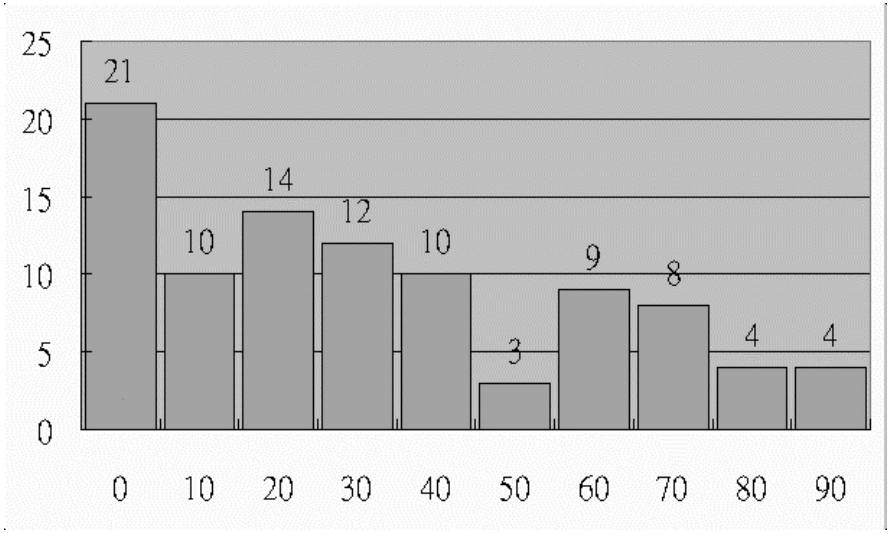
932 Midterm Exam Discussions

C++ Object Oriented Programming
Pei-yih Ting
94/04 NTOU CS

Score Statistics



Average: 42

Problem #1

✧ What are objects in a C++ program?

"物件有自己私有的狀態、有行為能力、負責提供某些指定功能的軟體單元"

用我們已經談過的 OO 性質來描述:

"物件可以保存資料, 但是*封裝*得很好,
只有透過適當的操作*介面*才能夠得到它的服務"

儲存 + 封裝 + 抽象化 + 介面

Problem #2

✧ Part A:

這一題的重點當然是 Stack() 建構元函式和 ~Stack() 解構元函式的呼叫, 程式裡面看不到, 但是你在用 C++ 寫程式時應該要隨時想到

✧ Part B:

描述高階程式功能的時候要把程式的表現描述清楚, 實際上用什麼變數, 哪裡加一, 哪裡迴圈, 哪裡刪除記憶體就不重要了: 這個程式片斷主要由 in 輸入串流中讀取一列一列的資料, 然後反序在螢幕上列印出來

這也是抽象化

Problem #3

- Part A: MyClass::printCounts()
- Part B: MyClass.cpp

  int MyClass::objectCounts = 0;
- Part C:

  類別內 static 變數的性質為：

  這個變數是所有此類別物件共享的, 唯一的一個變數, 這個變數所使用的記憶體不是在產生物件時配置的, 也不會在任何一個物件解構時刪除, 它的記憶體是如上一題所示, 在 cpp 檔案中另外定義的,

  這種變數是用來儲存所有此類別物件通用的屬性, 或是記錄一些共通的常數

Problem #4

```
01. char *buf1;
02. buf1 = "12345";
03. strcpy(buf1, "12345");
04. string buf2;
05. buf2 = "12345";
06. strcpy(buf2, "12345");
07. char buf3[10];
08. buf3 = "12345";
09. strcpy(buf3, "12345");
```

- 編譯錯誤:

  06. strcpy(buf2, "12345");
  strcpy cannot convert parameter 1 from string to char *

  08. buf3 = "12345";
  '=' cannot convert from char[6] to char[10]
- 執行錯誤

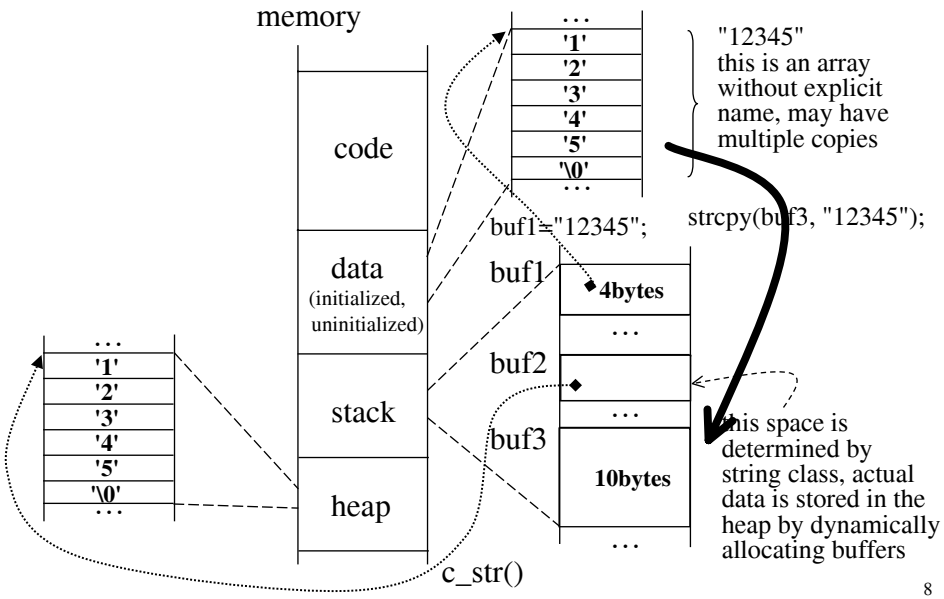
  03. strcpy(buf1, "12345");

  此列敘述在很多系統上都會造成執行的錯誤, 有的系統會說不允許寫入常數資料區段, 如果新的字串長度超過原來的長度的話, 更容易發生 BOF 的錯誤

Problem #4 (cont'd)

- 所有在高階語言程式裡出現的非 0 常數, 編譯器都需要替它配置一塊記憶體來存放, 一個非 0 的數字不可能在執行的時候直接出現在暫存器中, 一定要從記憶體中移入暫存器, 然後才能進行運算或是處理。
- 除了一些可以放進組合語言程式碼的常數之外, 每一個非 0 常數 (包括字串常數) 都需要放在資料區段裡, 例如： 字串常數 "12345" 在資料區段裡就佔有 6 個位元組, 看起來像是個字元陣列
- 對於編譯器而言, 字串常數 "12345" 是一個 char [6] 型態的陣列
- 第二列 buf1 = "12345"; 執行後 buf1指標變數裡存放著字串常數 "12345" 的記憶體位址, 第三列 strcpy(buf1, "12345") 強制拷貝資料到該字串常數所在的記憶體處

Problem #4 (cont'd)



Problem #5

✧ 建構元函式

```
Car::Car(int numCylinders, int displacement)
    : m_engine(numCylinders, displacement)
{
}
```

✧ 其它在此處執行時有可能會對的寫法

```
1. Car::Car(int numCylinders, int displacement)
   {
      Engine engine(numCylinders, displacement);
      m_engine = engine;
   }
```

正確的條件: 1. Engine 類別內的 assignment operator 可以配合
2. 多解構一次不會造成問題
3. Engine 類別需要有 default ctor

Problem #5 (cont'd)

```
2. Car::Car(int numCylinders, int displacement)
   {
      m_engine = Engine(numCylinders, displacement);
   }
```

正確條件：和方法 1 相同

```
3. Car::Car(int numCylinders, int displacement)
   {
      Engine *ptr = new Engine(numCylinders, displacement);
      m_engine = *ptr;
      delete ptr;
   }
```

正確條件：和方法 1 相同

不論是方法 1, 2, 或是 3, 就算是所有正確的條件都符合,其實還是多此一舉, 你的程式基本上對於兩個 Engine 物件個別作了初始化的動作, 多執行了一次設定的動作, 還有一次解構的動作

Problem #6

✧ Part A:

透過 accessor 成員函式直接把類別的資料傳出去, 通常是一種破壞封裝的作法, 另外由於像 print 這樣子的功能需要存取很多類別內的狀態變數而且常常會用到, 應該要由物件自己負責, 不應該推諉責任, 這是分工的錯誤, 破壞物件分工的基本原則

✧ Part B:

```
ostream &operator<<(ostream &out, const Fraction &obj)
{
        obj.print(out);
        return out;
}
```


Problem #7

```
01. ifstream infile("vector.dat");  // 開啓檔案串流
02. Vector vectA(infile);   // 由檔案串流中初始化向量 A
03. Vector vectB(infile);   // 由檔案串流中初始化向量 B
04.
05. double ip;                                  // 計算向量 A 與
06. ip = vectA.innerProduct(vectB);  // 向量 B的內積
07. cout << "The first vector A is " << endl;
08. vectA.print(cout);          // 列印 A 向量的內容
09. cout << "The second vector B is " << endl;
10. vectB.print(cout);          // 列印 B 向量的內容
11. cout << "The inner product of A and B is " << ip << endl;
```

✧ 題目給的這段程式並沒有直接指明 Vector 這個類別的介面 (public 的成員函式), 但是有示範性地使用 vectA 及 vectB 兩個 Vector 類別的物件, 由此可以看到 Vector 類別最少應該提供哪些成員函式 (如黃字所示)

Problem #7 (cont'd)

- Part A, B

```
class Vector {
public:
   Vector(ifstream &);
   ~Vector();
   double innerProduct(const Vector &) const;
   void print(ostream &) const;
private:
   int m_dim;
   double *m_data;
};
```

- Another version of part B (space are also dynamically allocated)

```
private:
   int m_dim;
   vector<double> m_data;
};
```


Problem #7 (cont'd)

- Part C

```
Vector::Vector(ifstream &is) {
   is >> m_dim;
   // you can check the range of m_dim
   // if m_dim is some illegal value, raise exception
   m_data = new double[m_dim];
   for (int i=0; i<m_dim; i++)
      is >> m_data[i];
}
```

如果你的 m_data 是 vector<double> 的話, 迴圈的內容換成

```
for (int i=0; i<m_dim; i++) {
   is >> tmp;
   m_data.push_back(tmp);
}
```

由 ifstream infile("vector.dat"); 可以知道資料是ASCII文字, 不過如果你把它當成二進位格式讀取的話, 讀取的方法如下

```
is.read((char *) &m_dim, sizeof(int));
is.read((char *) &m_data[i], sizeof(double));
```


Problem #7 (cont'd)

- Part D

```
double Vector::innerProduct(const Vector &rhs) const
{
   if (m_dim == rhs.m_dim)
   {
      double ip = 0;
      for (int i=0; i<m_dim; i++)
         ip += m_data[i]*rhs.m_data[i];
      return ip;
   }
   else
   {
      // raise some exception
      return 0.0; // not executed
   }
}
```

Should this be an assert statement?
Depends on whether you expect this to occur at runtime in your released software.

Problem #7 (cont'd)

- Part E

```
void Vector::print(ostream &os) const
{
   os << "Dimension is " << m_dim << endl;
   for (int i=0; i<m_dim; i++)
      os << m_data[i];
   os << endl;
}
```

- Part F

```
Vector::~Vector()
{
   delete[] m_data;
}
```

In case you use vector<double>, dtor should be empty

```
Vector::~Vector()
{
}
```

Summary

- 本次考試主要評估下列概念與語法：
  - 物件
  - 建構元/解構元/初始化串列的設計與呼叫時機
  - static member variable (class variable), static member function (class function)
  - C/C++ 中字串與字元陣列的表示方法與使用方法
  - 封裝與物件分工
  - 類別的設計
  - const 的使用 (參數, 成員函式)
  - 動態配置記憶體 new / delete
  - 基本迴圈練習
  - ifstream 輸入
  - ostream 輸出

Problems of Your Programming

1. 還不能夠短時間寫一段 20-30 列可以正常運作的程式
2. C/C++ 資料型態系統不清楚
3. 看到 C/C++ 指標/參考就昏了
4. 現在連陣列都列入討厭的對象
5. C++ 類別/成員函式/建構元/解構元語法不熟悉
6. 不喜歡 vector<…>
7. 不喜歡 iostream
8. 沒有站在物件的角度去想怎樣寫物件的成員函式, 常常忘了成員函式可以直接存取物件私有的資料成員

每次在寫程式的時候, 不要照著參考範例抄, 至少應該先看過, 瞭解該語法的意義與模型以後, 把資料蓋起來, 盡量讓自己練習把運作的邏輯用所學到的語法轉換爲程式, 熟悉以後才能夠應用