Lab of Object-Oriented Programming:

Class

黃威、陳岳紘、邱彥翔

2022

HW1 期限

- 繳交期限
  - 延長到 10/03 23:59:59
- 遲交期限不變

關鍵字

- 封裝(Encapsulation)、繼承(Inheritance)
  - 讓物件能夠重複利用
- 屬性(Property)、方法(Method)
  - 讓物件內的變數能夠受到保護

物件

- 用 `class` 定義物件(Object)
  - 屬性 - 物件內的變數，用名詞當作名稱
  - 方法 - 物件內的函式，用動詞當作名稱

```
class Car{
    int speed;
    void Run(){
        //...
    }
};
```

標頭檔

- .h 檔案
  - 物件和函式的宣告
  - 定義如何呼叫函式
- .cpp 檔案
  - 實作函式

存取修飾詞

- public
  - 可以直接(從任何地方)使用、呼叫
- private
  - 只能從 class 內部、friend class 使用、呼叫
- protected
  - 只能從內部、friend class、繼承的 class 使用、呼叫
- 沒有定義的話預設是 private

```
class Car {
  public:
   void public_method() {
        //...
   }

  private:
   void private_method() {
        //...
   }

  protected:
   void protected_method() {
        //...
   }
};
```

Friend Class

- 授權其他 class 存取自己的所有屬性、方法

```
#include <iostream>
using namespace std;

class Car {
   friend class CarFriend;

  private:
   int speed;

  public:
   Car() { speed = 70; }
   int GetSpeed() { return speed; }
};

class CarFriend {
  public:
   void ChangeCarSpeed(Car* car, int speed) {
       car->speed = speed;
   }
};

int main() {
   Car* BMW = new Car();
   CarFriend* CF = new CarFriend;
   CF->ChangeCarSpeed(BMW, 100);
   cout << BMW->GetSpeed() << endl;

   return 0;
}
```

範例

- public
  - 從外部直接修改屬性

```cpp
#include <iostream>
using namespace std;

class Car {
  public:
    int speed;
    void Turbo() {
        this->speed += 5;
    }
};

int main() {
    Car BMW;
    BMW.speed = 70;
    cout << "Current speed: " << BMW.speed << endl;
    BMW.Turbo();
    cout << "Current speed: " << BMW.speed << endl;

    return 0;
}
```

public

- 方便使用
  - 能夠從所有地方修改
- 不當存取的風險
  - 難以維護
  - 無法確保使用者遵照開發者設想的情況使用

封裝

- 保護物件內的屬性、方法
  - 使用 private、protected
  - 保護屬性不被直接修改
  - 使用方法對屬性進行修改
- 練習
  - https://onlinegdb.com/Shkrjpu-C

```
#include <iostream>
using namespace std;

class Car {
  private:
   int speed;

  public:
   void SetSpeed(int _speed) {
       //限制速度在60到120之間
   }
   int GetSpeed() {
       return speed;
   }
};

int main() {
   Car BMW;
   BMW.SetSpeed(70);
   cout << "Current speed: " << BMW.GetSpeed() << endl;
   BMW.SetSpeed(1000);
   cout << "Current speed: " << BMW.GetSpeed() << endl;

   return 0;
}
```

多載 (Overload)

- 相同名稱的函式對應不同型別的輸入
  - 編譯器會自動選擇對應的函式

```cpp
#include <iostream>
using namespace std;

class Car {
  private:
   int speed = 70;

  public:
   void Turbo() {
       speed += 5;
   }
   void Turbo(int dSpeed) {
       speed += dSpeed;
   }
   int GetSpeed() {
       return speed;
   }
};

int main() {
   Car BMW;
   BMW.Turbo();
   cout << "Current speed: " << BMW.GetSpeed() << endl;
   BMW.Turbo(10);
   cout << "Current speed: " << BMW.GetSpeed() << endl;

   return 0;
}
```

建構式 (Constructor)

- 在物件建立時定義物件的初始狀態
  - 不應指定傳回型別，亦不需傳回 值
  - 名稱和類別名稱相同
  - 建立物件時會自動呼叫

```
#include <iostream>
using namespace std;

class Car {
  private:
   int speed;

  public:
   Car() {
       speed = 70;
   }
   Car(int iSpeed) {
       speed = iSpeed;
   }
   int GetSpeed() {
       return speed;
   }
};

int main() {
   Car BMW;
   cout << "BMW speed: " << BMW.GetSpeed() << endl;
   Car BMW_2(100);
   cout << "BMW_2 speed: " << BMW_2.GetSpeed() << endl;

   return 0;
}
```

解構式 (Destructor)

- 刪除物件時使用
  - 檔案關閉
  - 釋放記憶體
  - 物件為指標時能使用 `delete` 刪除

```cpp
#include <iostream>
using namespace std;

class Car {
  private:
   int speed;

  public:
   Car() {
       speed = 70;
   }
   int GetSpeed() {
       return speed;
   }
   ~Car() {}
};

int main() {
   Car* BMW = new Car();
   cout << "BMW speed: " << BMW->GetSpeed() << endl;
   delete BMW;

   return 0;
}
```

Exercise 3

a153: 超人！不要再打啦！

內容 ：
有一天，你偶遇有一群超人在打架，請幫忙紀錄發生了什麼

超人的個資如下

category 種類

name 姓名

hp 血量

atk 攻擊值

def 防禦值

每個超人會遇到一個 event 事件

事件代號

A 超人發動攻擊技能　　發出的攻擊值=（atk+hp）/2

B 被對手攻擊 （ 輸入對手的攻擊值be_atk_value）　對超人造成的傷害=2*（be_atk_value）/def

C 展示當前hp值

Exercise 3

輸入說明

輸入n筆超人的個人資料

ex.

category種類 :SpiderMan

name姓名 :Peter

hp血量 :95

atk攻擊值 :25

def防禦值 :20

遭遇的事件 :B（對手的攻擊值=20）

不同種類可以有相同的名字

輸入錯誤字母沒有輸出

輸入數值若為負數，請變成0

輸出說明

輸出該超人遭遇的事件

ex

A 發動攻擊技能：IronMan Edward caused 60 points damage

B 被對手攻擊：SpiderMan Peter is damaged by 2 points

C 展示當前hp值：CaptainAmerica Steven currently has 89 points of HP

範例輸入

```
IronMan Edward 100 20 25 A

SpiderMan Peter 95 25 20 B 20

CaptainAmerica Steven 89 30 22 C
```

範例輸出

```
IronMan Edward caused 60 points damage

SpiderMan Peter is damaged by 2 points

CaptainAmerica Steven currently has 89 points of
```

補充說明

- 可參考右邊框架
- 請使用物件導向

```
#include <iostream>
using namespace std;

class Superman {
  private:
    string category;
    string name;
    int hp;
    float atk;
    float def;

  public:
    Superman() {
        category = "";
        name = "";
        hp = 0;
        atk = 0;
        def = 0;
    }
    ~Superman();

    int Attack();
    int Attacked(int damage);
    int ShowHP();
};

int main() {
    return 0;
}
```

Assign2

Assign2

Due: **10/16+3/2022 (三) 23:59:59 (+3: National Day)**

遲交一天扣該 Assign 分數 **10%**

Assign2 最遲繳交日: **10/23+3/2022 (三) 23:59:59**

超過最遲繳交日所繳交的 Assign 分數以 **0 分** 計算

Assign2

Topic: Poker game player simulation

Objective: Basic class constructs (basic constructor, accessor, mutator, access control, and encapsulation)

Description:

In this assignment, we will use C++ classes to simulate player in a poker (Soha) game. Many functions need to be implemented in order to simulate a complete poker game. However, in this assignment we ONLY simulate the player part of the game. There is more than one way to play a poker game but the basic rules for counting points for a hand of cards are the same. Each user is given a set of 5 cards, called a hand. The face values (*pips*) and patterns of the five cards determine the order of a hand of cards. For example, a pattern consisting of two cards of the same pip is called a pair. A pattern of cards with five consecutive pips is called a straight.

Assign2

Description:

You have two classes to implement in this simulation: Card and SHPlayer. The Card class encapsulates(封裝) what a regular card has in a poker game: pip and suit (or aunique ID number for a combination of pip and suit). A user of the Card class calls its member functions to query and set card ID's or other attributes. The SHPlayer class encapsulates what a poker player needs to have. For example, a player needs to keep track of a hand of 5 cards given one by one. You also need to check for overflows on array indices.

The specifications of the public interfaces for these two classes are given to you in the header files. You are asked to add necessary data and implement member functions for the given public interface. Read the comments in the class declaration for details of these functions. Not every member functions will be used in this assignment but you are still asked to implement them for completeness of the classes.

Assign2

Description:

To test these two classes, your program should draw 5 cards and print them to the screen in the way we did in the first assignment except for that the first card may not be printed with its face up according to command-line arguments. In addition, print the sum of the card values to the screen as well. (Ace is counted as one.)


```
16 const char* HandTypeName[9] = {
17     "Straight flush",
18     "Four of a kind",
19     "Full house",
20     "Flush",
21     "Straight",
22     "Three of a kind",
23     "Two pair",
24     "One pair",
25     "Other"
26 };
```

Assign2

Assignment Directory: /usr/local/class/oop/assign/assign2

```
> cp -r /usr/local/class/oop/assign/assign2 .      # 複製作業 2 至當前目錄
> cd assign2                                       # 進入作業 2 資料夾
> make                                             # 編譯
> make clean                                       # 清除編譯內容
> /usr/local/class/oop/bin/submit 2                # 繳交作業 2
```

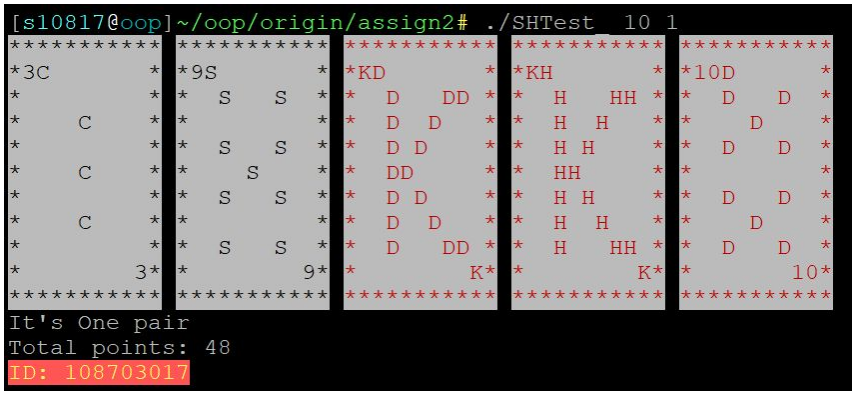
Sample Output:

A sample output from testing the classes is given as follows. A sample executable can be found in the assignment directory. The program takes two optional command-line arguments, which are a seed for the random number generator and a flag indicating whether to print the first card facing up or down.

Assign2

Sample Output:

A sample output from testing the classes is given as follows. A sample executable can be found in the assignment directory. The program takes two optional command-line arguments, which are a seed for the random number generator and a flag indicating whether to print the first card facing up or down.



Assign2

Files in the Assignment Directory:

- **Makefile**: a sample Makefile
- **SHTest.cc**: a testing program. You need to modify the PrintMyID function. You also need to add testing code such that the program produces the same result as shown in the sample output. (We provide partial code.)
- **Cards.h:** a sample header file
- **AnsiPrint.cc** and **AnsiPrint.h**: the AnsiPrint functions from assignment #1. (We provide.)
- **CardPattern.h**: an array of skeletons for card patterns. This array is the same as in the previous assignment. (We provide.)
- **Card.h** and **Card.cc**: the class for a poker card. The specification is given in the Card.h file. You need to define the private part of the class and implement all necessary member functions in the Card.cc file. (We define and you write.)
- **SHPlayer.h** and **SHPlayer.cc**: class for a poker player. The specification is given in the header file. You need to define the private part of the class and implement all necessary member functions in the SHPlayer.cc file. Specifically, you are asked to write a method that can sort the cards. To reduce your load, all the implementation for determining card patterns except for the flush straight pattern is given. (We define and you write partially.)

Assign2

What to hand in:

You need to hand in a complete version of your source code electronically as in the previous assignments.

Implementation Notes:

You can feel free to design your own data structure or internal functions as long as the interface/public functions adhere to the specification. You are fully responsible for maintaining the consistency of the data in your class. For example, you are responsible for checking if an array index is out of bound and all data members are appropriately initialized. In addition, to making grading easier, please do not add extra I/O's to your program. For example, please DO NOT add code to query options from the user or to output your results to a file.

Assign2 - 評分標準

項目	配分
有交（含屍體）	20
成功Compile（無Error）	10
亂數處理	10
5筆測資，每筆測資配分	可以覆蓋第一張牌:3分 牌型判斷正確:3分 點數判斷正確:3分 正確印牌(含顏色):3分