

通識計算機程式設計期中考

臺灣大學 鄭士康 4/24/2015

試題共 7 題，兩面印製 14 頁，滿分 100+2



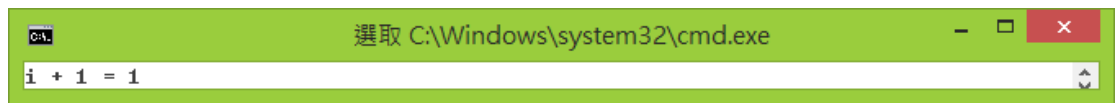
本講義除另有註明外，採創用CC姓名標示-非商業性-相同方式分享3.0臺灣版授權釋出

1. 撰寫一或數個C#敘述達成下列要求: (假設**using System**;敘述已經包含於程式中)
 - (a) 宣告**bool**變數**bl**，**int**變數**m**，**byte**變數**b** (3%)
 - (b) 在螢幕顯示一行字，要求使用者輸入一個整數 (3%)
 - (c) 自鍵盤讀入一個整數，並將其值存入已宣告之**int**變數**m** (3%)
 - (d) 將已宣告設值之**int**變數**m**強制轉型為**byte**數值，存入已宣告之**byte**變數**b** (3%)
 - (e) 將邏輯關係式**m >= 0 && m < 256**設定給已宣告之**bool**變數**bl** (3%)
2. 撰寫一或數個C#敘述達成下列要求: (假設**using System**;敘述已經包含於程式中)
 - (a) 將已宣告設值之**int**變數**m**設定(assign)給他處已宣告設值之**int**變數**n**，再於同一敘述，用遞減算子--讓**m**減1 (3%)
 - (b) 令他處已宣告設值之**int**變數**n**除以5的商設定給他處已宣告之**int**變數**q** (3%)
 - (c) 宣告**double**變數**db**，並設定其值為單一算式(expression) $20.0 \log_{10} |x|$ ；亦即**double**變數**x**取絕對值(**Math.Abs**)，對之求以10為底的對數(**Math.Log10**)，再乘以20.0，假設**x**已於他處宣告設值 (3%)
 - (d) 宣告**int**變數**sgn**，利用三元運算子使其在他處已宣告設值之**double**變數**u**小於0時等於-1，反之則等於1 (3%)
 - (e) 宣告變數**c**為**char**型別，並令其值為換行控制字元(new line) (3%)
3. 撰寫一或數個C#敘述達成下列要求: (假設**using System**; 敘述已經包含於程式中)
 - (a) 先以一個敘述宣告一個亂數產生器物件**rand**，以777為其種子數；再於另一個敘述利用**rand**產生一個亂數，取其對7之餘數，設定給在此宣告為**int**的變數**weekDay** (3%)
 - (b) 宣告一個**int**常數**ARRAY_SIZE**，設其值為5 (3%)
 - (c) 寫一個**do/while**迴圈，迴圈中自鍵盤讀入一個整數**workingHours**，**workingHours**大於24或小於0時繼續，否則離開迴圈。注意在迴圈前要宣告**workingHours**為**int**變數 (3%)

- (d) 利用 `Console.ReadLine().ToCharArray()` 在一個敘述中將讀入之字串轉為字元陣列 `characters`。在另一個敘述中以 `Array.Sort` 將字元陣列 `characters` 由小而大排列。再利用 `Array.Reverse` 倒排，而後取第一個元素，設定給他處已宣告之字元變數 `largest` (3%)
- (e) 寫一個 `void` 函式 `Initialize`，設其唯一的參數為 `out` 之 `double` 參數 `x`，在函式內將 `x` 定為 `1.0` (3%)

4. 找出以下程式片段之錯誤，並予更正。

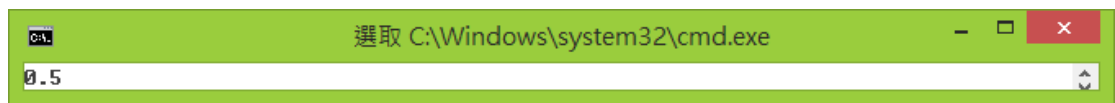
- (a) (3%) 執行時螢幕應顯示



```
C:\> 選取 C:\Windows\system32\cmd.exe
i + 1 = 1
```

```
int i = 0;
Console.WriteLine("i + 1 = " + i + 1);
```

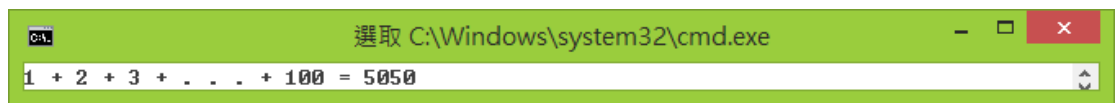
- (b) (3%) 執行時螢幕應顯示



```
C:\> 選取 C:\Windows\system32\cmd.exe
0.5
```

```
int n = 1;
double fraction = n / 2;
Console.WriteLine(fraction);
```

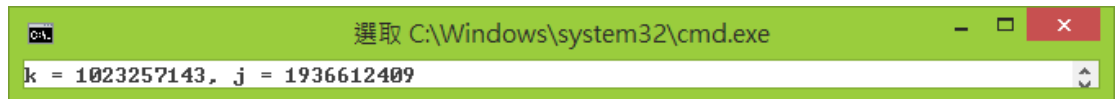
- (c) (3%) 計算 $1 + 2 + 3 + \dots + 100$ ，執行時螢幕應顯示



```
C:\> 選取 C:\Windows\system32\cmd.exe
1 + 2 + 3 + . . . + 100 = 5050
```

```
int i = 1;
int sum = 0;
while(i < 100)
{
    ++i;
    sum += i;
}
Console.WriteLine("1 + 2 + 3 + . . . + 100 = " + sum);
```

(d) (3%) 產生兩個不同的亂數 **k** 和 **j**，執行時螢幕可能顯示為

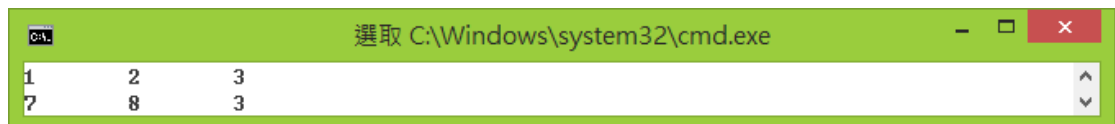


```
選取 C:\Windows\system32\cmd.exe
k = 1023257143, j = 1936612409
```

```
int k = ARandomNumber();
int j = ARandomNumber();
Console.WriteLine("k = {0}, j = {1}", k, j);

. . . . .
static int ARandomNumber()
{
    Random rand = new Random();
    return rand.Next();
}
```

(e) (3%) 執行時螢幕應顯示



```
選取 C:\Windows\system32\cmd.exe
1      2      3
7      8      3
```

```
int[] a = { 1, 2, 3 };
int[] b = a;
b[0] = 7;
b[1] = 8;
for(int i = 0; i < a.Length; i++)
{
    Console.Write(a[i] + "\t");
}
Console.WriteLine();
for(int i = 0; i < b.Length; i++)
{
    Console.Write(b[i] + "\t");
}
Console.WriteLine();
```

5. 試寫出下列程式的螢幕輸出 (5 %)

```
using System;

namespace Problem5
{
    enum Direction
    {
        LEFT,
        RIGHT
    }
    class Program
    {
        static void Main(string[] args)
        {
            Direction[] directionsWithReward = {
                Direction.LEFT,
                Direction.LEFT,
                Direction.RIGHT,
                Direction.LEFT,
                Direction.RIGHT
            };

            Direction[] decisions = {
                Direction.RIGHT,
                Direction.LEFT,
                Direction.LEFT,
                Direction.LEFT,
                Direction.RIGHT
            };

            int nTrials = decisions.Length;
            int nCorrectChoices = 0;
            for(int i = 0; i < decisions.Length; ++i)
            {
                int termScore = (decisions[i] ==
                    directionsWithReward[i]) ? 1 : 0;
                Console.WriteLine("{0} \t {1}", i, termScore);
                nCorrectChoices += termScore;
            }
        }
    }
}
```

```

        Console.WriteLine(
            "Total number of correct choices = :" +
            nCorrectChoices);
    }
}
}

```

6. 試寫出下列程式的螢幕輸出 (10 %)

```

using System;

namespace Problem6
{
    class Program
    {
        static void Main(string[] args)
        {
            const int N_VERTICES = 4;
            int[,] connection =
                new int [N_VERTICES, N_VERTICES]
                {
                    {0, 2, 2, 1},
                    {2, 0, 0, 1},
                    {2, 0, 0, 1},
                    {1, 1, 1, 0}
                };

            for (int i = 0; i < N_VERTICES; i++)
            {
                int start = i + 1;
                TraceOneTrailFromAStartingVertex(
                    connection, start);
            }
        }

        static void TraceOneTrailFromAStartingVertex(
            int[,] connection, int start)
        {

```

```

int[,] workingConnection =
    CopyOfConnection(connection);

int nVertices = connection.GetUpperBound(0) + 1;
int maxNVerticesInTrail = 4 * nVertices;
int[] trail = new int[maxNVerticesInTrail];
Array.Clear(trail, 0, maxNVerticesInTrail);

trail[0] = start;
int nVerticesInTrail = 1;

for (int j = 0; j < maxNVerticesInTrail; ++j)
{
    int from = trail[nVerticesInTrail-1];
    int to = NextVertex(workingConnection, from);
    if (to == 0) break;
    AddAnEdgeToTrail(from, to, workingConnection,
        trail, ref nVerticesInTrail);
}
OutputTrail(trail, nVerticesInTrail);
}

static void AddAnEdgeToTrail(int from, int to,
    int[,] workingConnection,
    int[] trail, ref int nVerticesInTrail)
{
    trail[nVerticesInTrail] = to;
    nVerticesInTrail++;
    workingConnection[from - 1, to - 1]--;
    workingConnection[to - 1, from - 1]--;
}

static void OutputTrail(int[] trail,
    int nVerticesInTrail)
{
    for(int i = 0; i < nVerticesInTrail; i++)
    {
        Console.Write(trail[i] + "\t");
    }
}

```

```

    }
    Console.WriteLine();
}

static int NextVertex(int[,] workingConnection,
    int from)
{
    int result = 0;
    for (int j = 0;
        j < workingConnection.GetUpperBound(0) + 1; j++)
    {
        if (workingConnection[from - 1, j] == 0)
            continue;
        result = j + 1;
        break;
    }
    return result;
}

static int[,] CopyOfConnection(int[,] connection)
{
    int nRow = connection.GetUpperBound(0) + 1;
    int nCol = connection.GetUpperBound(1) + 1;
    int[,] workingConnection = new int [nRow, nCol];
    for(int i = 0; i < nRow; i++)
    {
        for(int j = 0; j < nCol; j++)
        {
            workingConnection[i,j] = connection[i,j];
        }
    }
    return workingConnection;
}
}
}

```

7. 社會科學、行為科學、心理科學、教育心理學、精神醫學、醫學檢驗等領域，經常需要使用問卷、測驗卷、影像判讀結果等，以統計學處理問卷、測驗卷作答或影像判讀結果，支持或揚棄某一假說，建立解釋更廣泛現象的理論，或者解釋特定時空背景的社會、心生理現象，進而產生新的假說，設計新的問卷、測驗卷，進行新的調查與實驗，如此週而復始，這便是社會科學與心理科學中的「科學方法」(scientific method，見：M. Gazzaniga, T. Heatherton, and D. Halpern, *Psychological Science*, 4th ed., New York: W. W. Norton & Company, 2013, pp. 31 – 32)。

科學方法中，問卷、測驗卷等的編製，以及其結果的統計推理，影響所得結論甚鉅，因而產生「心理測驗與評估」(psychological testing and assessment)的「方法論」(methodology)學術研究，對於現代各相關學科的進展，有很大貢獻。

「心理測驗與評估」的研究顯示：問卷、測驗卷等的編製，必須兼具足夠「信度」(reliability)及「效度」(validity)，由其衍生的統計推理，方有足夠證據力支持或推翻某些學術理論假說。限於時間，本題僅與「信度」有關。

《心理測驗》(葉重新，三民書局，1992)第五章第一節說明「信度」的第一個涵義為測量的一致性(consistency)：相同受試者在不同時間，以相同測驗測量；或以複本測量受試者；或以相同測驗在不同情境下測量，所得結果具有高度穩定性(stability)、可靠性(dependability)、可預測性(predictability)。

同書第五章第二節說明常見之信度估計方法有四：

- 1). 再測法(test-retest)：一份測驗，對受試者重複施測。
- 2). 複本法(equivalent-forms)：讓受試者接受多份題目不同，但內容相似的複本測驗。
- 3). 內部一致法(internal consistency)：受試者僅測驗一次，以其結果分析測驗信度。
- 4). 評分者法(scorer)：不同評分者分別評閱試卷，以其評分估算評分者信度(scorer reliability)。

假使試題為單選，例如某軟體公司招考員工，測驗「計算機概論」科目之試題係五選一之選擇題，答對一題得 1 分，答錯及未答得 0 分。顯然四種估計方法中，以內部一致法測試者與受測者雙方需付出之時間與成本最低。此種方法針對不同測驗方式，發展出不同的信度評估法。以上述之「計算機概論」試題而言，Cronbach α 係數方法(L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, 1951, pp. 297-334) 最為普及合用，其計算公式為

$$\alpha = \frac{n}{n-1} \left(1 - \frac{\sum SD_j^2}{SD_t^2} \right)$$

α ：信度估計

n ：測驗每題選項數

N ：應試者總人數

SD_j^2 ：第 j 選項各應試者得分的變異數 (variance) $= \frac{\sum (s_{ij} - \bar{s}_j)^2}{N}$

SD_t^2 ：所有應試者總分的變異數 $= \frac{\sum (T_i - \bar{T})^2}{N}$

s_{ij} ：應試者 i 在第 j 選項答對的題數(得分)

\bar{s}_j ：第 j 選項各應試者答對分數的平均(mean) $= \frac{\sum s_{ij}}{N}$

T_i ：應試者 i 答對總分

\bar{T} ：所有應試者總分的平均 $= \frac{\sum T_i}{N}$

茲以表 1 說明 Cronbach α 係數的計算過程(修改自：葉重新，《心理測驗》，三民書局，1992，頁 94)。由表 1 下所附的計算，得知信度估計為 $\alpha = 0.90$ ，顯示測驗題內容有相當的一致性。至於各種信度估算法的誤差可能來源，請參閱《心理測驗》頁 100~102。

表 1. Cronbach α 係數的計算過程，各應試者均有若干題未答
(修改自：葉重新，《心理測驗》，三民書局，1992，頁 94)

		各應試者答對題數在各選項的分布					應試者 總分
		A	B	C	D	E	
應 試 者	甲	6	8	9	1	7	31
	乙	8	8	8	2	4	30
	丙	0	3	6	1	3	13
	丁	2	1	1	0	2	6
各選項全部應試者總答對分數		16	20	24	4	16	80
各選項全部應試者平均答對分數		4	5	6	1	4	20

計算：1. $n = 5, N = 4$

$$2. SD_t^2 = \frac{(31-20)^2 + (30-20)^2 + (13-20)^2 + (6-20)^2}{4} = 116.5$$

$$3. SD_1^2 = \frac{(6-4)^2 + (8-4)^2 + (0-4)^2 + (2-4)^2}{4} = 10.0$$

$$4. \text{同理, } SD_2^2 = 9.5, SD_3^2 = 9.5, SD_4^2 = 0.5, SD_5^2 = 3.5$$

$$5. \sum SD_j^2 = 10.0 + 9.5 + 9.5 + 0.5 + 3.5 = 33.0$$

$$6. \alpha = \frac{5}{4} \left(1.00 - \frac{33.0}{116.5} \right) = 0.90$$

本題請你依照如下要求，撰寫一個 C# 程式。

1). 程式功能：

計算並顯示某公司招考員工的「計算機概論」測驗之 Cronbach α 信度係數。

2). 系統方塊圖 (system block diagram):

程式的輸入和輸出如圖 1 所示。

3). 背景：

考題為 n 選項的單選題，共有 $nTestee$ 位應試者。

4). 輸入：

各人答對題數之分佈 (建議使用矩形二維陣列，以表 1 為例即

$$\begin{bmatrix} 6 & 8 & 9 & 1 & 7 \\ 8 & 8 & 8 & 2 & 4 \\ 0 & 3 & 6 & 1 & 3 \\ 2 & 1 & 1 & 0 & 2 \end{bmatrix})$$

5). 輸出:

對應測驗的 Cronbach α 信度係數。

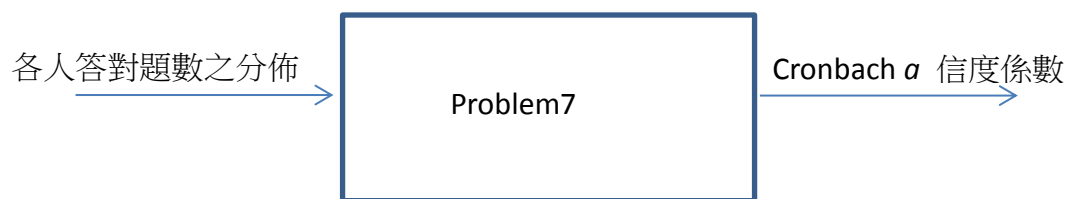


圖 1. 第 7 題的系統方塊圖

程式專案(非單元測試專案)執行畫面如圖 2。

```

C:\Windows\system32\cmd.exe
輸入受試人數: 4
輸入選項個數: 5
輸入第1位受試者在各選項答對之題數, 以單一逗點','間隔, 'Enter' 鍵結束
6,8,9,1,7
輸入第2位受試者在各選項答對之題數, 以單一逗點','間隔, 'Enter' 鍵結束
8,8,8,2,4
輸入第3位受試者在各選項答對之題數, 以單一逗點','間隔, 'Enter' 鍵結束
0,3,6,1,3
輸入第4位受試者在各選項答對之題數, 以單一逗點','間隔, 'Enter' 鍵結束
2,1,1,0,2

受試人數: 4
選項個數: 5

分數分佈表
受試者編號 選項代號      1      2      3      4      5
      1      6      8      9      1      7
      2      8      8      8      2      4
      3      0      3      6      1      3
      4      2      1      1      0      2

此一測驗之Cronbach alpha信度係數為: 0.9
請按任意鍵繼續 . . .

```

圖 2. 第 7 題的專案執行螢幕畫面範例

本題滿分 25 分，全部程式集中寫成一個大 **Main** 函式，不區分函式者，最高得 20 分；善用函式，乃至尚未教到的物件導向程式設計(object-oriented programming)者，最高得 23 分；能利用虛擬碼或流程圖思考，適當劃分函式或類別(class)者，最高得 25 分(使用虛擬碼)或 24 分(使用流程圖)。如能善用 **Debug.Assert** 或單元測試(unit test，建議使用表 1 提供的計算過程數據作為測試範例)，再外加 1~2 分。(25%)

以下的建議與程式碼片段可供參考利用(呼叫此處提供的函式，例如 **MeanSquareError** 或 **InputScore** 時，可不必在答案卷又宣告一次 **MeanSquare** 及 **InputScore** 等函式)：

- 1). 使用 **Debug.Assert** 或單元測試時，注意兩個浮點數(float 或 double)，假設為變數 **actual** 與 **expected**，很難恰巧相等；所以只要滿足 **Math.Abs(actual-expected) < 1.0e-2** (小數點以下 2 位數都相等) 就可以算 **actual == expected** 了。使用 **Debug.Assert** 時，寫成 **Debug.Assert(Math.Abs(actual-expected) < 1.0e-2)**；而在單元測試中，則使用 **Assert.IsTrue(Math.Abs(actual-expected) < 1.0e-2)**。
- 2). 要判斷兩個一維 **double** 陣列(例如以下程式碼中的 **actual** 和 **expected**) 是否可視為相等，可以利用常見的 **mean square error** 公式，寫成

```
Debug.Assert(
    MeanSquareError(actual, expected) < 1.0e-8);
```

或

```
Assert.IsTrue(
    Program.MeanSquareError(actual, expected) < 1.0e-8);
```

其中的 **MeanSquareError** 函式程式碼如下 (假設位於 **Program** 類別內):

```
public static double MeanSquareError(double[] actual,
    double[] expected)
{
    int n = actual.Length;
    double sumOfSquareErrors = 0.0;
    for (int i = 0; i < n; i++)
    {
        double error = actual[i] - expected[i];
        sumOfSquareErrors += error * error;
    }
    return sumOfSquareErrors / n;
}
```

- 3). 輸入成績分布、複誦(echo)成績分布、輸出 Cronbach α 係數，可以利用如下函式：

```
public static int[,] InputScores()
{
    Console.Write("輸入受試人數: ");
    int nTestees = int.Parse(Console.ReadLine());
    Console.Write("輸入選項個數: ");
    int n = int.Parse(Console.ReadLine());
    int[,] scores = new int[nTestees, n];
    for(int i = 0; i < nTestees; i++)
    {
        Console.WriteLine(
```

"輸入第{0}位受試者在各選項答對之題數，以單一逗點','間隔，'Enter'鍵結束",

```

        i + 1);
        string[] str = Console.ReadLine().Split(',');
        for(int j = 0; j < n; j++)
        {
            scores[i, j] = int.Parse(str[j]);
        }
    }
    return scores;
}

public static void Echo(int[,] scores)
{
    Console.WriteLine();
    int nTestees = scores.GetUpperBound(0) + 1;
    Console.WriteLine("受試人數: " + nTestees);
    int n = scores.GetUpperBound(1) + 1;
    Console.WriteLine("選項個數: " + n);
    Console.WriteLine();

    Console.WriteLine("分數分佈表");
    Console.Write("受試者編號\\選項代號\t");
    for (int j = 0; j < n; j++ )
    {
        Console.Write("{0}\t", j + 1);
    }
    Console.WriteLine();
    for (int i = 0; i < nTestees; i++)
    {
        Console.Write("\t{0}\t\t", i + 1);
        for (int j = 0; j < n; j++)
        {
            Console.Write(scores[i, j] + "\t");
        }
        Console.WriteLine();
    }
    Console.WriteLine();
}

```

```

public static void OutputAlpha(double alpha)
{
    Console.WriteLine();
    Console.WriteLine(
        "此一測驗之Cronbach alpha信度係數為: " + alpha);
}

```

- 4). Visual Studio 2013 Professional 加裝 NUnit Test Adapter 後，其 UnitTest Project 內的單元測試程式架構為

```

using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMethod1()
        {
        }
    }
}

```