer* player, int32_t numberOfPlayer,sCard* card, int32_t* fee)
;
void quarry(sPlayer* player, int32_t numberOfPlayer,sCard* card, int32_t* fee)
;
```

```
void black_market(sPlayer* player, int32_t amountOfPlayers, int32_t numberOfPl
ayer,int32_t* fee);
void poor_house(sPlayer* player,int32_t amountOfPlayer, int32_t numberOfPlayer
); //判別是否發動濟貧院的條件，如果符合就多抽一張牌
void library(sPlayer* player, int32_t numberOfPlayer, int32_t* fee, int32_t pr
ofession);
int32_t crane(sPlayer* player, int32_t amountOfPlayers, int32_t numberOfPlayer
, sCard* card);
void archive(int32_t numberOfPlayer, int32_t* original_NumberOfHandcard);
void prefecture(int32_t numberOfPlayer, int32_t amountOfCard, int32_t* amountO
fDiscard);
void customs_office(sPlayer* player ,int32_t amountOfPlayer, int32_t numberOfP
layer, int32_t tablecardIdx);
void aquaduct(int32_t numberOfPlayer,int32_t* amountProduct);
void well(sPlayer* player,int32_t amountOfPlayers,int32_t numberOfPlayer,int32
_t amountProduct);
//判別是否符合發動條件，如果符合就多抽一張牌
void goldsmith(sPlayer* player, int32_t amountOfPlayers, int32_t numberOfPlaye
r);
void goldmine(sPlayer* player, int32_t amountOfPlayers, int32_t numberOfPlayer
);// //判別是否符合發動條件，如果符合就發動並處理玩家手牌增減
void trading_post(int32_t numberOfPlayer, int32_t* mostProduct);
void market_stand(sPlayer* player, int32_t amountOfPlayers, int32_t numberOfPl
ayer, int32_t amountProduct);
//判別是否符合發動條件，如果符合就多抽一張牌
void            market_hall(sPlayer* player, int32_t amountOfPlayers, int32_t
numberOfPlayer, int32_t amountProduct);
//判別是否符合發動條件，如果符合就多抽一張牌
void            harbor(sPlayer* player, int32_t amountOfPlayers, int32_t numbe
rOfPlayer, int32_t tableCardIndex);
void            tavern(sPlayer* player, int32_t amountOfPlayers, int32_t numbe
rOfPlayer);
void            cottage(sPlayer* player, int32_t amountOfPlayers, int32_t numb
erOfPlayer);
void            guild_hall(sPlayer* player, int32_t numberOfPlayer);
void            city_hall(sPlayer* player, int32_t numberOfPlayer);
void            triumphal_arch(sPlayer* player, int32_t numberOfPlayer);
void            palace(sPlayer* player, int32_t numberOfPlayer);
void            residence(sPlayer* player, int32_t numberOfPlayer);
```

```
//上面五個為處理總分結算是否有額外加分的 function
```