Computer Programming II Final Project - San Juan

資工113 40947031S 李聖澄

檔案

- main.c
- sanJuan.c
- card.c
- print.c
- sanJuan.h
- makefile

Struct

- player

```
typedef struct {
    int32_t vp; //victory point
    sCard* handcard;    //指向第一張手牌
    sCard tablecard[13];    //場上該玩家區域的牌，僅使用index 1 to 12
    int32_t num_of_handcard;    //手牌數
    int32_t num_of_tablecard;   //該玩家場上卡牌數量
} sPlayer;
```

- card

```
typedef struct _sCard{
    int32_t id; //卡片id，方便辨識用
    char* name; //卡片名稱
    char* description;  //卡片敘述
    int32_t cost;   //建築所需費用
    int32_t score;  //得分
    bool hasProduct;    //是否有生產貨物
    int32_t subcard;    //在場上時，任何附屬於此卡片的子卡數
    struct _sCard* next;    //Linked list時使用之link
} sCard;
```

全域變數

```
char* clear = "\e[H\e[2J\e[3J"; //清空Terminal用
sCard deck[139];    //牌庫
sCard discardDeck[139]; //棄牌堆
int32_t deckIdx = 0;    //牌庫Index
```

```
int32_t discardDeckIdx = 0; //棄牌堆Index

//option var

int32_t bot_level = 1;  //電腦玩家Level，Level 1是基於隨機選擇而行動，Level 2則是在以隨機選擇為基礎下，綜合考量玩家現況再做出抉擇
int32_t end_condition = 12; //遊戲結束條件，需要有一名玩家的建築數量達到此數目，遊戲才可結束
int32_t game_version = 2;   //遊戲版本，可選擇遊玩初代版本或是擴充版版本

//cheat var
//此為作弊碼狀態記錄，在README中有更詳盡的描述

bool libcli = false;
bool noBonus = false;
bool neokent = false;
bool covid19 = false;
bool mask = false;

bool profession_table[6] = {0}; //記錄職業選擇與否之陣列
bool bank_used_table[5] = {0};  //記錄銀行被使用與否之陣列
int32_t price_table[5][6] = {{0, 1, 1, 1, 2, 2},    //價目表
                             {0, 1, 1, 2, 2, 2},
                             {0, 1, 1, 2, 2, 3},
                             {0, 1, 2, 2, 2, 3},
                             {0, 1, 2, 2, 3, 3}};
int32_t* price = NULL;  //記錄目前的交易價格
```

SanJuan.c

> 此檔案中的函式大多為功能性函式，與遊戲進行或與玩家進行動作有關

```
void setting();
//設定頁面
void global_var_init();
//初始化全域變數
void player_init(sPlayer* player, int32_t num_of_player);
//初始化玩家資訊 (i.e. sPlayer陣列)
void card_init(int32_t num, int32_t id, char* name, char* description, int32_t cost, int32_t score);
//初始化單張卡牌
void deck_init();
//初始化牌庫
void shuffle(int32_t num_of_card);
//洗牌
void draw(sPlayer* player, int32_t playerNum, int32_t num_of_card);
//抽牌
void discard(sPlayer* player, int32_t playerNum, sCard* target);
//棄牌
bool discard_with_instruction(sPlayer* player, int32_t num_of_player, int32_t playerNum, int32_t num_of_discard, char* afterError);
```

```
//包含文字指導的棄牌程序，回傳值為是否棄牌成功
void put_under_card(sPlayer* player, int32_t playerNum, int32_t
tablecardIdx, sCard* target);
//將手牌放至場上指定卡牌下
void PUC_with_instruction(sPlayer* player, int32_t num_of_player, int32_t
playerNum, int32_t tablecardIdx, char* afterError);
//同上，包含文字指導
void produce_product(sPlayer* player, int32_t playerNum, int32_t
tablecardIdx);
//生產貨物
void discard_product(sPlayer* player, int32_t playerNum, int32_t
tablecardIdx);
//丟棄貨物
void check_handcard_description(sPlayer* player, int32_t num_of_player,
int32_t playerNum, char* afterEnd);
//查看手牌敘述
void check_tablecard_description(sPlayer* player, int32_t num_of_player,
int32_t playerNum, char* afterEnd);
//查看場上卡牌敘述
void check_price_table(sPlayer* player, int32_t num_of_player, int32_t
playerNum, int32_t* price, char* afterEnd);
//查看銷售價目表
void distribute(sPlayer* player, int32_t num_of_player, int32_t governor);
//遊戲開始時，呼叫此函式進行發牌及棄牌
char* print_tablecard(sPlayer* player, int32_t num_of_player, int32_t
playerNum, int32_t tablecardIdx, int32_t type);
//給print.c中的table函式所使用，依輸入值回傳應印出的內容
int32_t choose_profession(sPlayer* player, int32_t num_of_player, int32_t
playerNum);
//呼叫此函式，讓玩家選擇職業
bool bot_decision(int32_t chance);
//依給定機率，隨機回傳結果之真假值的函式，用於電腦玩家的決策
void build(sPlayer* player, int32_t num_of_player, int32_t
playerNum_profession, int32_t playerNum);
//建造建築時呼叫此函式
void councillor(sPlayer* player, int32_t num_of_player, int32_t
playerNum_profession, int32_t playerNum);
//議員階段抽牌時呼叫此函式
void produce(sPlayer*player, int32_t num_of_player, int32_t
playerNum_profession, int32_t playerNum);
//生產貨物時呼叫此函式
void prospector(sPlayer* player, int32_t num_of_player, int32_t
playerNum_profession, int32_t playerNum);
//礦工階段抽牌或淘金時呼叫此函式
void trade(sPlayer* player, int32_t num_of_player, int32_t
playerNum_profession, int32_t playerNum, int32_t* price);
//銷售貨物時呼叫此函式
void round_start(sPlayer* player, int32_t num_of_player, int32_t governor);
//新的州長指派後呼叫此函式，檢查手牌及發動特定技能
void builder_phase(sPlayer* player, int32_t num_of_player, int32_t
playerNum_profession, int32_t playerNum_act);
//建築階段時呼叫此函式
```

```
void councillor_phase(sPlayer* player, int32_t num_of_player, int32_t
playerNum_profession, int32_t playerNum_act);
//議員階段時呼叫此函式
void producer_phase(sPlayer* player, int32_t num_of_player, int32_t
playerNum_profession, int32_t playerNum_act);
//生產階段時呼叫此函式
void prospector_phase(sPlayer* player, int32_t num_of_player, int32_t
playerNum_profession, int32_t playerNum_act);
//礦工階段時呼叫此函式
void trader_phase(sPlayer* player, int32_t num_of_player, int32_t
playerNum_profession, int32_t playerNum_act);
//交易階段時呼叫此函式
void reset_profession_table();
//重新初始化記錄職業被選擇與否的陣列
bool game_end(sPlayer* player, int32_t num_of_player);
//遊戲結束時呼叫此函式，進行vp總結算
void free_player(sPlayer* player, int32_t num_of_player);
//free player array
void free_card(sCard* card);
//free a single card
```

card.c

> 此檔案中的函式與卡片有關，包含卡片功能及尋找卡片等函式

```
int32_t find_tablecard(sPlayer* player, int32_t playerNum, int32_t
card_id);
//尋找該玩家是否場上擁有指定卡牌，並回傳該卡片之索引位置，若不存在則回傳0
sCard* find_handcard(sPlayer* player, int32_t playerNum, int32_t card_id);
//尋找該玩家手上是否擁有指定卡牌，並回傳該卡片之位置，若不存在則回傳NULL

//下方函式皆為尋找到該卡牌後才執行，函式名稱即為卡片名稱，因此不再贅述
void smithy(sPlayer* player, int32_t playerNum, sCard* target, int32_t*
fee);
void goldmine(sPlayer* player, int32_t num_of_player, int32_t playerNum);
void poor_house(sPlayer* player, int32_t num_of_player, int32_t playerNum);
void market_stand(sPlayer* player, int32_t num_of_player, int32_t
playerNum, int32_t num_of_product);
void archive(int32_t playerNum, int32_t* num_of_handcard_origin);
void black_market(sPlayer* player, int32_t num_of_player, int32_t
playerNum, int32_t* fee);
void trading_post(int32_t playerNum, int32_t* most_num_of_product);
void well(sPlayer* player, int32_t num_of_player, int32_t playerNum,
int32_t num_of_product);
int32_t crane(sPlayer* player, int32_t num_of_player, int32_t playerNum,
sCard* target);
//回傳欲改建的建築物索引值，若不使用起重機則回傳0
void chapel(sPlayer* player, int32_t num_of_player, int32_t playerNum,
int32_t tablecardIdx);
void aquaduct(int32_t playerNum, int32_t* most_num_of_product);
void carpenter(sPlayer* player, int32_t num_of_player, int32_t playerNum,
sCard* target);
void prefecture(int32_t playerNum, int32_t num_of_card, int32_t*
num_of_discard);
void market_hall(sPlayer* player, int32_t num_of_player, int32_t playerNum,
int32_t num_of_product);
void quarry(sPlayer* player, int32_t playerNum, sCard* target, int32_t*
fee);
void library(sPlayer* player, int32_t playerNum, int32_t* fee, int32_t
phase);
void guild_hall(sPlayer* player, int32_t playerNum);
void city_hall(sPlayer* player, int32_t playerNum);
void triumphal_arch(sPlayer* player, int32_t playerNum);
void palace(sPlayer* player, int32_t playerNum);
void office_building(sPlayer* player, int32_t num_of_player, int32_t
playerNum);
void cottage(sPlayer* player, int32_t num_of_player, int32_t playerNum);
void tavern(sPlayer* player, int32_t num_of_player, int32_t playerNum);
void bank(sPlayer* player, int32_t num_of_player, int32_t playerNum,
int32_t tablecardIdx);
void customs_office(sPlayer* player ,int32_t num_of_player, int32_t
playerNum, int32_t tablecardIdx);
void harbor(sPlayer* player, int32_t num_of_player, int32_t playerNum,
int32_t tablecardIdx);
void goldsmith(sPlayer* player, int32_t num_of_player, int32_t playerNum);
void residence(sPlayer* player, int32_t playerNum);
```

print.c

> 此檔案中的函式皆為輸出用函式

```
void error();
//錯誤訊息
void flush_buffer();
//清空stdin緩衝區
void notice();
//程式開始時，提醒玩家設置好遊戲環境
void menu();
//主選單
void about();
//顯示關於此專題之相關訊息
void choose_player();
//選擇玩家數量之介面
void table(sPlayer* player, int32_t num_of_player);
//印出目前場上情況，搭配sanJuan.c中的print_tablecard函式使用
void handcard(sPlayer* player, int32_t playerNum);
//印出玩家手牌
void handcard_part(sPlayer* player, int32_t playerNum, sCard* start);
//印出玩家部分手牌
void result(sPlayer* player, int32_t num_of_player);
//印出總排名
```

或許值得一提的Macro

```
#define NEW_PAGE printf("%s", clear)
//清空Terminal用
#define CARD(x, y) print_tablecard(player, num_of_player, (x), (y),
TYPE_CARD)
#define SUBC(x, y) print_tablecard(player, num_of_player, (x), (y),
TYPE_SUBCARD)
//提供給table函式使用，避免printf因參數過多且名稱過長，造成可讀性過低
#define find(x) find_tablecard(player ,playerNum, x)
```

遇到最困難的部分

- ### 描述

  在寫某些函式時，會突然意識到前面也有寫過同樣的功能，這時候就會下意識地想再包成一個函式

  於是乎會出現`discard_with_instruction`這類的函式

  雖然寫「好」之後，後續操作很方便，只要呼叫函式並處理回傳值即可

  但在初期沒有寫好的狀態，要一直考慮寫法是否可以通用，抑或是沒有起到「節省重複程式碼」的作用

- ### 解決方法

  時間砸下去，不斷地觀察與比對，找出最通用的寫法

  有效提高程式了可維護性及可讀性

  > 舉例來說，`discard_with_instruction`及`PUC_with_instruction`後皆有個參數叫`afterError`
  >
  > 此參數便是為了讓輸出錯誤訊息後，能再輸出不同的提示訊息
  >
  > 減少反覆出現的棄牌程序，也增加了程式的維護性(至少自己認為如此)

- ### 描述

  在製作專題時，我一直抱持著這支程式不僅要能正常運行

  還必須忠於現實桌遊的操作(像是卡片不能無緣無故消失)，以及要能夠給任何人遊玩

  為了將遊戲做的更人性化及容易理解，我花了很多時間在調整顯示的內容上

  而這也成為了我很大的困擾，因為每次試玩總是會發現不同的狀況

  > 舉例來說，每次遊玩遊戲時
  >
  > 棄牌前大多都會檢查一下卡片的敘述再棄牌，尤其是對卡片不熟悉的人
  >
  > 但我到最後試玩才發現，很多時候在進入棄牌階段前
  >
  > 都沒有讓玩家檢視手牌敘述的機會
  >
  > 直到最後我才加入此功能

- **解決方法**

  除了反覆遊玩，找出其不忠實於現實桌遊的部分

  由於在測試程式時，常常會有系統訊息過多，還沒看完就消失的情況

  因此我還需要在遊玩過程中進行錄影

  待遊戲告一段落後，再重新觀看影片

  檢查過程中是否有不合常理的情況出現

  這種方法在技術上的Debug上也十分有用