國立台灣海洋大學資訊工程系 C++ 程式設計 期中考試題解答

姓名：_____ 系級：_____ 學號：_____

97/04/15

考試時間：**10:00 – 12:00**

試題敘述蠻多的，看清楚題目問什麼，針對重點回答是很重要的；你知道文字也是一種語言，我把想要問你的問題轉換成語言，你再轉換成所了解的去執行，這中間很容易有表達或是轉換的誤差 – 就和你寫程式會有 *bug* 一樣，不確定的請一定要提出問題；我唯一能夠做的就是把題目的敘述變長一點，希望你在看的時候如果想的和我想的不一樣時會有一些不一致的敘述 (*assertion error*)；總分有 *160*，超過 *140* 的部份不計分，請看清楚每一題所佔的分數再回答

考試規則：1. 不可以翻閱參考書、作業及程式
2. 不可以使用任何形式的電腦 (包含計算機)
3. 不可以左顧右盼、不可以交談、不可以交換任何資料、試卷題目有任何疑問請舉手發問 (看不懂題目不見得是你的問題，有可能是中英文名詞的問題)、隔壁的答案可能比你的還差，白卷通常比**錯得和隔壁成績好的同學一模一樣**要好很多
4. 提早繳卷同學請直接離開教室，不得逗留喧嘩
5. **違反上述任何一點之同學一律送請校方處理**
6. 請在答案卷上標示題號回答，繳卷時請繳交***簽名過之試題卷及答案卷***

1. 請參考右圖程式回答下列問題 [95] (請注意引入適當的定義檔案)

```
1.  #include <iostream>
2.  #include <iomanip>
3.  using namespace std;
4.  struct Employee
5.  {
6.      char *name;
7.      double payRate;
8.      double workHours;
9.  };
10. double calcNetPay(Employee);
11. int main()
12. {
13.     Employee employee = {"Bob", 8.93, 40.5};
14.     double netPay;
15.     netPay = calcNetPay(Bob);
16.     cout << "The net pay for " << employee.name
17.          << " is $ " << netPay <<endl;
18.     return 0;
19. }
20. double calcNetPay(Employee employee)
21. {
22.     return employee.payRate * employee.workHours;
23. }
```

a. [5] 物件導向程式中整個程式的功能是由許多物件合作來完成的，請問如果有一些物件的基本功能是相同的 (例如公司裡有很多的員工)，在 C++ 中我們用什麼語法來描述一整類的物件?

Sol:

類別 (class), 例如

```
class Employee
{
  …
};
```

請運用這學期學到的方法，改寫右圖中的程式，讓它基於物件來完成同樣的工作?

b. [5] 首先是設計員工的類別，請問在員工類別裡你會設計哪些資料成員呢?

c. [5] 請在這個類別裡設計 calcNetPay() const 成員函式?

d. [5] 請在這個類別裡設計 printNetPay() const 成員函式?

e. [10] 請運用初始化串列設計建構元函式以便初始化物件的內容? (建構元函式的參數有三個，例如 "Bob", 8.93, 40.5，請以 new 配置字元陣列所需要的記憶體存放員工的名字)

f. [5] 請設計解構元函式與其內容?

g. [5] 請改寫 main() 函式，運用上述員工類別以及其物件來完成 11-19 列的工作?

(b. c, d, e, f, g 請合併撰寫程式碼回答)

Sol:

```
---- employee.h ----
1.  #ifndef EMPLOYEE_H
2.  #define EMPLOYEE_H
3.  #include <iostream>
4.  class Employee
5.  {
6.  public:
7.          Employee(char *name, double rate, double hour);
8.          ~Employee();
9.          void printNetPay() const;
10.         void printNetPay(std::ostream &os) const;
11. private:
12.         double calcNetPay() const;
13.         char *m_name;
14.         double m_payRate;
15.         double m_workHours;
16. };
17. #endif
---- employee.cpp ----
18. #include "employee.h"
19. double Employee::calcNetPay() const
20. {
21.         return m_payRate * m_workHours;
22. }
```

```
23. void Employee::printNetPay() const
24. {
25.         cout << "The net pay for " << m_name
26.                 << " is $ " << calcNetPay() <<endl;
27. }
28. Employee::Employee(char *name, double rate,
29.                                         double hour)
30.         : m_name(new char[strlen(name)+1]),
31.           m_payRate(rate), m_workHours(hour)
32. {
33.         strcpy(m_name, name);
34. }
35. Employee::~Employee()
36. {
37.         delete m_name;
38. }
---- main.cpp
39. #include "employee.h"
40. void main()
41. {
42.         Employee employee("Bob", 8.93, 40.5);
43.         employee.printNetPay();
44. }
```

h. [5] 請問 g. 的答案中何時建構元函式會被執行到，何時解構元函式會被執行到?

Sol:

Ctor: line 42, Dtor: line 44 (dtor will be inserted right before each “return” statement for a function with many return statements)

i. [5] 請問上面的答案裡哪些部份要放在 .h 檔案，哪些要放在 .cpp 檔案中?

Sol:

在上圖中標示為 employee.h, employee.cpp 以及 main.cpp。

j. [5] 請解釋什麼狀況下會發生重複引入 .h 檔案，如何以前處理器指令避免重複引入?

Sol:

假設有一個 a.h 檔案中有 #include "employee.h" 的敘述, main.cpp 中如果同時有 #include "a.h" 以及 #include "employee.h" 的敘述就會發生重複引入的問題，此時 compiler 會發現類別 Employee 定義了兩次，解決的方法就是如上圖 line 1, line 2, line 17 使用 #ifndef, #define, #endif 的前處理器指令來避免。

k. [5] 請問 C++ 中的物件和變數在概念上和使用上有什麼相同的地方? 有什麼不同的地方?

Sol:

相同處: 都需要使用一個型態來定義以後才能夠使用，都會配置適當大小的記憶體來存放。

不同處: 物件有封裝的機制，需要透過物件的介面來操作，變數只是資料存放的地方，任何看得到這個變數的程式碼都可以自由運用這個變數。

l. [5] 請問為什麼要設計 printNetPay() const 成員函式? 不設計這個函式而在主程式裡直接列印的話會有什麼問題?

Sol:

在主程式 main() 中基本上是沒有辦法直接存取 employee 物件的私有資料成員 (data member) 的，除非資料成員的存取權限是公開的 (public)，但是這樣子就嚴重地破壞了物件的封裝。

m. [5] 請問上題中 const 代表的意義為何?

Sol:

代表 Employee 類別的成員函式 printNetPay() 不會去更動目標物件的狀態 (任何資料成員的內容)，例如在上圖中 line 43 執行完畢之後，可以確定 employee 物件的內容和執行前是完全一樣的。

n. [5] 請問 calcNetPay() 成員函式設計為 public 和設計為 private 各有什麼設計上的好處?

Sol:

基本上這個題目裡並沒有辦法看出絕對要用 public 的 calcNetPay() 或是用 private 的 calcNetPay()，封裝物件時界面函式的設計原則是 “界面越簡單越好”，刪除不必要的界面，或是至少等到需要用時再換成公開的界面。

o. [5] 請修改 printNetPay() 函式的參數, 用參考變數將 cout 當成參數傳進函式, 並修改呼叫 printNetPay() 的程式? (請注意引入檔案) 請解釋如果不用參考變數會有什麼問題? [5]

Sol:

類別宣告裡 employee.h 增加

```
#include <iostream>
using namespace std;
…
void printNetPay(ostream &) const;
```

employee.cpp 檔案裡增加

```
void Employee::printNetPay(ostream &os) const
{
      os << "The net pay for " << m_name
          << " is $ " << calcNetPay() <<endl;
}
```

main.cpp 裡增加

```
employee.printNetPay(cout);
```

p. [5] 請再修改 printNetPay() 函式的參數型態，允許傳入輸出檔案串流 ofstream 的物件，並且開啟一個 netpay.txt 檔案，呼叫 printNetPay(…) 將資料寫入檔案中?

Sol:

類別宣告裡 employee.h 增加

```
#include <fstream>
using namespace std;
…
void printNetPay(ofstream &) const;
```

employee.cpp 檔案裡增加

```
void Employee::printNetPay(ofstream &ofs) const
{
```

```
    ofs << "The net pay for " << m_name
        << " is $ " << calcNetPay() <<endl;
}
```

main.cpp 裡增加

```
ofstream ofs("netpay.txt");
employee.printNetPay(ofs);
```

q. [5] 請定義全域的 operator<<() 函式擴充 iostream 的功能，使得員工類別的客戶程式可以運用 cout << employee; 列印出員工的姓名和薪資?

Sol:

類別宣告裡 employee.h 增加

```
ostream &operator<<(ostream &, Employee &);
```

employee.cpp 檔案裡增加

```
ostream &operator<<(ostream &os, Employee &emp)
{
    emp.printNetPay(os);
}
```

2. 請參考右圖回答下列問題 [45]

```
1.  class Complex
2.  {
3.  public:
4.      void setValue(double, double);
5.      void add(Complex rhs);
6.      bool compare(Complex rhs);
7.  private:
8.      double m_real;
9.      double m_imaginary;
10. };
11. void Complex::setValue(double x, double y)
12. {
13.     m_real = x; m_imaginary = y;
14. }
15. void Complex::add(Complex rhs)
16. {
17.     m_real += rhs.m_real;
18.     m_imaginary += rhs.m_imaginary;
19.     return;
20. }
21. bool Complex::compare(Complex rhs)
22. {
23.     …
24. }
```

a. [10] 請舉例解釋什麼是 this 指標? (有什麼作用?)

Sol:

this 是一個常數指標，在透過一個物件呼叫一個非 static 的成員函式時 (例如： employee.printNetPay())，employee 物件的位址 (&employee) 會被隱藏地傳入函式中，此位址也就是 this 的內容；通常透過這個指標有三種功能：1. 存取物件中一些名稱和成員函式區域變數相同的資料成員，this->imaginary；2. 比對傳入的物件是否和本身相同，if (&rhs == this)；3. 存取整個物件，例如 return *this;

b. [5] 請運用 add() 成員函式設計一個 subtract() 界面? (請說明透過 add() 函式實作的好處?)

Sol:

```
void Complex::subtract(Complex rhs)
{
    rhs.m_real = -rhs.m_real;
    rhs.m_imaginary = -rhs.m_imaginary;
    add(rhs);
}
```

盡量透過已經有的函式來實作可以降低測試的成本，同時也避免造成程式維護時維持一致性的困難。

c. [5] 請完成 compare() 成員函式，比較傳入的物件 rhs 裡的複數是否和自己這個物件是相同的，請自行設定一個比對的精確度?

Sol:

```
#include <math.h>
bool Complex::compare(Complex rhs)
```

```
{
    double dist = (m_real – rhs.m_real)*(m_real – rhs.m_real);
    dist += (m_imaginary – rhs.m_imaginary)*(m_imaginary – rhs.m_imaginary);
    dist = sqrt(dist);
    precision = 1e-8;
    return (dist < precision);
}
```

d. [5] 請由 Complex 物件使用者的角度判斷 compare() 這個函式名稱是否恰當? 該如何修改?

Sol:

一個成員函式的名稱基本上是一個動作，所以一定是一個動詞，要決定一個成員函式的名稱必須由這個函式使用的環境來決定，例如：

```
Complex x, y
if (x.compare(y)) …   // 這個敘述就不太有意義
if (x.equals(y)) … // 這個敘述比較好
if (x.isEqualTo(y)) … // 這個有 be 動詞也不錯
```

e. [10] 請定義一個靜態的 unitTest()單元測試函式，裡面運用 assert() 測試 add() 界面函式? 請撰寫 main() 函式呼叫 unitTest()? (請引入適當定義檔案)

Sol:

類別宣告裡 complex.h 增加

```
…
static void unitTest();
…
```

complex.cpp 檔案裡增加

```
#include <assert.h>
void Complex::unitTest()
{
    Complex x1, x2, x3;
    x1.setValue(1.0,2.0);
    x2.setValue(1.9,-1.5);
    x3.setValue(2.9, 0.5);
    x1.add(x2);
    assert(x1.equals(x3));
}
```

main.cpp 裡增加

```
#include "complex.h"
…
Complex::unitTest();
```

```
1.  Complex counter, offset;
2.  counter.setValue(0.0, 0.0);
3.  offset.setValue(1.0, 1.0);
4.  for (int i=0; i<100000; i++)
5.      counter.add(offset);
```

f. [10] 假設在某一個 Complex 的成員函式中有右圖的程式片段，如果在測試這個功能時發現程式執行的速度有點慢，請問可以用什麼語法，在不修改程式邏輯的情況下，調整程式執行的速度? (請說明原因?)

Sol:

在不修改程式邏輯的情況下，可以先把 Complex::add() 函式調整為 inline 函式，由於 compiler 把看得到原始碼的 inline 函式完全展開在呼叫這個函式的地方，可以省去參數 push 到系統堆疊，參數拷貝，控制權移轉等等時間，在函式不長的情況下會有很大的改進。

3. [10] 在下圖中第 8 列和第 14 列是一模一樣的敘述，請問由程式設計者的角度，哪一個用法是對的? (請解釋原因)

```
1.  #include <assert.h>
2.  #include <cstdio>
3.  using namespace std;
4.  void someFunction(FILE *fp)
5.  …
6.  FILE *fp;
7.  fp = fopen("input.txt", "r");
8.  assert(0 != fp);
9.  …
10. someFunction(fp);
```

```
11. …
12. void someFunction(FILE *fp)
13. {
14.         assert(0 != fp);
15.         …
16. }
```

Sol:

兩個敘述都一樣是 assert(0 != fp); 都在檢查 fp 是否為 0 (NULL)，但是實際上由程式設計者的角度來看，一個是好的用法，一個卻是不好的用法：前者是不好的的，後者則是合理的用法，原因在於第 7 列使用 fopen() 函式去開啟檔案時，本來就有很多狀況可能導致開啟失敗，因此回傳值有很多機會是 0，也就是回傳 0 時是常常發生的，可以說是正常的狀況，程式的邏輯應該要能夠處理這種狀況，不能夠裝死，看到 0 程式就直接掛點，平常的處理方法也許重新要求使用者輸入別的檔案名稱，也許關閉一些檔案再嘗試開檔，也許要求使用者檢查檔案系統，也許…反正就是不能直接出現 assert 錯誤；第 14 列在 someFunction() 中撰寫這個函式的人要求“呼叫這個函式的人”傳入一個已經開啟的檔案指標，因為是程式設計者和程式設計者之間的約定，如果呼叫端不按照這個約定來做，自然需要修改呼叫端程式，沒有什麼可以通融轉圜的，所以可以用 assert 檢查，如果違反的話程式立刻終止，要求程式設計者修改呼叫端程式。簡單的原則就是 assert 錯誤不是給“使用者”看的，是給“程式設計的人”看的，被使用者看到的 assert 錯誤一定是不好的。

4. [10] 作業題: 請手動將有理數 235 / 93 轉換為連續分數?

Sol:

235 / 93 = 2 + 49 / 93

93 / 49 = 1 + 44 / 49

49 / 44 = 1 + 5 / 44

44 / 5 = 8 + 4 / 5

5 / 4 = 1 + 1 / 4

$$\frac{235}{93}=2+\frac{1}{\frac{93}{49}}=2+\frac{1}{1+\frac{44}{49}}=2+\frac{1}{1+\frac{1}{1+\frac{5}{44}}}=2+\frac{1}{1+\frac{1}{8+\frac{4}{5}}}=2+\frac{1}{1+\frac{1}{8+\frac{1}{1+\frac{1}{4}}}}$$

i.e. 235 / 93 = [2;1,1,8,1,4]

星期四早上 11:00-12:00 請至 INS 201 及 INS 203 demo 作業二, 如果在此時段無法 demo 而需要在其它時段, 請與我或是助教預約

釐清、正視自己學習路程上遇見的問題,運用所有的資源(老師、助教、學長、同學、email, MSN, 討論區、實習課、Office Hour、…)尋找問題的答案，就是一種積極進取的負責態度!! 這種態度是需要練習才能擁有的，不是與生俱來的。

我的 Office Hour 是什麼時候? 星期二、四早上上課前，星期二、四中午，星期二下午，星期三下午

學校裡容許你不斷地在練習的過程裡失誤，不要因為這種失誤就停止你的練習與追尋

一份你不太瞭解的考卷不是為了打擊你的信心，而是為了讓你由各種角度來檢視自己學習的成果，離開學校以後失敗的學習通常會伴隨你不願意承受的後果