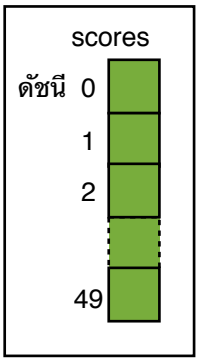
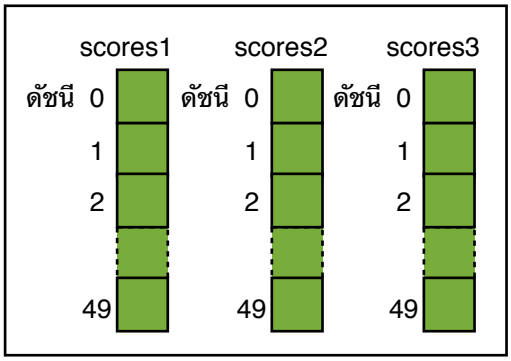
ชื่อ_____รหัสนิสิต_____หมู่_____

Lab 12 - อาร์เรย์ 2 มิติ

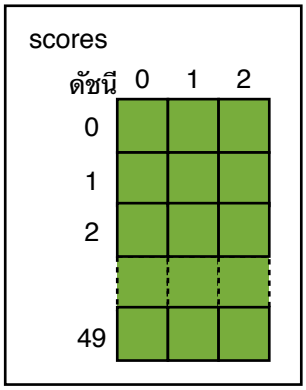
1. อาร์เรย์ 2 มิติ



ภาพที่ 1.1: อาร์เรย์ score



ภาพที่ 1.2: อาร์เรย์ score1, score2 และ score3



ภาพที่ 1.3: อาเรย์ 2 มิติ
ชื่อว่า score ที่มีขนาด 3 x 50 ช่อง

จาก Lab 11 นักเรียนได้เรียนรู้การใช้อาร์เรย์ไปแล้วซึ่งสามารถทำให้นักเรียนเก็บข้อมูลหลาย ๆ ตัวภายในโครงสร้างเดียวได้ เช่นต้องการเก็บคะแนนในการสอบของนักเรียน 50 คนสามารถเก็บไว้ในอาร์เรย์ดังนี้

```
int[ ] scores;
score = new int[50];
```

จากการประกาศอาร์เรย์ข้างต้นจะได้โครงสร้างดังภาพที่ 1.1 คราวนี้นักเรียนลองพิจารณาว่าถ้ามีการสอบทั้งหมด 3 ครั้ง เราก็ต้องประกาศอาเรย์ 3 ตัวแปรเพื่อเก็บข้อมูลคะแนนสอบของนักเรียนทั้ง 3 ครั้งของนักเรียน 50 คนดังนี้

```
int[] scores1;          //คะแนนสอบครั้งที่ 1 ของนักเรียนทั้ง 50 คน
score1 = new int[50];
int[] scores2;          //คะแนนสอบครั้งที่ 2 ของนักเรียนทั้ง 50 คน
score2 = new int[50];
int[] scores3;          //คะแนนสอบครั้งที่ 3 ของนักเรียนทั้ง 50 คน
score3 = new int[50];
```

จะได้อาเรย์ 3 ที่มีขนาด 50 ช่องดังภาพที่ 1.2

ซึ่งการประกาศอาเรย์ที่มีชื่อคล้ายกันดังภาพที่ 1.2 นั้นสามารถทำให้ง่ายขึ้นได้โดยการใช้อาเรย์ 2 มิติโดยหากต้องการอาเรย์ที่มีความยาวขนาด 50 ช่องจำนวน 3 ชุดสามารถใช้อาเรย์ 2 มิติโดยการประกาศง่าย ๆ ดังนี้

```
int[ , ] scores;
score = new int [3, 50];
```

จะมีโครงสร้างเป็นดังภาพที่ 1.3

2. การประกาศและการสร้างอาร์เรย์ 2 มิติ

เช่นเดียวกันกับอาร์เรย์ 1 มิติ ที่เราจะใช้งานได้เราจำเป็นต้องทำการประกาศและสร้างอาร์เรย์เสียก่อนมีรูปแบบดังนี้

2.1 การประกาศอาร์เรย์ 2 มิติ

การประกาศอาร์เรย์ 2 มิติมีรูปแบบดังนี้

```
ชนิดข้อมูล [ , ]  ชื่ออาร์เรย์ ;
```

โดยชนิดข้อมูลอาทิเช่น int, double หรือ char เป็นต้น ตัวอย่างเช่นหากต้องการประกาศอาร์เรย์สองมิติชื่อ numbers ซึ่งเป็นอาร์เรย์ที่เก็บข้อมูลชนิดจำนวนเต็มสามารถประกาศได้ดังนี้

```
int [ , ]  numbers;
```

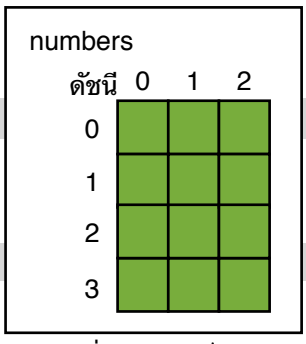
2.2 การสร้างอาร์เรย์ 2 มิติ

การสร้างอาร์เรย์ 2 มิติมีรูปแบบดังนี้

```
ชื่ออาร์เรย์ = new ชนิดของอาร์เรย์ [จำนวนแถวในอาร์เรย์ , จำนวนหลักในอาร์เรย์] ;
```

ตัวอย่างเช่นถ้าเราต้องการสร้างอาร์เรย์ numbers ที่เราได้ประกาศไว้แล้ว โดยที่อาร์เรย์ numbers นี้เป็นอาร์เรย์สองมิติที่มี 4 แถว แต่ละแถวมี 3 หลัก สามารถทำได้ดังนี้

```
numbers = new int[4, 3] ;
```



ภาพที่ 2.1: อาเรย์ 2 มิติ
ชื่อว่า numbers ที่มีขนาด 4 x 3 ช่อง

สรุปเพื่อให้เราสามารถใช้งานอาร์เรย์ 2 มิติได้นั้น เราจะต้องทำสองสิ่งคือการประกาศและการสร้างอาร์เรย์ดังนี้

```
ชนิดข้อมูล [ , ] ชื่ออาร์เรย์ ;
ชื่ออาร์เรย์ = new ชนิดของอาร์เรย์ [จำนวนสมาชิกในอาร์เรย์] ;
```

ทั้งนี้เราสามารถเขียนย่อได้ตามรูปแบบดังนี้

```
ชนิดข้อมูล [ , ] ชื่ออาร์เรย์ = new ชนิดของอาร์เรย์ [จำนวนแถวในอาร์เรย์ , จำนวนหลักในอาร์เรย์] ;
```

ตัวอย่างเช่นการประกาศและการสร้างอาร์เรย์ numbers โดยที่อาร์เรย์ numbers นี้เป็นอาร์เรย์สองมิติที่มี 4 แถว แต่ละแถวมี 3 หลัก สามารถทำได้ดังนี้

```
int [ , ] numbers;
numbers = new int[4, 3] ;
```

หรือแบบย่อดังนี้

```
int [ , ] numbers = new int[4, 3] ;
```

แบบฝึกหัดที่ 2.1 : การประกาศ สร้าง และกำหนดค่าเริ่มต้นให้กับอาร์เรย์

ให้นักเรียนประกาศและสร้างอาร์เรย์ต่อไปนี้

- อาร์เรย์สองมิติ ขนาด 120 x 90 (ตารางมิติแรกมีขนาด 120 ช่อง, มิติที่สองมีขนาด 90 ช่อง) ที่เก็บข้อมูลประเภท char ชื่อว่า 'name'
- อาร์เรย์สองมิติ ขนาด 1235 x 8 (ตารางมิติแรกมีขนาด 1235 ช่อง, มิติที่สองมีขนาด 8 ช่อง) ที่เก็บข้อมูลประเภท int ชื่อว่า 'std_ID'

ถ้านักเรียนประกาศได้ถูกต้อง โปรแกรมจะสามารถทำงานได้

เติมโปรแกรมต่อไปนี้ให้ถูกต้อง

```
using System;
class decl{
   static void Main(){
      double[,] lenght = new double[3,10];
      _________  name   ___________________;
      _________ std_ID  ___________________;
      int a = int.Parse(Console.ReadLine());
      int b = int.Parse(Console.ReadLine());
      int c = int.Parse(Console.ReadLine());
      int d = int.Parse(Console.ReadLine());
      name[a,b] = Convert.ToChar(Console.ReadLine());
      std_ID[c,d] = int.Parse(Console.ReadLine());
      Console.WriteLine(name[a,b]);
      Console.WriteLine(std_ID[c,d]);
   }

}
```

3. การกำหนดค่าเริ่มต้นให้กับอาร์เรย์

อาร์เรย์สองมิติที่เราประกาศและสร้างขึ้นมาในหัวข้อที่ 2 นั้นเราสามารถกำหนดค่าเริ่มต้นให้กับอาร์เรย์นั้น ๆ ได้ ซึ่งสามารถทำได้หลายรูปแบบตัวอย่างเช่นการประกาศพร้อมทั้งกำหนดค่าเริ่มต้นให้กับตัวแปรแบบอาเรย์ 2 มิติชื่อ MatrixA ขนาด 4 x 3 ที่มีสมาชิกในองค์ประกอบต่าง ๆ เป็นดังนี้

$$A = \begin{bmatrix} 4 & 3 & 2 \\ 6 & 7 & 10 \\ 11 & 2 & 5 \\ 8 & 0 & 12 \end{bmatrix}$$

สามารถทำได้ 3 รูปแบบดังนี้

รูปแบบที่ 1	รูปแบบที่ 2	รูปแบบที่ 3

int[,] MatrixA = new int[4,3] {
{4, 3, 2},
{6, 7, 10},
{11, 2, 5},
{8, 0, 12}
};
	int[,] MatrixA = new int[,] {
	{4, 3, 2},
	{6, 7, 10},
	{11, 2, 5},
	{8, 0, 12}
	};
		int[,] MatrixA = {
		{4, 3, 2},
		{6, 7, 10},
		{11, 2, 5},
		{8, 0, 12}
		};

แบบฝึกหัดที่ 3.1 : การประกาศ สร้าง และกำหนดค่าเริ่มต้นให้กับอาร์เรย์

จงประกาศ สร้าง พร้อมทั้งกำหนดค่าเริ่มต้นให้กับตัวแปรแบบอาเรย์ 2 มิติชนิด double ชื่อ data ซึ่งมีขนาด 2 แถว 4 หลัก และมีการกำหนดค่าเริ่มต้นเป็นดังภาพที่ 3.1

data

ดัชนี	0	1	2	3
0	1.5	5.7	8.2	6.7
1	4.9	9.8	3.2	7.1

ภาพที่ 3.1: อาเรย์ data

4. โครงสร้างและการอ้างถึงข้อมูลในอาร์เรย์

เนื่องจากอาเรย์ 2 มิตินั้นแบ่งข้อมูลเป็นแถวและหลักอย่างชัดเจน ฉะนั้นการอ้างถึงข้อมูลจึงจำเป็นต้องใช้ดัชนีของทั้งแถวและหลักเข้ามาร่วมในการอ้างอิงด้วยดังนี้

ชื่ออาร์เรย์ [ดัชนีของแถว, ดัชนีของหลัก]

ตัวอย่างเช่นจากข้อมูล data ในภาพที่ 3.1 หากเราต้องการเข้าถึงข้อมูล 5.7 เราต้องใช้ data[0, 1] ในการอ้างถึงเป็นต้น

point

4	14	6	11	-5
-12	7	5	3	-9
10	-32	1	0	15

ภาพที่ 4.1: อาเรย์ point

แบบฝึกหัดที่ 4.1 : การอ้างถึงข้อมูลในอาร์เรย์

กำหนดอาร์เรย์ 2 มิติชื่อ point มีขนาด 3 x 5 ที่ประกอบไปด้วยข้อมูลประเภท double ดังภาพที่ 4.1 ให้นักเรียนเติมโปรแกรมต่อไปนี้ให้พิมพ์ผลลัพธ์ดังที่กำหนดให้

เติมโปรแกรมต่อไปนี้	ผลลัพธ์ที่ต้องการ

`using System;`
`class decl`
`{`
`  static void Main()`
`  {`
`  double[,] point = { {4, 14, 6, 11, -5},`
`  {-12, 7, 5, 3, -9},`
`  {10, -32, 1, 0, 15} };`
 	
`  Console.WriteLine(point_____);`
`  Console.WriteLine(point_____);`
`  Console.WriteLine(point_____);`
`  Console.WriteLine(point_____);`
`  Console.WriteLine(point_____);`
`  }`
`}`
	-5
	5
	11
	4
	1

5. คำสั่งวนซ้ำ 2 ชั้น

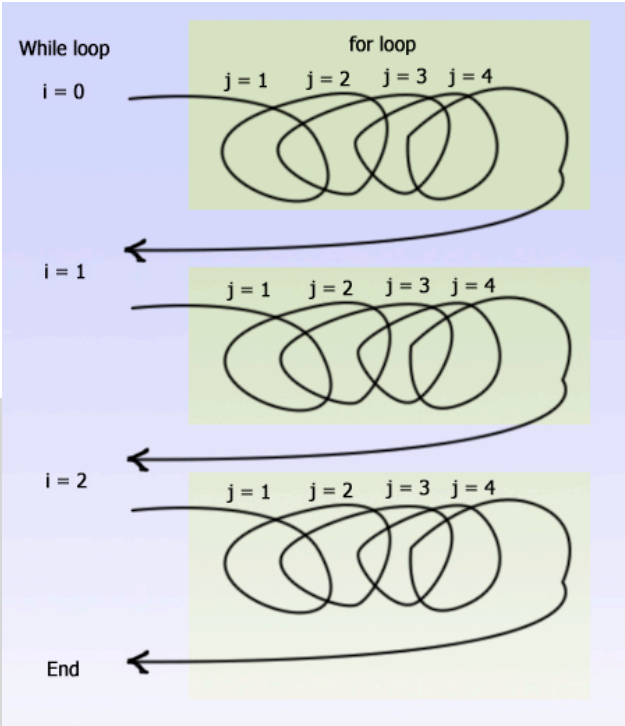
คำสั่งวนซ้ำทุกรูปแบบสามารถเขียนซ้อนกันได้ กล่าวคือในคำสั่งวนซ้ำอันหนึ่ง (อันนอก) มีคำสั่งวนซ้ำอีกอันหนึ่ง (อันใน) อยู่ การทำงานของโปรแกรมจะทำงานโดยให้พิจารณาว่าในคำสั่งวนซ้ำอันนอกแต่ละรอบ จะมีการทำงานของคำสั่งวนซ้ำอันในตั้งแต่ต้นจนจบอันในอยู่ทุกรอบไป ดังตัวอย่างส่วนของโปรแกรมที่ 5.1 จะมีลักษณะการทำงานดังภาพที่ 5.1

ตัวอย่างส่วนของโปรแกรมที่ 5.1 คำสั่งวนซ้ำ 2 ชั้น

```
int i = 0;
while(i<3){
   for(int j=1; j<=4; j=j+1)
   {
      Console.WriteLine("{0},{1}",i,j);
   }
   i = i+1;
}
```

ทำให้ผลลัพธ์ที่ได้จากตัวอย่างส่วนของโปรแกรมที่ 5.1 เป็นดังนี้

```
0,1
0,2
0,3
0,4
1,1
1,2
1,3
1,4
2,1
2,2
2,3
2,4
```



ภาพที่ 5.1 : ลักษณะการทำงานของตัวอย่างส่วนของโปรแกรมที่ 5.1

แบบฝึกหัดที่ 5.1 : ฝึกหัดคำสั่ง for สองชั้น ภาค 1

โปรแกรมต่อไปนี้ รับจำนวนเต็ม n จากนั้นพิมพ์คู่ลำดับของจำนวนเต็ม i j ที่ $0 \le i \le n - 1$ และ $0 \le j \le n - 1$ ทุก ๆ คู่ ไล่ไปตามลำดับดังตัวอย่างผลลัพธ์ที่กำหนดให้

เติมข้อความที่เหมาะสมในโปรแกรมต่อไปนี้	ผลลัพธ์ที่ต้องการ

`using System;`
`class example`
`{`
`  public static void Main(string[] args)`
`  {`
`  int n = int.Parse(Console.ReadLine());`
`  for(int i = 0; __________ ; i++)`
`  for(int j = 0; __________ ; j++)`
`  Console.WriteLine("{0}, {1}",i,j);`
`  }`
`}`
	3
	0, 0
	0, 1
	0, 2
	1, 0
	1, 1
	1, 2
	2, 0
	2, 1
	2, 2

แบบฝึกหัดที่ 5.2 : ฝึกหัดคำสั่ง for สองชั้น ภาค 2

โปรแกรมต่อไปนี้ รับจำนวนเต็ม n จากนั้นพิมพ์คู่ลำดับของจำนวนเต็ม (i, j) ที่ $0 \le i \le n - 1$ และ $0 \le j \le n - 1$ เฉพาะที่ $j \le i$ ไล่ไปตามลำดับดังตัวอย่างผลลัพธ์ที่กำหนดให้

เติมข้อความที่เหมาะสมในโปรแกรมต่อไปนี้	ตัวอย่างผลลัพธ์ที่ 1	ตัวอย่างผลลัพธ์ที่ 2

`using System;`
`class example`
`{`
`  public static void Main(string[] args)`
`  {`
`  int n = int.Parse(Console.ReadLine());`
`  for(__________; __________ ; __________)`
`  for(__________; __________ ; __________)`
`  Console.WriteLine("{0}, {1)",i,j);`
`  }`
`}`
	3
	0, 0
	1, 0
	1, 1
	2, 0
	2, 1
	2, 2
		4
		0, 0
		1, 0
		1, 1
		2, 0
		2, 1
		2, 2
		3, 0
		3, 1
		3, 2
		3, 3

แบบฝึกหัดที่ 5.3 : คำสั่ง for วาดสามเหลี่ยมง่าย

เขียนโปรแกรมที่อ่านขนาดของสามเหลี่ยม จากนั้นพิมพ์สามเหลี่ยมด้วยอักขระ x ให้มีลักษณะตามแสดงในตัวอย่าง

โปรแกรมที่เขียนได้คือ	ผลลัพธ์ที่ต้องการ
	ตัวอย่างที่ 1 Input n : 2 x xx **ตัวอย่างที่ 2** Input n : 4 x xx xxx xxxx

แบบฝึกหัดที่ 5.4 : คำสั่ง for วาดสามเหลี่ยมยากขึ้น

เขียนโปรแกรมที่อ่านขนาดของสามเหลี่ยม จากนั้นพิมพ์สามเหลี่ยมด้วยอักขระ x ให้มีลักษณะตามแสดงในตัวอย่าง

โปรแกรมที่เขียนได้คือ	ผลลัพธ์ที่ต้องการ
	ตัวอย่างที่ 1 Input n : 2 x xxx **ตัวอย่างที่ 2** Input n : 4 x xxx xxxxx xxxxxxx

แบบฝึกหัดที่ 5.5 : เติมค่าในอาร์เรย์ 1

ส่วนที่หายไปในโปรแกรมต่อไปนี้คือโปรแกรมที่กำหนดค่าให้กับอาร์เรย์ a ที่มีขนาดแตกต่างกัน ให้นักเรียนเติมส่วนที่ขาดหายไปในโปรแกรมเพื่อให้โปรแกรมกำหนดค่าให้กับอาร์เรย์ a และแสดงผลลัพธ์ตามตัวอย่าง

โปรแกรมที่เขียนได้คือ	ผลลัพธ์ที่ต้องการ

`using System;`
 	
`class MainClass`
`{`
`  public static void Main(string[] args)`
`  {`
`  int n = int.Parse(Console.ReadLine());`
`  int [,] a = new int[n,n];`
`  _____`
`  _____`
`  _____`
`  _____`
`  _____`
`  for(int i=0; i<n; i++) {`
`  for(int j=0; j<n; j++)`
`  Console.Write("{0} ",a[i,j]);`
`  Console.WriteLine();`
`  }`
`  }`
`}`
	ตัวอย่างที่ 1
	3
	1 2 3
	2 3 4
	3 4 5
	ตัวอย่างที่ 2
	5
	1 2 3 4 5
	2 3 4 5 6
	3 4 5 6 7
	4 5 6 7 8
	5 6 7 8 9

แบบฝึกหัดที่ 5.6 : เติมค่าในอาร์เรย์ 2

ส่วนที่หายไปในโปรแกรมต่อไปนี้คือโปรแกรมที่กำหนดค่าให้กับอาร์เรย์ a ที่มีขนาดแตกต่างกัน ให้นักเรียนเติมส่วนที่ขาดหายไปในโปรแกรมเพื่อให้โปรแกรมกำหนดค่าให้กับอาร์เรย์ a และแสดงผลลัพธ์ตามตัวอย่าง

โปรแกรมที่เขียนได้คือ

```
using System;

class MainClass
{
    public static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());
        int [,] a = new int[n,n];

        ____________________________________________
        ____________________________________________
        ____________________________________________
        ____________________________________________
        ____________________________________________

        for(int i=0; i<n; i++) {
            for(int j=0; j<n; j++)
                Console.Write("{0} ",a[i,j]);
            Console.WriteLine();
        }
    }
}
```

ผลลัพธ์ที่ต้องการ

ตัวอย่างที่ 1

```
3
3 4 5
2 3 4
1 2 3
```

ตัวอย่างที่ 2

```
5
5 6 7 8 9
4 5 6 7 8
3 4 5 6 7
2 3 4 5 6
1 2 3 4 5
```

แบบฝึกหัดที่ 5.7 : เมธอด RowSum

เขียนเมธอด RowSum ที่รับพารามิเตอร์ A และ r โดยที่อาร์เรย์ A มีขนาด 5 x 5 จากนั้นคืนผลรวมของข้อมูลในแถวที่ r. ยกตัวอย่างเช่น ถ้า อาร์เรย์ A ถูกกำหนดให้มีค่าเริ่มต้นด้วยส่วนของโปรแกรมด้านล่าง

```
int [,] A = new int [5,5] { {1,2,3,4,5},
                            {6,7,8,9,10},
                            {11,12,13,14,15},
                            {16,17,18,19,20},
                            {21,22,23,24,25}
};
```

เมื่อเราเรียก RowSum(A,2) ผลลัพธ์ที่ได้คือ 65 (ซึ่งมีค่าเท่ากับ 11+12+13+14+15).

หมายเหตุ: ในการทดสอบเมธอดดังกล่าว นักเรียนจะต้องเขียนเมธอดประกอบอื่น ๆ เช่น เมธอด Main ขึ้นเองเพื่อเรียกใช้เมธอด RowSum นักเรียนสามารถใช้ส่วนของโปรแกรมที่กำหนดค่าให้กับอาร์เรย์ด้านบนประกอบได้ด้วย

เมธอด RowSum ที่เขียนได้คือ

```
static int RowSum(int[,] A, int r)
{





}
```

แบบฝึกหัดที่ 5.8 : เมธอด MaxRowSum

จงเขียนเมธอด MaxRowSum ที่รับอาร์เรย์ A ขนาด 5 x 5 แล้วคืนผลรวมของข้อมูลในแถวที่มากที่สุด

ในการเขียนเมธอดดังกล่าว คุณสามารถเรียกใช้เมธอด RowSum ได้ โดยไม่ต้องเขียนใหม่ ยกตัวอย่างเช่น ถ้าอาร์เรย์ A ถูกกำหนดให้มีค่าเริ่มต้นด้วยส่วนของโปรแกรมด้านล่าง

```
int [,] A = new int [5,5] { {1,2,3,4,5},
                            {6,7,8,9,10},
                            {21,22,23,24,25},
                            {11,12,13,14,15},
                            {16,17,18,19,20}
};
```

ผลลัพธ์ของการเรียก MaxRowSum(A) คือ 115 (ซึ่งเป็นผลรวมของบรรทัดที่แถวหมายเลข 3).

หมายเหตุ: ในการทดสอบเมธอดดังกล่าว คุณจะต้องเขียนเมธอดประกอบอื่น ๆ เช่น เมธอด Main ขึ้นเอง เพื่อเรียกใช้เมธอด MaxRowSum คุณสามารถใช้ส่วนของโปรแกรมที่กำหนดค่าให้กับอาร์เรย์ด้านบนประกอบได้ด้วย

เมธอด MaxRowSum ที่เขียนได้คือ

`static int MaxRowSum(int[,] A)`
`{`
 	
`}`

6. การหาจำนวนแถวและจำนวนหลักของอาร์เรย์ 2 มิติ

ในตัวแปรแบบอาเรย์ 1 มิติ เราใช้คำสั่ง .Length ในการหาจำนวนช่องของอาเรย์ แต่สำหรับอาเรย์ 2 มิติ ถ้าต้องการทราบว่าอาเรย์ 2 มิตินี้มีจำนวนแถวกี่แถว หรือจำนวนหลักกี่หลัก สามารถทำได้โดยใช้คำสั่ง .GetLength() ดังนี้

คำสั่งในการหาจำนวนแถวของอาเรย์ 2 มิติใช้ **ชื่ออาร์เรย์.GetLength(0)**

คำสั่งในการหาจำนวนหลักของอาเรย์ 2 มิติใช้ **ชื่ออาร์เรย์.GetLength(1)**

point				
4	14	6	11	-5
-12	7	5	3	-9
10	-32	1	0	15

ภาพที่ 4.1: อาเรย์ point

ตัวอย่างเช่นจากอาร์เรย์ point ในภาพที่ 4.1 หากเรียกใช้คำสั่ง point.GetLength(0) จะได้ค่าเท่ากับ 3 หากเรียกใช้คำสั่ง point.GetLength(1) จะได้ค่าเท่ากับ 5 และหากเรียนใช้คำสั่ง point.Length จะได้ค่าเท่ากับ 15 เป็นต้น

แบบฝึกหัดที่ 6.1 : เมธอด ShowMatrix

เราจะเขียนเมธอดชื่อ ShowMatrix เพื่อแสดงข้อมูลภายในเมตริกซ์ที่ประกอบไปด้วยจำนวนเต็ม ออกทางหน้าจอ เมธอดจะรับเมตริกซ์ในรูปของพารามิเตอร์แบบอาเรย์สองมิติของ int

จงเติมส่วนที่ขาดหายไปของเมธอดเพื่อให้โปรแกรมทำงานได้อย่างสมบูรณ์ ตามตัวอย่างของผลลัพธ์

โปรแกรมที่เขียนได้คือ

```
using System;
class Matrix
{
    static ________ ShowMatrix(_______________)
    {
        for (int i = 0; i < ________; i++)
        {
            for (int j = 0; j < ________; j++)
            {
                Console.Write("{0,4}", __________);
            }
            Console.WriteLine();
        }
    }

    static void Main()
    {
        int [,] A = { { 5, 3,  8},
                      { 2, 6, 10},
                      { 1, 8, 25},
                      {12, 3, 30}
                    };
        int [,] B = { { 1, 2, 3, 4 }, { 5, 6, 7, 8 } };
        ShowMatrix(A);
        Console.WriteLine();
        ShowMatrix(B);
    }
}
```

ผลลัพธ์ที่ได้จากโปรแกรม

```
   5   3   8
   2   6  10
   1   8  25
  12   3  30

   1   2   3   4
   5   6   7   8
```

แบบฝึกหัดที่ 6.2 : เมธอด ReadMatrix

ในคาบบรรยายเราได้ดูตัวอย่างเมธอด ReadMatrix ที่อ่านเมตริกซ์ของจำนวนจริงมาแล้ว สำหรับข้อนี้เราจะเขียนเมธอด ReadMatrix ที่อ่านเมทริกซ์ที่ประกอบไปด้วยจำนวนเต็มแทน

เมธอดดังกล่าวจะรับพารามิเตอร์ r และ c แทนจำนวนแถวและจำนวนหลัก จากนั้นจะอ่านข้อมูลจากผู้ใช้ บรรทัดละจำนวน โดยไล่จาก ข้อมูลแถวที่ 1 คอลัมน์ที่ 1, ข้อมูลแถวที่ 1 คอลัมน์ที่ 2, จนจบแถวแรก แล้วรับข้อมูลในแถวที่สอง สาม สี่ต่อไปเรื่อย ๆ

ในการทดสอบเมธอดดังกล่าวกับโปรแกรมหลักด้านล่าง ให้นิสิตคัดลอกเมธอด ShowMatrix จากแบบฝึกหัดที่ 5.9 มาด้วย แต่ในการส่งไม่จำเป็นต้องใช้เมธอดนี้

โปรแกรมที่เขียนได้คือ

```
using System;

class Matrix
{
     /* คัดลอกเมธอด ShowMatrix จากข้อก่อนหน้ามาวางในตำแหน่งนี้ */

    static int[,] ReadMatrix(int r, int c)
    {
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
    }

    static void Main()
    {
        int num_rows, num_cols;
        int[,] A;

        Console.Write("Enter number of rows: ");
        num_rows = int.Parse(Console.ReadLine());
        Console.Write("Enter number of columns: ");
        num_cols = int.Parse(Console.ReadLine());
        A = ReadMatrix(num_rows, num_cols);
        Console.WriteLine("Matrix A is");
        ShowMatrix(A);
    }
}
```

ผลลัพธ์ที่ได้จากโปรแกรม

```
Enter number of rows: 3
Enter number of columns: 2
1
5
3
4
9
10
Matrix A is
   1   5
   3   4
   9  10
   ____________________
```

ตัวอย่างที่ 2

```
Enter number of rows: 2
Enter number of columns: 2
10
20
30
40
Matrix A is
  10  20
  30  40
```

แบบฝึกหัดที่ 6.3 : เมธอด TransposeMatrix

จงเขียนเมธอด TransposeMatrix ที่รับเมตริกซ์ m ที่มีขนาด r แถว c คอลัมน์ จากนั้นคืนเมตริกซ์ k ที่เป็นเมตริกซ์ m ที่ถูกทราสไพส (transpose) แล้ว ที่มีขนาด c แถว r คอลัมน์

ในการทดสอบเมธอดดังกล่าวกับโปรแกรมหลักด้านล่าง ให้นิสิตคัดลอกเมธอด ShowMatrix และ ReadMatrix จากข้อก่อน ๆ มาด้วย แต่ในการส่ง ไม่จำเป็นต้องใช้เมธอดนี้

โปรแกรมที่เขียนได้คือ

```
using System;

class Matrix
{
     /* คัดลอกเมธอด ShowMatrix และ ReadMatrix */
     /*    จากข้อก่อนหน้ามาวางในตำแหน่งนี้          */

    static int[,] TransposeMatrix(int [,] m)
    {
        int [,] k = new int[_________, m.GetLength(0)];
        for(int i = 0; i < m.GetLength(0); i++)
            for(__________; ___________; j++)
                k[_______] = m[_______];
        return ;
    }

    static void Main()
    {
        int num_rows, num_cols;
        int [,] mat1;
        int [,] mat2;

        Console.Write("Enter number of rows: ");
        num_rows = int.Parse(Console.ReadLine());
        Console.Write("Enter number of columns: ");
        num_cols = int.Parse(Console.ReadLine());
        mat1 = ReadMatrix(num_rows, num_cols);
        Console.WriteLine("Matrix A is");
        ShowMatrix(mat1);
        Console.WriteLine("Matrix A transpose is");
        mat2 = TransposeMatrix(mat1);
        ShowMatrix(mat2);
    }
}
```

ผลลัพธ์ที่ได้จากโปรแกรม

```
Enter number of rows: 3
Enter number of columns: 2
1
5
3
4
9
10
Matrix A is
   1   5
   3   4
   9  10
Matrix A transpose is
   1   3   9
   5   4  10
```

ตัวอย่างที่ 2

```
Enter number of rows: 2
Enter number of columns: 2
10
20
30
40
Matrix A is
  10  20
  30  40
Matrix A transpose is
  10  30
  20  40
```

แบบฝึกหัดที่ 6.4 : เมธอด SumDiag

สำหรับข้อนี้เราจะเขียนเมธอด SumDiag ที่รับเมตริกซ์ขนาด n x n แล้วหาผลรวมของข้อมูลในแนวเส้นแทยงมุม (นั่นคือ ข้อมูลในตำแหน่งแถวที่ 1 คอลัมน์ 1, แถวที่ 2 คอลัมน์ 2, แถวที่ 3 คอลัมน์ 3, ไปเรื่อย ๆ)

เมธอดดังกล่าวรับพารามิเตอร์ m เป็นอาร์เรย์ของจำนวนเต็มแทนเมตริกซ์จัตุรัส แล้วคือผลรวมดังกล่าว

ในการทดสอบเมธอดดังกล่าวกับโปรแกรมหลักด้านล่าง ให้นิสิตคัดลอกเมธอด ReadMatrix จากข้อที่แล้วมาด้วย แต่ในการส่ง ไม่จำเป็นต้องใช้เมธอดนี้

หมายเหตุ: จากตัวอย่างผลลัพธ์ที่ได้จากโปรแกรมข้อมูลที่อยู่ที่เส้นทแยงมุมคือ 1 9 และ 40 ดังนั้นผลรวมคือ 50

โปรแกรมที่เขียนได้คือ

```
using System;

class Matrix
{
     /* คัดลอกเมธอด ReadMatrix จากข้อก่อนหน้ามาวางในตำแหน่งนี้  */

    static int SumDiag(int [,] m)
    {
        ____________________________________________
        ____________________________________________
        ____________________________________________
        ____________________________________________
    }

    static void Main()
    {
        int n, s;
        int[,] mat;

        Console.Write("Enter n: ");
        n = int.Parse(Console.ReadLine());
        mat = ReadMatrix(n,n);
        s = SumDiag(mat);
        Console.WriteLine("Sum of its diagonal is {0}", s);
    }
}
```

ผลลัพธ์ที่ได้จากโปรแกรม

```
Enter n: 3
1
5
3
4
9
10
20
30
40
Sum of its diagonal is 50
```