Lab of Object-Oriented Programming:

Assignment檢討

黃威、陳岳紘、邱彥翔

2022

使用 moodle 點名

請登入實習課的 moodle 課程

點擊出缺席並完成今日的點名

Agenda

- Assignment1
- Assignment2

程式碼的寫法並非唯一，以下列出的程式碼僅供參考

Assignment1

CardTest.cc

```
bool flag[52] = {false};
int RNumber = 0;
int RNumberArray[5] = {0};
```

flag[] 判斷數字是否已用過
RNumber 儲存隨機數字
RNumberArray，儲存 row_of_cards

```
        for(int i = 0; i < ncards; ++i){
            // 0 ~ 51
            RNumber = rand() % 52;

            // if RNumber already exist
            if(flag[RNumber]){
                --i;
                continue;
            }

            // put RNumber into RNumberArray
            RNumberArray[i % 5] = RNumber;

            // set flag true
            flag[RNumber] = true;

            // if row_of_cards or final_row
            if((i + 1) % 5 == 0 || i == ncards - 1){
                // cuz ArraySize: 1 ~ 5
                PassArray(RNumberArray, i % 5 + 1);
            }
        }
```

如果集滿一列或是跑到最後一輪時，準備印出

Generate Random Card(Advanced)

```cpp
#include <iostream>
#include <stdlib.h>
#include <iomanip>

using namespace std;

void shuffle(int *flag){
    int card, tmp;
    for(int i = 0; i < 52; ++i){
        card = rand() % 52;
        tmp = flag[i];
        flag[i] = flag[card];
        flag[card] = tmp;
    }
}

```

```cpp
int main(){
    int seed;
    cout << "seed = ";
    cin >> seed;
    srand(seed);

    int flag[52];
    for(int i = 0; i < 52; ++i)
        flag[i] = i;

    shuffle(flag);

    for(int i = 0; i < 52; ++i){
        cout << setw(2) << flag[i] << " ";
        if(i % 5 == 4)
            cout << endl;
    }
    return 0;
}
```

shuffle() 內可改寫為:

```cpp
#include <algorithm>
```

```cpp
void shuffle(int (&flag)[52]){    // reference to array, prevent array decay to pointer
    random_shuffle(std::begin(flag), std::end(flag));
}
```

Array of Pointers & Pointer to Array

- Array of Pointers:

  precedence: **[]** 先於 *****

```
int a[3] = {1, 3, 5};
int *ptr1[3] = {&a[0], &a[1], &a[2]};
```

本質是 array
裡面儲存 pointer

- Pointer to Array:

```
int arr[] = {10, 20, 30};
int (*ptr2)[3] = &arr;
```

本質是 pointer
指向 array

https://onlinegdb.com/5q8QYT7Jg

Try It !

shuffle
https://onlinegdb.com/YztEgF6tAQ

shuffle advanced
https://onlinegdb.com/qQ5jJLNML

Try replacing "reference to array" with "pointer to array"

Cards

```
18 const int kNPip=13;
19 const int kNSuit=4;
20 const int kNCards=52;
21 const int kCardWidth=11;
22 const int kCardHeight=11;
23 void PassArray(int*, int);
```

Cards.h

```
19 void PassArray(int *RNumberArray, int ArraySize){
20
21     int pipArray[ArraySize] = {0}, suitArray[ArraySize] = {0};
22
23     char suitCharArray[ArraySize] = {' '};
24
25     bool colorArray[ArraySize] = {0};
26
27     color colorPrint;
28
29     string suitStr = "SHDC";
30
```

Cards.cc

Cards.cc

```
31      for(int sizeNum = 0; sizeNum < ArraySize; ++sizeNum){
32          // 0 ~ 12
33          pipArray[sizeNum] = RNumberArray[sizeNum] / kNSuit;
34          // 0 ~ 3
35          suitArray[sizeNum] = RNumberArray[sizeNum] % kNSuit;
36          // 'S', 'H', 'D', 'C'
37          suitCharArray[sizeNum] = suitStr[suitArray[sizeNum]];
38          // 0 stands for black, 1 stands for red
39          if(suitCharArray[sizeNum] == 'S' || suitCharArray[sizeNum] == 'C')
40              colorArray[sizeNum] = 0;
41          else
42              colorArray[sizeNum] = 1;
43      }
```

决定牌的 pip, suit, color

Cards.cc

```
45      // loop Height
46      for(int i = 0; i < kCardHeight; ++i){      ... (1)
47          // loop ArraySize
48          for(int sizeNum = 0; sizeNum < ArraySize; ++sizeNum){      ... (2)
49
50              string widthStore;
51
52              // decide color
53              colorPrint = colorArray[sizeNum] ? red : black;
54
55              // first print CardWidth
56              for(int j = 0; j < kCardWidth + 1; ++j){      ... (3)
57                  // replace suitChar
58                  if(card[pipArray[sizeNum]][i][j] == 'x')
59                      widthStore += suitCharArray[sizeNum];
60                  // origin card
61                  else
62                      widthStore += card[pipArray[sizeNum]][i][j];
63              }
64
```

```
65              // string to const char*
66              const char* c = widthStore.c_str();
67              // print
68              AnsiPrint(c, colorPrint, white, false, false);
69              // Width space
70              cout << " ";
71          }
72          // Height newline
73          cout << endl;
74      }
```

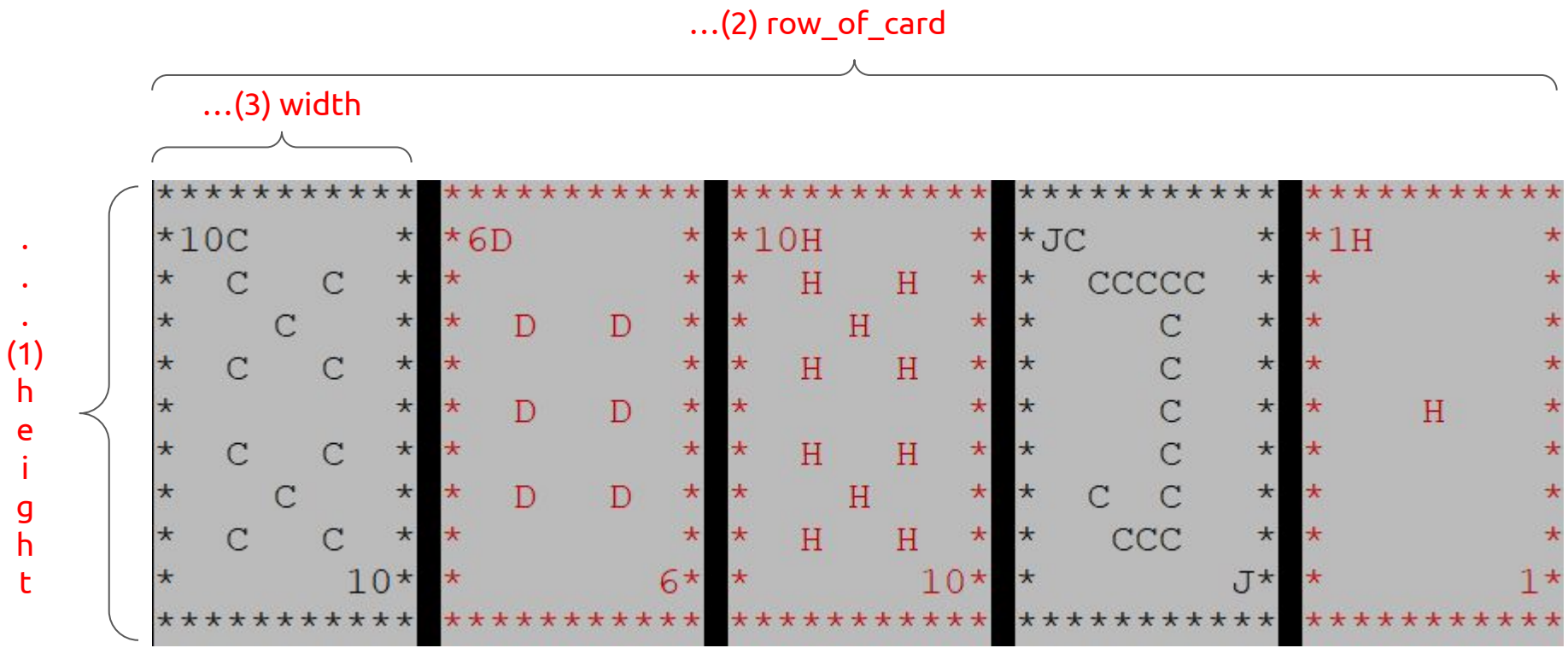
巢狀迴圈要從內到外進行解讀

...(3) 先遍歷 width

...(2) 再遍歷 row_of_card

...(1) 最後遍歷 height

Cards.cc


...(2) row_of_card
...(3) width
...(1) height
*********** *********** *********** *********** ***********
*10C   * *6D   * *10H   * *JC   * *1H   *
* C   C   * *   * * H   H * * CCCCC * *   *
*   C   * * D   D * *   H   * *   C   * *   *
* C   C   * *   * * H   H * *   C   * *   *
*   * * D   D * *   * *   C   * *   H   *
* C   C   * *   * * H   H * *   C   * *   *
*   C   * * D   D * *   H   * * C C   * *   *
* C   C   * *   * * H   H * *   CCC   * *   *
*   10* *   6* *   10* *   J* *   1*
*********** *********** *********** *********** ***********

Assignment2

SHPlayer - start()

```
60
61      bool flag[52] = {false};
62
63      SHPlayer shplayer("Player");
64
65      // put your code here
66
67      shplayer.start();
```

SHTest.cc

```
34 // initialize the number of cards to 0
35 void
36 SHPlayer::start()
37 {
38 // restart
39      kCurrentCards = 0;
40      showFirstCard = false;
41      totalCardPips = 0;
42 }
```

SHPlayer.cc

SHPlayer - addCard(), getNumCards()

```
69      while(shplayer.getNumCards() < 5){
70          // kNCards: 52
71          int tmp = rand() % Card::kNCards;
72          // check if used or not
73          if(!flag[tmp])
74              flag[tmp] = true;
75          else
76              continue;
77
78          // add card
79          shplayer.addCard(Card(tmp));
80      }
```

SHTest.cc

```
137 // return the number of cards at hand
138 int
139 SHPlayer::getNumCards() const
140 {
141     return kCurrentCards;
142 }
```

```
46 void SHPlayer::addCard(Card newCard)
47 {
48 // one more card
49
50     // protect array overflowing
51     if(kCurrentCards >= kMaxCards)
52         return;
53
54     // add to the current hand
55     cards[kCurrentCards] = newCard;
56
57     ++kCurrentCards;
58
59     // newCard.getPip(): 0 ~ 12
60     totalCardPips += (newCard.getPip() + 1);
61 }
```

count pips

SHPlayer.cc

Card.cc

```
// constructor
Card::Card(int newId)
{
    id = newId;
    // kNSuit: 4
    pip = id / kNSuit;
    suit = id % kNSuit;
}
```

```
int
Card::getID() const
{
    return id;
}

// get the pip of the card (0-12)
int
Card::getPip() const
{
    return pip;
}

// get the suit of the card (0-3)
int
Card::getSuit() const
{
    return suit;
}
```

SHTest.cc - First Card

```
82      // enough cards
83      if(shplayer.getNumCards() == kNCardPerRow){
84          // first card
85          if(showFirst)
86              shplayer.openFirstCard();
87          // print
88          shplayer.showCards();
89          // handtype
90          cout << "It's " << HandTypeName[shplayer.getHandPattern()] << endl;
91          // points
92          cout << "Total points: " << shplayer.totalPips() << endl;
93      }
```

SHPlayer - openFirstCard()

```
89          bool showFirstCard;
```

SHPlayer.h

```
63 // open the first card so it faces up
64 void
65 SHPlayer::openFirstCard()
66 {
67     showFirstCard = true;
68 }
```

SHPlayer.cc

SHTest.cc - Print Cards

```
82      // enough cards
83      if(shplayer.getNumCards() == kNCardPerRow){
84          // first card
85          if(showFirst)
86              shplayer.openFirstCard();
87          // print
88          shplayer.showCards();
89          // handtype
90          cout << "It's " << HandTypeName[shplayer.getHandPattern()] << endl;
91          // points
92          cout << "Total points: " << shplayer.totalPips() << endl;
93      }
```

SHPlayer.cc - showCards()

```
70 // print the current hand to the screen graphically
71 void
72 SHPlayer::showCards() const
73 {
74     for(int i = 0; i < kCardHeight; ++i){
75         for(int k = 0; k < kMaxCards; ++k){
76             // initialize str
77             string str = "";
78             for(int j = 0; j < kCardWidth; ++j){
79             // if first card and not want to show, print empty
80             if(k == 0 && !showFirstCard)
81                 str += card[13][i][j];
82             else if(card[cards[k].getPip()][i][j] == 'x')
83                 str += cardAbbrev[cards[k].getSuit()];
84             else
85                 str += card[cards[k].getPip()][i][j];
86             }
87             const char* c = str.c_str();
88
89             // decide color
90             if(cardAbbrev[cards[k].getSuit()] == 'S' || cardAbbrev[cards[k].getSuit()] == 'C')
91                 AnsiPrint(c, black, white);
92             else
93                 AnsiPrint(c, red, white);
94             cout << " ";
95         }
96         cout << endl;
97     }
98 }
99
```

```
22 // data for use with the card printing
23 const char cardAbbrev[] = "SHDC";
```

in CardPattern.h

SHTest.cc - Print HandType

```
82      // enough cards
83      if(shplayer.getNumCards() == kNCardPerRow){
84          // first card
85          if(showFirst)
86              shplayer.openFirstCard();
87          // print
88          shplayer.showCards();
89          // handtype
90          cout << "It's " << HandTypeName[shplayer.getHandPattern()] << endl;
91          // points
92          cout << "Total points: " << shplayer.totalPips() << endl;
93      }
```

SHPlayer - getHandPattern()

```cpp
// judge what kind of hand type you own
// you need to have 5 cards
HandType
SHPlayer::getHandPattern()
{
    // sort first
    this->sortCards();

    // judge handtype
    if(isStraightFlush())
        return STRAIGHT_FLUSH;
    if(isFourOfAKind())
        return FOUR_OF_A_KIND;
    if(isFullHouse())
        return FULL_HOUSE;
    if(isFlush())
        return FLUSH;
    if(isStraight())
        return STRAIGHT;
    if(isThreeOfAKind())
        return THREE_OF_A_KIND;
    if(isTwoPair())
        return TWO_PAIR;
    if(isOnePair())
        return ONE_PAIR;

    return OTHER;
}
```

```cpp
enum HandType {
    STRAIGHT_FLUSH,
    FOUR_OF_A_KIND,
    FULL_HOUSE,
    FLUSH,
    STRAIGHT,
    THREE_OF_A_KIND,
    TWO_PAIR,
    ONE_PAIR,
    OTHER
};
```

SHPlayer.h

SHPlayer.cc

```cpp
const char* HandTypeName[9] = {
    "Straight flush",
    "Four of a kind",
    "Full house",
    "Flush",
    "Straight",
    "Three of a kind",
    "Two pair",
    "One pair",
    "Other"
};
```

SHPlayer.cc - sortCards()

```
7 #include <algorithm>
```

```
144 bool compare(const Card& a, const Card& b){
145     // ascending
146     return a.getID() < b.getID();
147 }
148
149 void
150 SHPlayer::sortCards()
151 {
152     // 0 ~ 51 ascending
153     // sortedCards
154     for(int i = 0; i < 5; ++i){
155         sortedCards[i] = cards[i];
156     }
157     sort(sortedCards, sortedCards + kMaxCards, compare);
158 }
```

SHTest.cc - Print Points

```
82      // enough cards
83      if(shplayer.getNumCards() == kNCardPerRow){
84          // first card
85          if(showFirst)
86              shplayer.openFirstCard();
87          // print
88          shplayer.showCards();
89          // handtype
90          cout << "It's " << HandTypeName[shplayer.getHandPattern()] << endl;
91          // points
92          cout << "Total points: " << shplayer.totalPips() << endl;
93      }
```

SHPlayer.cc - totalPips()

```
// get the total points of the current hand
int
SHPlayer::totalPips() const
{
    return totalCardPips;
}
```

Any questions?