ชื่อ_____รหัสนิสิต_____หมู่_____

Lab 10 - เมธอด ภาค 2 (การส่งพารามิเตอร์) และการใช้งานไฟล์

จากเนื้อหาที่ผ่านมาเราได้เรียนรู้วิธีการประกาศ การใช้เรียกใช้เมธอด การส่งค่าผ่านพารามิเตอร์เข้าสู่เมธอดและการคืนค่าจากเมธอดสู่โปรแกรมหลัก ซึ่งในสัปดาห์นี้เราจะได้เรียนการส่งพารามิเตอร์ให้ลึกซึ้งยิ่งขึ้น

1. การส่งพารามิเตอร์แบบการส่งค่า (Pass by value)

การส่งข้อมูลผ่านพารามิเตอร์เข้าไปในเมธอดที่เราทำกันในสัปดาห์ที่แล้วนั้นล้วนเป็นการส่งแบบ pass by value ทั้งสิ้น โดยค่าที่ส่งตอนแรกเมธอดนั้นจะถูกคัดลอกไปให้ตัวแปรที่เป็นพารามิเตอร์ภายในเมธอด ให้นักเรียนศึกษาจากแบบฝึกหัดที่ 1.1 ต่อไปนี้

แบบฝึกหัดที่ 1.1 : การส่งค่าให้กับเมธอดผ่านพารามิเตอร์

ให้นักเรียนคัดลอกโปรแกรมและรันโปรแกรมต่อไปนี้จากนั้นบันทึกผลลัพธ์

	โปรแกรมที่เขียนได้คือ	บันทึกผลลัพธ์ที่ได้ลงในช่องด้านล่าง
1	`using System;`	
2	`class MainClass`	
3	`{`	
4	`  static void sample(int n)`	
5	`  {`	
6	`  Console.WriteLine("In method n = {0}",n);`	
7	`  n = 77;`	
8	`  }`	
9		
10	`  static void Main ()`	
11	`  {`	
12	`  int a = 99;`	
13	`  sample(a);`	
14	`  Console.WriteLine("After method a = {0}",a);`	
15	`  }`	
16	`}`	

จากโปรแกรมในแบบฝึกหัดที่ 1.1 เมื่อเราเรียกใช้เมธอด sample โดยระบุ a เป็นพารามิเตอร์ในบรรทัดที่ 13 โปรแกรมจะทำการคัดลอกค่าในตัวแปร a (ในที่นี้คือ 99) ส่งให้กับตัวแปร n ในเมธอด sample (บรรทัดที่ 4) ทำให้ค่าในตัวแปร n ในบรรทัดที่ 4 มีค่าเท่ากับ 99 ด้วย ดังนั้นคำสั่ง Console.WriteLine() ในบรรทัดที่ 6 จึงแสดงข้อความ “In method n = 99” ออกมาทางหน้าจอ

อย่างไรก็ตามตัวแปร n ในเมธอด sample หลังจากรับค่า 99 มาแล้ว ก็ไม่ได้มีความสัมพันธ์กับตัวแปร a อีกแต่อย่างใด ทำให้แม้ว่าค่าของตัวแปร n จะถูกเปลี่ยนไปเป็น 77 ในบรรทัดที่ 7 ก็ไม่ได้ส่งผลให้ค่าในตัวแปร a เปลี่ยนตามด้วย ทำให้ค่าในตัวแปร a ยังคงเป็น 99 ดังเดิม ดังนั้นคำสั่ง Console.WriteLine() ในบรรทัดที่ 14 จึงแสดงข้อความ “After method a = 99” (ค่า a ยังคงเป็น 99 ดังเดิม)

2. การส่งพารามิเตอร์แบบใช้การอ้างอิง (Pass by reference)

2.1 การส่งพารามิเตอร์แบบใช้การอ้างอิงด้วยคำสงวน ref

การส่งพารามิเตอร์แบบใช้การอ้างอิง (pass by reference) มีความแตกต่างจากการส่งแบบใช้ค่า (pass by value) ตรงที่ตัวแปรในเมธอดที่ถูกเรียกจะอ้างอิงกับตัวแปรต้นทางที่ส่งเป็นพารามิเตอร์ตอนเรียกเมธอดในโปรแกรมหลัก ทำให้ตัวแปรทั้งสองตัวนี้เปรียบเสมือนเป็นตัวแปรเดียวกัน ดังนั้นเมื่อตัวแปรในเมธอดเกิดการเปลี่ยนค่า ตัวแปรที่อ้างอิงกับตัวแปรนั้นในส่วนของโปรแกรมหลักก็จะเปลี่ยนค่าด้วย

การระบุให้การส่งพารามิเตอร์เป็นแบบใช้การอ้างอิงสามารถทำได้โดยระบุคำพิเศษ **ref** ไว้หน้าตัวแปรทั้งสองตำแหน่ง (ทั้งในส่วนของตัวแปรที่เรียกใช้เมธอด และตัวแปรในเมธอดที่อ้างอิงกัน)

แบบฝึกหัดที่ 2.1.1 : การส่งค่าให้กับเมธอดผ่านพารามิเตอร์แบบใช้การอ้างอิง

ให้นักเรียนนำโปรแกรมในแบบฝึกหัดที่ 1.1 มาดัดแปลงโดยแก้ข้อความ int n ในบรรทัดที่ 4 เป็น ref int n และ sample(a); ในบรรทัดที่ 13 เป็น sample(ref a); จากนั้นให้สังเกตผลลัพธ์ที่แปลี่ยนแปลงไปและบันทึกผลลัพธ์ที่ได้

ผลลัพธ์ที่ได้จากโปรแกรมที่ดัดแปลงแล้วคือ

แบบฝึกหัดที่ 2.1.2 : เมธอด AddBonus

เกมสที่มีผู้เล่น 3 คนเมื่อจบแต่ละด่านจะมีคะแนน Bonus สะสมเพิ่มให้กับทั้งสี่คนคนละเท่า ๆ กัน หากคะแนนของผู้เล่นทั้งสี่เก็บอยู่ในตัวแปร s1, s2 และ s3 ตามลำดับ จะเขียนเมธอด add_bonus ให้สมบูรณ์เพื่อทำหน้าที่เพิ่มคะแนนโบนัสให้กับตัวแปรทั้ง 3 ตัวแปรข้างต้น

โปรแกรมที่เขียนได้คือ

```
using System;
class MainClass
{
    static void add_bonus(____________________){
        ________________________________________
        ________________________________________
    }
    public static void Main(string[] args)
    {
        int s1 = 1, s2 = 2, s3 = 3;
        Console.Write("Enter Bonus: ");
        int bonus = double.Parse(Console.ReadLine());
        add_bonus(ref s1, ref s2, ref s3, bonus);
        Console.WriteLine("{0}, {1}, {2}",s1,s2,s3);
        Console.ReadKey();
    }
}
```

ตัวอย่างการทำงาน

ตัวอย่างที่ 1

```
Enter Bonus : 5
6, 7, 8
```

ตัวอย่างที่ 2

```
Enter Bonus : 20
21, 22, 23
```

แบบฝึกหัดที่ 2.1.3 : เมธอด Swap

งานของคุณคือการเติมโปรแกรมด้านล่างให้สมบูรณ์ โปรแกรมจะอ่านจำนวนเต็มสองจำนวน คือ a และ b จากนั้นสลับค่าของตัวแปรทั้งสอง

โปรแกรมที่เขียนได้คือ

```
using System;
class MainClass
{
    static void swap(____________________){
        ________________________________________
        ________________________________________
        ________________________________________
    }
    public static void Main(string[] args)
    {
        int sum = 0;
        Console.Write("Enter a: ");
        int a = int.Parse(Console.ReadLine());
        Console.Write("Enter b: ");
        int b = int.Parse(Console.ReadLine());
        swap(______________________);
        Console.WriteLine("Swap a and b");
        Console.WriteLine("a={0} b={1}",a,b);
        Console.ReadKey();
    }
}
```

ตัวอย่างการทำงาน

```
Enter a: 25
Enter b: 70
Swap a and b
a=70 b=25
```

แบบฝึกหัดที่ 2.1.4 : เมธอด read_two_number

ให้นักเรียนเขียนเมธอด ReadTwoInput สำหรับรับข้อมูลเป็นเลขจำนวนเต็ม 2 จำนวนพร้อมเติมคำในช่องว่างของเมธอด Main() ที่กำหนดให้ต่อไปนี้ให้สมบูรณ์เพื่อให้โปรแกรมแสดงผลตามตัวอย่าง

	โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน
1	`using System;`	`Enter two number:`
2	`class RefExample {`	`10`
3	`  static void ReadTwoNums(_____) {`	`5`
4	`  Console.WriteLine("Enter two number:");`	`Numbers are 10 and 5.`
5	`  a = int.Parse(Console.ReadLine());`	
6	`  b = int.Parse(Console.ReadLine());`	
7	`  }`	
8		
9	`  static void Main() {`	
10	`  int n1=0, n2=0;`	
11	`  ReadTwoNums(_____);`	
12	`  Console.WriteLine("Numbers are {0} and {1}", n1, n2);`	
13	`  }`	
14	`}`	

2.2 การส่งพารามิเตอร์แบบใช้การอ้างอิงด้วยคำสงวน out

ในบางสถานการณ์เราต้องการเขียนเมธอดเพื่อทำการส่งค่าออกเพียงอย่างเดียวผ่านทางพารามิเตอร์โดยไม่สนใจค่าปัจจุบันของพารามิเตอร์นั้น ณ ขณะที่เมธอดถูกเรียกใช้จะมีค่าเป็นเท่าใด หากเราใช้คีย์เวิร์ด ref ในกรณีนี้ภาษา C# จะบังคับให้เราระบุค่าเริ่มต้นให้กับพารามิเตอร์นั้น ๆ แม้ว่าค่าที่กำหนดให้จะไม่ได้ถูกนำไปใช้ก็ตาม ให้นิสิตลองพิจารณาแบบฝึกหัดต่อไปนี้

แบบฝึกหัดที่ 2.2.1 : เมธอด read_two_number

จากโปรแกรมในแบบฝึกหัดที่ 2.1.4 หากคำสั่งในบรรทัดที่ 10 ถูกเปลี่ยนเป็น int n1,n2; ผลลัพธ์ของโปรแกรมที่เกิดขึ้นจะเป็นเช่นไร

เหตุใดจึงเป็นเช่นนั้น

จากแบบฝึกหัดข้างต้นจะเห็นว่าพารามิเตอร์ที่ใช้คำสงวน ref นั้นจะต้องถูกกำหนดค่าเริ่มต้น เอาไว้เสมอซึ่งถ้าหากเราพิจารณาจุดมุ่งหมายของเมธอด ReadTwoNums แล้วเมธอดนี้ต้องการเพียงแค่ใช้พารามิเตอร์ สำหรับส่งค่าที่รับจากผู้ใช้กลับคืนไปยังโปรแกรมหลักในเมธอด Main เท่านั้น และไม่ได้มีการนำค่าเดิมไปใช้

เพื่อหลีกเลี่ยงความไม่จำเป็นดังกล่าว ภาษา C# ได้จัดเตรียมคำสงวน out เพื่อใช้ระบุว่าพารามิเตอร์ใดบ้างจะถูกใช้เพื่อส่งค่าออกเพียงอย่างเดียว การใช้งานคำสงวนนี้จะมีผลเช่นเดียวกับการใช้คำสงวน ref คือ พารามิเตอร์จะถูกส่งแบบอ้างอิง สิ่งที่แตกต่างกันก็คือ เมธอดนั้น ๆ จะไม่สนใจค่าที่มีอยู่แล้วภายในพารามิเตอร์ กล่าวคือพารามิเตอร์เหล่านี้จะมีผลภายในเมธอดเสมือนกับการประกาศตัวแปรที่ไม่ได้ระบุค่าเริ่มต้นให้ โดยตารางด้านล่างเปรียบเทียบความแตกต่างของการใช้ ref และ out

ref	out
ไม่ยอมรับพารามิเตอร์ที่เป็นตัวแปรที่ไม่ถูกระบุค่าเริ่มต้น	ตัวแปรที่เป็นพารามิเตอร์จะถูกระบุค่าเริ่มต้นหรือไม่ก็ได้
ค่าปัจจุบันในพารามิเตอร์ถูกนำไปใช้ในเมธอดได้	ค่าปัจจุบันของพารามิเตอร์จะไม่ถูกส่งไปยังเมธอด

ดังนั้นจากแบบฝึกหัดที่ 2.2.1 จึงสามารถเขียนโดยใช้คำสงวน out แทนการใช้ ref ได้ดังนี้

```
using System;
class OutOnly {
    static void ReadTwoNums(out int a, out int b) {
        Console.WriteLine("Enter two number:");
        a = int.Parse(Console.ReadLine());
        b = int.Parse(Console.ReadLine());
    }

    static void Main() {
        int n1, n2;
        ReadTwoNums(out n1, out n2);
        Console.WriteLine("Numbers are {0} and {1}", n1, n2);
    }
}
```

แบบฝึกหัดที่ 2.2.2 : เมธอด Distance และการอ่านค่า

โปรแกรมด้านล่างอ่านพิกัดของจุดสองจุดเป็นจำนวนจริง จากนั้นคำนวณระยะทางระหว่างจุดสองจุด จงเขียนเมธอด ReadPoint และ Distance ให้สมบูรณ์ หมายเหตุ ระยะทางระหว่างจุด (x1,y1) และ (x2,y2) คือ

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class MainClass {`
`static void ReadPoint(_____)`
`{`
`Console.Write("Enter x-coordinate: ");`
`x = double.Parse(Console.ReadLine());`
`Console.Write("Enter y-coordinate: ");`
`y = double.Parse(Console.ReadLine());`
`}`
 	
`static double Distance(_____)`
`{`
`_____`
`_____`
`}`
 	
`static void Main()`
`{`
`double ax, ay, bx, by;`
`Console.WriteLine("Point A:");`
`ReadPoint(out ax, out ay);`
`Console.WriteLine("Point B:");`
`ReadPoint(out bx, out by);`
`Console.WriteLine("The distance between A and B is {0:f2}"`
`,Distance(ax,ay,bx,by));`
`}`
`}`
	ตัวอย่างที่ 1
	`Point A:`
	`Enter x-coordinate: 1`
	`Enter y-coordinate: 2`
	`Point B:`
	`Enter x-coordinate: 3`
	`Enter y-coordinate: 4`
	`The distance between A and B is 2.83`
	
	ตัวอย่างที่ 2
	`Point A:`
	`Enter x-coordinate: 20.55`
	`Enter y-coordinate: -18.12`
	`Point B:`
	`Enter x-coordinate: -49.74`
	`Enter y-coordinate: -3.82`
	`The distance between A and B is 71.73`
	
	ตัวอย่างที่ 3
	`Point A:`
	`Enter x-coordinate: 2`
	`Enter y-coordinate: -4`
	`Point B:`
	`Enter x-coordinate: -4`
	`Enter y-coordinate: 2`
	`The distance between A and B is 8.49`

3. ฝึกเขียนโปรแกรมแสนสนุก

แบบฝึกหัดที่ 3.1 : โรงงานขนม

คุณกำลังเตรียมการเพื่อผลิตลูกกวาดชั้นดี กำไรที่คุณจะได้เมื่อผลิตสินค้าจำนวน หน่วยคือ

$$1000 + 50x + 150x^2 - x^3$$

คุณต้องการจะได้กำไรมากที่สุด อย่างไรก็ตาม คุณต้องรับประกันด้วยว่า จำนวนสินค้าที่ผลิตจะต้องอยู่ระหว่าง ถึง หน่วย (นั่นคือ,). นอกจากนี้คุณจะต้องผลิตสินค้าเป็นจำนวนเต็มหน่วย นั่นคือ จะต้องเป็นจำนวนเต็ม

เขียนเมธอด Profit ด้านล่าง ที่คำนวณกำไร และเติมโปรแกรมสำหรับหาค่า ที่ให้กำไรมากที่สุดให้สมบูรณ์

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class MainClass {`
`  static int Profit(int x)`
`  {`
`  ____________________________`
`  ____________________________`
`  }`
 	
`  static void Main()`
`  {`
`  int a, b;`
`  Console.Write("Enter A:");`
`  a = int.Parse(Console.ReadLine());`
`  Console.Write("Enter B:");`
`  b = int.Parse(Console.ReadLine());`
 	
`  int best = Profit(a);`
`  int bestAt = a;`
`  ____________________________`
`  ____________________________`
`  ____________________________`
`  ____________________________`
`  ____________________________`
`  ____________________________`
`  ____________________________`
`  ____________________________`
 	
`  Console.WriteLine("Best profit is {0} at {1}", best, bestAt);`
`  }`
`}`
	`Enter A: 0`
	`Enter B: 10`
	`Best profit is 15500 at 10`

แบบฝึกหัดที่ 3.2 : กบ

เช้าวันหนึ่ง กบตัวหนึ่งตกลงไปในบ่อความลึก H เมตร. ในเวลากลางวัน กบสามารถไต่ขึ้นมาได้ D เมตร แต่จะไถลลงไปในหลุมอีก ในเวลากลางคืนเป็นระยะ N เมตร

เขียนโปรแกรมที่อ่านจำนวนเต็ม 3 สามจำนวน คือ H, D และ N จากนั้นคำนวณจำนวนวันที่กบจะสามารถออกจากหลุมได้

คุณสามารถสมมติได้ว่า ถ้าทำไปเรื่อย ๆ แล้ว กบจะออกจากหลุมได้

โปรแกรมที่เขียนได้คือ

```
using System;
class MainClass {
    static int h, d, n;

    /* this method checks if y is out of the well of depth h */
    static bool IsOut(int y, int h)
    {
        ________________________________
        ________________________________
    }

    static void ClimbUp(ref int y)
    {
        ________________________________
        ________________________________
    }

    static void FallDown(ref int y)
    {
        ________________________________
        ________________________________
    }

    static void Main()
    {
        h = int.Parse(Console.ReadLine());
        d = int.Parse(Console.ReadLine());
        n = int.Parse(Console.ReadLine());

        int day = 1;
        int y = 0;
        while(!IsOut(y,h))
        {
            ___________________________________
            ___________________________________
            ___________________________________
            ___________________________________
            ___________________________________
            day++;
        }
        Console.WriteLine("Get out on day {0}", day);
    }
}
```

ตัวอย่างการทำงาน

ตัวอย่างที่ 1

```
10
3
1
Get out on day 5
```

ตัวอย่างที่ 2

```
10
4
2
Get out on day 4
```

แบบฝึกหัดที่ 3.3 : ตัวประกอบกำลังสอง (while loop)

เขียนโปรแกรมที่รับจำนวนเต็ม X หลังจากนั้นพิมพ์ตัวประกอบของ X ที่เป็นกำลังสองทั้งหมดออกมา ทั้งนี้จำนวนเต็มจะเป็น **กำลังสอง** ถ้าจำนวนดังกล่าวเขียนให้อยู่ในรูปจำนวนเต็มบางจำนวนยกกำลังสองได้ เช่น 1, 4, 9, และ 25 ต่างก็เป็นกำลังสอง แต่ 3, 15 และ 30 ไม่ใช่

ตัวประกอบกำลังสอง คือ ตัวประกอบที่เป็นกำลังสอง ตัวอย่างเช่น 1000 มีตัวประกอบกำลังสอง 4 จำนวนคือ 1, 4, 25 และ 100

เติมโปรแกรมด้านล่างให้สมบูรณ์

*หมายเหตุ: โปรแกรมด้านล่างทำงานค่อนข้างช้าถ้า X มีค่ามาก (ทดลองดูได้) ในอนาคตเราจะเขียนโปรแกรมเดียวกันที่ทำงานเร็วขึ้นในโจทย์อีกข้อหนึ่ง

โปรแกรมที่เขียนได้คือ

```
using System;
class MainClass
{
    /* method IsSquare checks if y is a square. */
    static bool IsSquare(int y)
    {
        ____________________________________
        ____________________________________
        ____________________________________
        ____________________________________
    }

    /* method IsSquareFactor checks if f is a square factor of x
*/
    static bool IsSquareFactor(int f, int x)
    {
       ____________________________________
       ____________________________________
       ____________________________________
    }

    public static void Main(string[] args)
    {
        int x = int.Parse(Console.ReadLine());
        int i = 1;
        while(_________________)
        {
            if(___________________)
                Console.Write("{0} ",______________);
            i++;
        }
        Console.WriteLine();
    }
}
```

ตัวอย่างการทำงาน

```
1000
1 4 25 100
```

แบบฝึกหัดที่ 3.4 : บวกหรือลบ

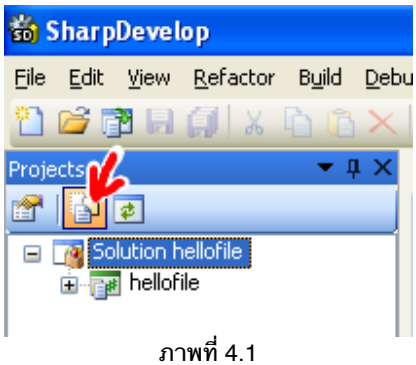
จงเขียนเมธอด CountPlusMinus ที่อ่านจำนวนเต็มจากผู้ใช้ไปเรื่อย ๆ จนกระทั่งผู้ใช้ป้อน 0 ระหว่างนั้นให้นับว่ามีจำนวนเต็มบวกกี่จำนวน และจำนวนเต็มลบกี่จำนวน แล้วส่งค่ากลับไปให้โปรแกรมหลักผ่านทางพารามิเตอร์

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
 	
`class Program`
`{`
`  static void CountPlusMinus(_____)`
`  {`
`  _____`
`  _____`
`  _____`
`  _____`
`  _____`
`  }`
 	
`  public static void Main(string[] args)`
`  {`
`  int plus, minus;`
`  CountPlusMinus(_____);`
`  Console.WriteLine("Positive: {0}",plus);`
`  Console.WriteLine("Negative: {0}",minus);`
`  Console.ReadLine();`
`  }`
`}`
	10
	20
	30
	-10
	-20
	0
	Positive: 3
	Negative: 2

4. การใช้งานไฟล์

ในหลาย ๆ โปรแกรมเราอาจต้องการเก็บผลลัพธ์ในการคำนวณไว้ ใช้ในโอกาสอื่น ๆ การใช้ตัวแปรเก็บค่า เมื่อปิดโปรแกรมค่าที่ได้ก็จะหายไปด้วย การใช้ไฟล์จึงเป็นทางเลือกที่น่าสนใจทางหนึ่งในการเก็บข้อมูลระยะยาว



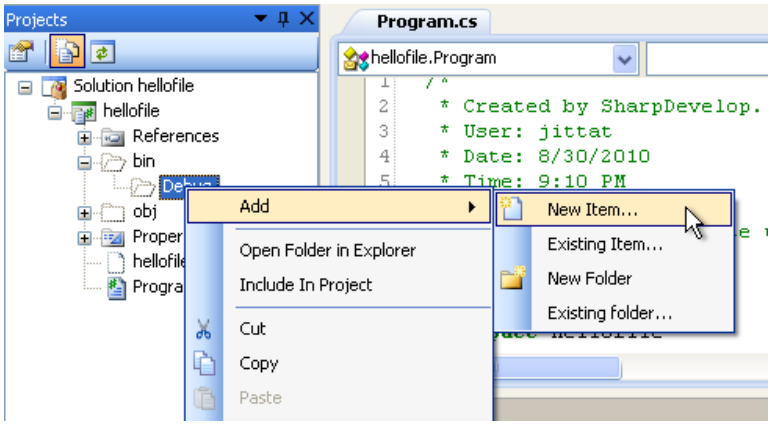
ภาพที่ 4.1

แบบฝึกหัดที่ 4.1 : ทดลองอ่านไฟล์

ในข้อนี้เราจะทดลองสร้างไฟล์ข้อมูลแล้วอ่าน จากนั้นจะแก้โปรแกรมเล็กน้อย ให้เริ่มต้นจากการสร้าง solution ใหม่ จากนั้นเราจะไปสร้างไฟล์ข้อมูล name.txt สำหรับทดลองตามขั้นตอนต่อไปนี้

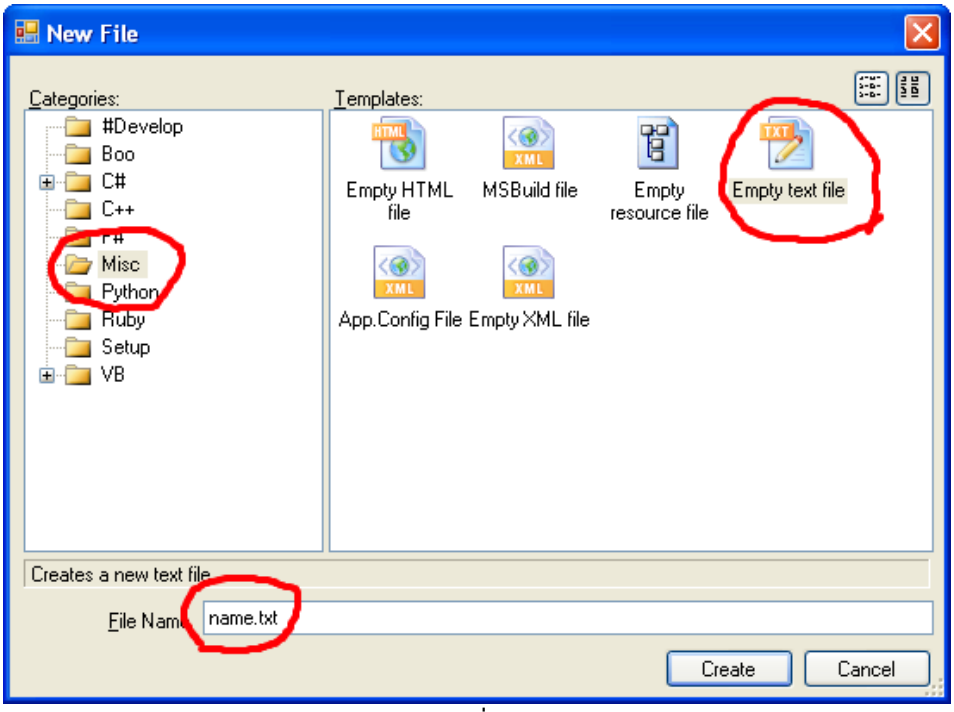
1. ก่อนอื่นไปกดเลือกให้ส่วน Projects ของ Sharp Dev แสดงแฟ้มข้อมูลอื่น ๆ ด้วย โดยกดเลือกไอคอน Show all files ดังภาพที่ 4.1

2. จากนั้นกดสร้างไฟล์ใหม่ดังนี้ เลือกไฟล์โดยกดไล่ในรายการ Solution ไปที่ bin กดต่อไปที่ Debug จากนั้นกดขวาแล้วเลือก Add และ New Item... ดังแสดงในภาพที่ 4.2

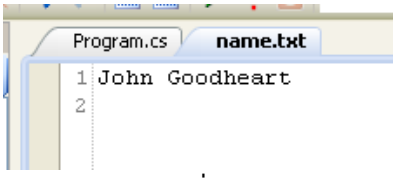


ภาพที่ 4.2

3. เมื่อเลือก New Item แล้ว จะมีหน้าต่างสอบถามชนิดแฟ้มข้อมูลและชื่อ ให้เลือก Categories เป็น Misc และประเภทเป็น Empty text file จากนั้นในช่อง File name ให้ใส่เป็น name.txt เมื่อเลือกเรียบร้อยให้กด Create



ภาพที่ 4.3



ภาพที่ 4.4

4. จากนั้นในส่วน Editor จะมีแทบเพิ่มขึ้น ชื่อว่า name.txt ให้ทดลองพิมพ์ข้อความ John Goodheart ลงไปดังภาพที่ 4.4 แล้วจัดเก็บ โดยอาจจะกด Save จากเมนู File หรือกด Ctrl-S ก็ได้ (เมื่อ Save แล้วจะเห็นว่าเครื่องหมาย * จะหายไป)

5. ให้ทดลองป้อนโปรแกรมต่อไปนี้ แล้วสั่งให้ทำงาน เราจะเห็นผลลัพธ์เป็นข้อมูลบรรทัดแรกในแฟ้ม name.txt ถ้าโปรแกรมเกิดข้อผิดพลาดขึ้น โดยมากจะมาจากการจัดเก็บชื่อไฟล์ผิด หรือยังไม่ได้เก็บไฟล์ name.txt

```
using System;
using System.IO;

namespace hellofile
{
  class Program
  {
     public static void Main(string[] args)
     {
        StreamReader reader = new StreamReader("name.txt");
        string name = reader.ReadLine();
        Console.WriteLine(name);
        Console.ReadLine();
        reader.Close();
     }
  }
}
```

6. ให้แก้ข้อมูลบรรทัดแรกในแฟ้ม name.txt ดังกล่าว จัดเก็บ แล้วทดลองสั่งให้ทำงานอีกครั้ง ผลลัพธ์ที่ได้จะเปลี่ยนไปเป็นข้อมูลใหม่

งานของคุณ

ให้เพิ่มบรรทัดที่สองลงในแฟ้ม name.txt จากนั้นแก้โปรแกรมให้พิมพ์ข้อมูลในแฟ้ม name.txt ทั้งสองบรรทัดแรกตามลำดับ ดังตัวอย่างในตารางด้านล่าง

สมมุติข้อมูลในแฟ้ม name.txt คือ	ผลลัพธ์ที่โปรแกรมแสดงทางหน้าจอคือ
Hello Goodbye My dear friend	Hello Goodbye My dear friend

โปรแกรมที่เขียนได้คือ

แบบฝึกหัดที่ 4.2 : ทดลองอ่านไฟล์ 2

จงเขียนโปรแกรมอ่านแฟ้ม name.txt ซึ่งประกอบด้วยข้อมูลสองบรรทัด จากนั้นแสดงข้อมูลสองบรรทัดสลับกัน ดังตัวอย่างในตารางด้านล่าง

โปรแกรมที่เขียนได้คือ	สมมุติข้อมูลในแฟ้ม name.txt คือ
	Hello Goodbye My dear friend
	ผลลัพธ์ที่โปรแกรมแสดงทางหน้าจอคือ
	My dear friend Hello Goodbye

แบบฝึกหัดที่ 4.3 : อ่านเลขเพื่อบวก

ในข้อนี้เราจะเขียนโปรแกรมที่อ่านแฟ้มข้อมูลชื่อ numbers.txt ที่ประกอบไปด้วยข้อมูล 3 บรรทัด แต่ละบรรทัดมีจำนวนเต็ม 1 จำนวน จากนั้นคำนวณผลบวกของจำนวนทั้งสาม แล้วพิมพ์ผลลัพธ์ออกทาง Console

โปรแกรมที่เขียนได้คือ	สมมุติข้อมูลในแฟ้ม numbers.txt คือ

`using System;`
`_____`
 	
`namespace hellofile`
`{`
`  class Program`
`  {`
`  public static void Main(string[] args)`
`  {`
`  StreamReader reader = _____`
`  int a = int.Parse(_____);`
`  int b = _____`
`  int c = _____`
`  Console.WriteLine(a+b+c);`
`  Console.ReadLine();`
`  reader._____`
`  }`
`  }`
`}`
	1
	10
	200

	ผลลัพธ์ที่โปรแกรมแสดงทางหน้าจอคือ
	211

แบบฝึกหัดที่ 4.4 : รวมตัวเลขจากแฟ้ม

จงเขียนโปรแกรมที่อ่านข้อมูลเป็นจำนวนเต็ม N จำนวนจากแฟ้ม input.txt แล้วหาผลรวมเฉพาะข้อมูลที่เป็นบวก ทั้งนี้แฟ้มชื่อ input.txt โดยข้อมูลมีรูปแบบดังนี้

- บรรทัดแรกในแฟ้ม มีจำนวนเต็ม N ระบุจำนวนข้อมูลที่ต้องการหาผลรวม
- อีก N บรรทัดถัดไป แต่ละบรรทัดมีจำนวนเต็มหนึ่งจำนวน

ให้โปรแกรมพิมพ์ผลลัพธ์เป็นผลรวมของข้อมูลที่เป็นบวกทั้งหมด

โปรแกรมที่เขียนได้คือ

```
using System;
using System.IO;

namespace hellofile
{
    class Program
    {
        public static void Main(string[] args)
        {
            StreamReader reader = ________________________
            int n = int.Parse(________________________);
            int i = 0;
            int total = 0;
            while(i < n)
            {
                ________________________
                ________________________
                ________________________
                ________________________
            }
            Console.WriteLine(total);
            Console.ReadLine();
            reader.Close();
        }
    }
}
```

สมมุติข้อมูลในแฟ้ม input.txt คือ

```
5
10
-4
-2
7
5
```

ผลลัพธ์ที่โปรแกรมแสดงทางหน้าจอคือ

```
22
```

แบบฝึกหัดที่ 4.5 : ค่าเฉลี่ยล่าสุด

เมื่อเราได้รับข้อมูลในรายการทีละตัว เราสามารถคำนวณหาค่าเฉลี่ยล่าสุดเมื่อได้รับข้อมูลเพิ่มขึ้นแต่ละตัวได้ ยกตัวอย่างเช่น ถ้าเราได้รับข้อมูลเป็นรายการ 1, 5, 2, 7 ค่าเฉลี่ยล่าสุดในแต่ละครั้งที่เราได้รับข้อมูลคือ 1, 3, 2.667 และ 3.750 ทั้งนี้เนื่องจาก 1 = 1/1, 3 = (1+5)/2, 2.667 คือค่าประมาณของ (1+5+2)/3 และ 3.750 = (1+5+2+7)/4

เราจะเขียนโปรแกรมอ่านจำนวนจริงจากแฟ้ม input.txt ไปเรื่อย ๆ จนกว่าจะได้รับค่า 0 ระหว่างที่อ่านให้พิมพ์ค่าเฉลี่ยล่าสุดและผลรวมออกมาเป็นเลขทศนิยม 3 ตำแหน่ง

โปรแกรมที่เขียนได้คือ

```
using System;
using System.IO;
namespace moving_average
{
  class Program
  {
    static void Update(_______________________)
    {
      ____________________________________________
      ____________________________________________
      ____________________________________________
    }

    public static void Main(string[] args)
    {
      StreamReader reader = new StreamReader("input.txt");
      int n = 0;
      double total = 0;
      double avr = 0;
      double x;
      do {
        x = double.Parse(reader.ReadLine());
        if(x != 0)
        {
          Update(__________________________);
          Console.WriteLine("{0:f3} {1:f3}", total, avr);
        }
      } while(___________________);
      reader.Close();
      Console.ReadLine();
    }
  }
}
```

สมมุติข้อมูลในแฟ้ม input.txt คือ

```
1
5
2.5
7
0
```

ผลลัพธ์ที่โปรแกรมแสดงทางหน้าจอคือ

```
1.000 1.000
6.000 3.000
8.500 2.833
15.500 3.875
```

แบบฝึกหัดที่ 4.6 : ค่าเฉลี่ยล่าสุด (ไม่ส่งค่า)

โจทย์นี้เป็นโจทย์เดียวกันกับโจทย์ในแบบฝึกหัดที่ 4.5 อย่างไรก็ตาม โครงของโปรแกรมที่ให้เพิ่มมีลักษณะแตกต่างกัน ให้นักเรียนเขียนโปรแกรมที่ตอบโจทย์ข้อ 4.5 โดยใช้ส่วนหนึ่งของโปรแกรมเป็นดังด้านล่าง

*คำใบ้ ข้อนี้ต้องใช้ตัวแปร global (เหมือนตัวแปร global ในภาษาไพทอน) โดยการประกาศตัวแปร global ในภาษา C# สามารถทำได้โดยใช้รูปแบบ

static <ชนิดของตัวแปร> <ชื่อตัวแปร> ;

อาทิเช่น static int n; โดยประกาศไว้ใน class แต่อยู่นอกเมธอดทุกเมธอด ซึ่งถ้าเป็นโปรแกรมด้านล่างจะหมายถึงบริเวณบรรทัดที่ 7 - 9

โปรแกรมที่เขียนได้คือ

```
1  using System;
2  using System.IO;
3  namespace moving_average
4  {
5    class Program
6    {
7      ____________________________________________
8      ____________________________________________
9      ____________________________________________
10
11     static void Update(double x)
12     {
13       ________________________________________
14       ________________________________________
15       ________________________________________
16     }
17
18     public static void Main(string[] args)
19     {
20       StreamReader reader = new StreamReader("input.txt");
21       double x;
22       do {
23         x = double.Parse(reader.ReadLine());
24         if(x != 0)
25         {
26           Update(x);
27           Console.WriteLine("{0:f3} {1:f3}", total, avr);
28         }
29       } while(____________________);
30       reader.Close();
31       Console.ReadLine();
32     }
33   }
34 }
```

สมมุติข้อมูลในแฟ้ม input.txt คือ

```
1
5
2.5
7
0
```

ผลลัพธ์ที่โปรแกรมแสดงทางหน้าจอคือ

```
1.000 1.000
6.000 3.000
8.500 2.833
15.500 3.875
```

แบบฝึกหัดที่ 4.7 : นับสินค้า

แฟ้มชื่อ input.txt เก็บข้อมูลสินค้าหลายประเภทไว้ในรูปแบบดังนี้

- บรรทัดแรกเก็บจำนวนเต็ม N แทนจำนวนสินค้าทั้งหมด
- อีก N บรรทัดถัดไป แต่ละบรรทัดเก็บประเภทของสินค้าแต่ละชิ้น โดยประเภทของสินค้ามี 3 แบบคือ A B และ C

ให้เขียนโปรแกรมอ่านแฟ้มดังกล่าว จากนั้นนับจำนวนสินค้าแต่ละประเภท แล้วพิมพ์ผลลัพธ์ออกมาในรูปแบบ X Y Z โดยที่ X แทนจำนวนชิ้นสินค้าประเภท A, Y แทนจำนวนชิ้นสินค้าประเภท B, และ Z แทนจำนวนชิ้นของสินค้าประเภท C

โปรแกรมที่เขียนได้คือ	สมมุติข้อมูลในแฟ้ม input.txt คือ
	6 A B A C A C
	ผลลัพธ์ที่โปรแกรมแสดงทางหน้าจอคือ
	3 1 2

แบบฝึกหัดที่ 4.8 : ดูคนทายเลข (โบนัส)

คุณสังเกตคนสองคนเล่นเกมทายเลขระหว่าง 1 ถึง 1000 ลักษณะของเกมก็เหมือนกับเกมทั่วไป กล่าวคือ คนทายเลขก็จะขานเลขจำนวนเต็มหนึ่งจำนวน คนตั้งโจทย์ก็จะบอกว่า มากกว่า (H) น้อยกว่า (L) หรือเท่ากับ (E)

คุณนั่งฟังและเก็บข้อมูลมาเป็นเวลาระยะหนึ่ง คุณสามารถสรุปผลออกมาได้ว่า (1) คุณทราบคำตอบ, (2) คุณทราบช่วงของคำตอบ (นั่นคือมีคำตอบที่เป็นไปได้มากกว่าหนึ่งค่า), หรือ (3) ไม่มีคำตอบ

ให้เขียนโปรแกรม ที่อ่านข้อมูลการเล่น จากนั้นให้โปรแกรมสรุปผล

รูปแบบของข้อมูลการเล่นจะเป็นดังนี้

- ในแต่ละรอบที่มีการทาย จะมีข้อมูลสองบรรทัด บรรทัดแรกระบุค่าที่ทาย เป็นจำนวนเต็มระหว่าง 1 ถึง 1000 อีกบรรทัดถัดไประบุผลลัพธ์ เป็นสตริง H แทนว่ามากกว่า L แทนว่าน้อยกว่า หรือ E แทนว่าเท่ากับ
- การจบการเล่น ระบุด้วย 0

รูปแบบผลลัพธ์เป็นดังนี้

- ถ้าทราบคำตอบ ให้พิมพ์ว่า Solution is X เมื่อ X คือคำตอบนั้น
- ถ้าไม่ทราบคำตอบแต่ทราบช่วงของคำตอบ ให้พิมพ์ว่า Solution in range X - Y เมื่อ X และ Y คือขอบเขตค่าที่เป็นไปได้
- ถ้าไม่มีทางมีคำตอบได้ ให้พิมพ์ว่า No solution

ตัวอย่างที่ 1	ตัวอย่างที่ 2	ตัวอย่างที่ 3	ตัวอย่างที่ 4
500 **L** **540** **H** **0** Solution in range 501 - 539	**500** **H** **540** **L** **0** No solution	**500** **L** **540** **E** **550** **H** **0** Solution is 540	**500** **L** **515** **L** **517** **H** **0** Solution is 516