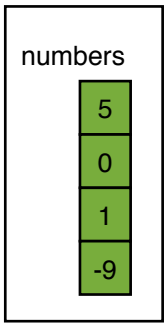
ชื่อ_____รหัสนิสิต_____หมู่_____

Lab 11 - อาร์เรย์

นักเรียนได้เรียนเรื่องลิสต์ในภาษาไพทอนไปแล้ว โดยลิสต์ทำให้เราสามารถรวมกลุ่มของข้อมูลหลายข้อมูลเข้าไว้ด้วยกัน ในภาษา C# ก็มีการจัดการกับข้อมูลที่คล้ายกับการใช้ลิสต์ในภาษาไพทอนเช่นเดียวกัน นั้นก็คือ “อาร์เรย์” นั้นเอง



ภาพที่ 1.1: อาร์เรย์ numbers

1. อะไรคืออาร์เรย์

อาร์เรย์ (Array) คือโครงสร้างของข้อมูลที่รวบรวมข้อมูลที่มีชนิดเดียวกันเข้าไว้ด้วยกัน (คล้ายกับลิสต์ในไพธอนแต่ลิสต์ในไพทอนสามารถรวมรวมข้อมูลหลากหลายชนิดเข้าไว้ด้วยกันได้) ก่อนที่อาร์เรย์จะสามารถใช้งานได้ในภาษา C# นั้นจะประกอบไปด้วยสองขั้นตอนก่อนเริ่มใช้งาน คือ 1.ประกาศอาร์เรย์ และ 2.การสร้างอาร์เรย์

1.1 การประกาศอาร์เรย์

การประกาศอาร์เรย์มีรูปแบบดังนี้

```
ชนิดข้อมูล [ ]  ชื่ออาร์เรย์ ;
```

โดยชนิดข้อมูลอาทิเช่น int, double หรือ char เป็นต้น ตัวอย่างเช่นหากต้องการประกาศอาร์เรย์ชื่อ numbers ซึ่งเป็นอาร์เรย์ที่เก็บข้อมูลชนิดจำนวนเต็มสามารถประกาศได้ดังนี้

```
int [ ]  numbers;
```

1.2 การสร้างอาร์เรย์

เมื่อเราประกาศอาร์เรย์เสร็จแล้วเรายังไม่สามารถใช้อาร์เรย์นั้น ๆ ได้ ทั้งนี้เพราะการประกาศอาร์เรย์เปรียบเสมือนเราได้ทำป้ายบ้านมาเตรียมปักลงบนที่ดิน แต่ยังไม่ได้ซื้อที่ดินและยังไม่ได้สร้างบ้าน เราจึงจำเป็นต้องซื้อที่ดินและสร้างบ้านก่อน ซึ่งก็คือการสร้างอาร์เรย์นั้นเอง โดยการสร้างอาร์เรย์นั้นมีรูปแบบดังนี้

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

ตัวอย่างเช่นถ้าเราต้องการสร้างอาร์เรย์ numbers ที่เราได้ประกาศไว้แล้ว โดยที่อาร์เรย์ numbers นี้มีสมาชิกเป็นเลขจำนวนเต็มทั้งหมด 4 จำนวน สามารถทำได้ดังนี้

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

สรุปเพื่อให้เราสามารถใช้งานอาร์เรย์ได้นั้น เราจะต้องทำสองสิ่งคือการประกาศและการสร้างอาร์เรย์ดังนี้

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

ทั้งนี้เราสามารถเขียนย่อได้ตามรูปแบบดังนี้

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

ตัวอย่างเช่นการประกาศและการสร้างอาร์เรย์ numbers ที่มีสมาชิกเป็นจำนวนเต็มจำนวน 4 จำนวนนั้นสามารถเขียนย่อได้ดังนี้

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

2. การกำหนดค่าเริ่มต้นให้กับอาร์เรย์

อาร์เรย์ที่เราประกาศและสร้างขึ้นมาในหัวข้อที่ 1 นั้นเราสามารถกำหนดค่าเริ่มต้นให้กับอาร์เรย์นั้น ๆ ได้ ซึ่งสามารถทำได้หลายรูปแบบดังต่อไปนี้

รูปแบบที่ 1

```
 ชื่ออาร์เรย์  =  new  ชนิดของอาร์เรย์ [จำนวนสมาชิกในอาร์เรย์] {ข้อมูลสมาชิกตัวที่ 1, ข้อมูลสมาชิกตัวที่ 2, ...} ;
```

รูปแบบที่ 2

```
 ชื่ออาร์เรย์  =  new  ชนิดของอาร์เรย์ [ ] {ข้อมูลสมาชิกตัวที่ 1, ข้อมูลสมาชิกตัวที่ 2, ...} ;
```

รูปแบบที่ 3

```
ชื่ออาร์เรย์  =  {ข้อมูลสมาชิกตัวที่ 1, ข้อมูลสมาชิกตัวที่ 2, ...} ;
```

ตัวอย่างหากเราต้องการสร้างอาร์เรย์ numbers ที่มีข้อมูลค่าเริ่มต้นเป็นดังภาพที่ 1.1 สามารถทำได้ 3 รูปแบบดังต่อไปนี้

รูปแบบที่ 1

```
numbers =  new  int [4] {5, 0, 1, -9} ;
```

รูปแบบที่ 2

```
numbers =  new  int [ ] {5, 0, 1, -9} ;
```

รูปแบบที่ 3

```
numbers =  {5, 0, 1, -9} ;
```

แบบฝึกหัดที่ 2.1 : การประกาศ สร้าง และกำหนดค่าเริ่มต้นให้กับอาร์เรย์

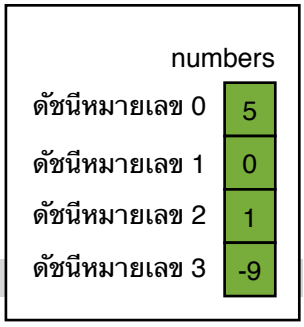
ให้นักเรียนเขียนคำสั่งที่ใช้ในการประกาศ สร้าง และกำหนดค่าเริ่มต้นให้กับอาร์เรย์ที่มีชื่อ ชนิดของข้อมูล และ ค่าเริ่มต้นเป็นดังที่กำหนดให้ต่อไปนี้

	ชื่ออาร์เรย์	ชนิดของข้อมูล	ค่าเริ่มต้น	คำสั่งที่เขียนได้คือ
1	score	double	6.5, 7.9, 4.3, 9.1, 8.8	
2	letter	char	'm', 'c'	
3	carbrand	string	"Honda", "Toyota", "Mazda", "BMW"	

3. โครงสร้างและการอ้างถึงข้อมูลในอาร์เรย์

เนื่องจากอาร์เรย์หนึ่งตัวอาจมีสมาชิกมากกว่าหนึ่งตัว ฉะนั้นการอ้างถึงข้อมูลในอาร์เรย์จึงจำเป็นต้องใช้ดัชนีระบุช่องของอาร์เรย์กำกับเช่นเดียวกับการใช้ลิสต์ในภาษาไพทอน โดยดัชนีของแต่ละสมาชิกในอาร์เรย์จะเริ่มที่หมายเลขศูนย์เช่นเดียวกันกับลิสต์ในภาษาไพทอน ตัวอย่างเช่น หากเราประกาศ สร้าง และกำหนดค่าเริ่มต้น ของอาร์เรย์ numbers เป็น

```
int[ ]  numbers = new int [4] {5, 0, 1, -9} ;
```



ภาพที่ 3.1: อาร์เรย์ numbers

จะได้อาร์เรย์ numbers ที่มีโครงสร้างและข้อมูลเป็นดังภาพที่ 3.1 ฉะนั้นหากเราต้องการอ้างถึงข้อมูลใดก็ใช้ชื่ออาร์เรย์กับดัชนีของช่องนั้น ๆ กำกับเหมือนกับลิสต์ในภาษาไพทอน ดังตัวอย่างต่อไปนี้

การทำงาน	คำสั่ง
แสดงค่าของสมาชิกดัชนีหมายเลข 2 ของอาร์เรย์ numbers ออกมาทางหน้าจอ (ซึ่งมีค่าเท่ากับ 1)	Console.WriteLine(numbers[2]) ;
หาผลรวมของสมาชิกตัวแรกกับตัวสุดท้ายของอาร์เรย์ numbers เก็บใส่ตัวแปร sum	sum = numbers[0] + number[3] ;
เปลี่ยนค่าของอาร์เรย์ในช่องดัชนีหมายเลข 1 ให้เป็น 15	numbers[1] = 15 ;

*หมายเหตุ คำสั่งที่ใช้ระบุจำนวนสมาชิกของอาร์เรย์ในภาษา C# ใช้คำสั่ง <ชื่ออาร์เรย์>.length ซึ่งทำงานเช่นเดียวกันกับคำสั่ง len[ชื่อลิสต์] ในภาษาไพทอน จากอาร์เรย์ numbers ข้างต้น หากใช้คำสั่ง numbers.length จะมีค่าเท่ากับ 4 (เนื่องจากมีสมาชิก 4 จำนวน)

แบบฝึกหัดที่ 3.1 : การอ้างถึงข้อมูลในอาร์เรย์

กำหนดให้อาร์เรย์ arr ถูกประกาศ สร้าง และกำหนดค่าเริ่มต้นดังนี้

```
int [ ] arr = {1,3,7,10,15,20,19};
```

จงเติมข้อความที่เหมาะสมลงในช่องว่างของโปรแกรมแต่ละคำสั่งให้ผลลัพธ์เป็นตามที่ระบุ

การทำงาน	คำสั่ง
ต้องการให้พิมพ์ 15	Console.WriteLine(arr[_____]) ;
ต้องการให้พิมพ์ 19	Console.WriteLine(arr[_____]) ;
ต้องการให้พิมพ์ 29	Console.WriteLine(arr[_____] + arr[_____]) ;
พิมพ์ขนาดของอาร์เรย์ arr	Console.WriteLine(arr._____);

4. คำสั่งวนซ้ำแบบ for

เราได้เรียนคำสั่งวนซ้ำแบบ while ไปแล้ว ซึ่งหากนักเรียนสังเกตจะพบว่ามีหลายครั้งที่เราใช้คำสั่งแบบ while ในลักษณะคล้ายกับระเบิดเวลา กล่าวคือ จะมีตัวแปรที่คอยนับว่าทำคำสั่ง while ไปทั้งหมดกี่รอบแล้ว และตัวแปรที่ใช้นับนี้จะค่อย ๆ เปลี่ยนแปลงค่า (ค่อย ๆ มากขึ้นหรือน้อยลง) จนกระทั่งถึงจุดหนึ่ง ค่าของตัวแปรนี้ก็จะมากพอ (หรือน้อยพอ) ที่จะทำให้เงื่อนไขหลัง while เป็นเท็จ ทำให้หยุดการทำคำสั่งวนซ้ำใน while ได้ อาทิเช่น โปรแกรมพิมพ์เลขที่หารด้วย 3 ลงตัวตั้งแต่ 0 ถึง 10 อาจเขียนด้วยคำสั่ง while ได้ตัวอย่างส่วนของโปรแกรมที่ 4.1

ตัวอย่างส่วนของโปรแกรมที่ 4.1 ส่วนของโปรแกรมที่ใช้ในการแสดงเลขที่หารด้วย 3 ลงตัว โดยใช้ while

```
1 i = 0;
2 while(i<= 10)
3 {
4    if(i%3 == 0)
5    {
6       Console.WriteLine(i);
7    }
8    i = i + 1;
9 }
```

จากตัวอย่างส่วนของโปรแกรมที่ 4.1 นั้นมีส่วนที่สำคัญอยู่สามส่วน นั้นคือ

1. การกำหนดค่าเริ่มต้นของตัวนับ (ในที่นี้คือตัวแปร i) ซึ่งถูกเขียนไว้ในคำสั่งในบรรทัดที่หนึ่ง

2. การปรับค่าของตัวนับ (ในส่วนของโปรแกรมนี้ปรับโดยเพิ่มค่าขึ้นครั้งละหนึ่ง) ซึ่งถูกเขียนไว้ในบรรทัดที่แปด

3. การกำหนดเงื่อนไขในการทำซ้ำ (ในส่วนของโปรแกรมจะทำซ้ำเรื่อย ๆ ตราบใดที่ค่าของตัวนับยังไม่มากกว่า 10) ซึ่งถูกเขียนไว้เป็นเงื่อนไขหลัง while ในบรรทัดที่สอง

เนื่องจากการใช้คำสั่ง while ในลักษณะระเบิดเวลานี้มีขึ้นบ่อยครั้ง ภาษา C# จึงออกแบบคำสั่งวนซ้ำแบบ for เพื่อให้ง่ายต่อการใช้คำสั่งวนซ้ำในลักษณะของระเบิดเวลา โดยมีรูปแบบดังนี้

```
for( กำหนดค่าเริ่มต้นของตัวนับ ;  เงื่อนไขในการทำซ้ำ ;  การปรับค่าของตัวนับหลังจบการทำซ้ำในแต่ละรอบ )
{
    ส่วนในการเขียนโปรแกรมสำหรับคำสั่งต่าง ๆ ที่ต้องการทำซ้ำในแต่ละรอบ
}
```

ทั้งนี้เช่นเดียวกันกับคำสั่ง if หรือ while คือถ้าหากคำสั่งที่ต้องการทำซ้ำมีเพียงคำสั่งเดียวอาจไม่ต้องเขียนเครื่องหมาย { } เพื่อระบุขอบเขตก็ได้

ตัวอย่างเช่นคำสั่ง while ในตัวอย่างส่วนของโปรแกรมที่ 4.1 ข้างต้นอาจเปลี่ยนมาใช้เป็นคำสั่ง for ได้ดังตัวอย่างส่วนของโปรแกรมที่ 4.2

ตัวอย่างส่วนของโปรแกรมที่ 4.2 ส่วนของโปรแกรมที่ใช้ในการแสดงเลขที่หารด้วย 3 ลงตัว โดยใช้ for

```
1 for( i=0; i<=10; i=i+1)
2 {
3    if(i%3 == 0)
4    {
5       Console.WriteLine(i);
6    }
7 }
```

แบบฝึกหัดที่ 4.1 : ทดลองคำสั่ง for: นับขึ้น

เติมโปรแกรมต่อไปนี้ให้เป็นโปรแกรมที่รับจำนวนเต็ม n จากนั้นพิมพ์จำนวนตั้งแต่ 1 ถึง n

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class counting`
`{`
`  public static void Main(string [] args)`
`  {`
`  int n = int.Parse(Console.ReadLine());`
`  for(_____; _____; _____)`
`  Console.WriteLine(_____);`
`  }`
`}`
	7
	1
	2
	3
	4
	5
	6
	7

แบบฝึกหัดที่ 4.2 : ทดลองคำสั่ง for: นับขึ้นจากศูนย์

เติมโปรแกรมต่อไปนี้ให้เป็นโปรแกรมที่รับจำนวนเต็ม n จากนั้นพิมพ์จำนวนตั้งแต่ 0 ถึง n-1

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class counting`
`{`
`  public static void Main(string [] args)`
`  {`
`  int n = int.Parse(Console.ReadLine());`
`  for(_____; _____; _____)`
`  Console.WriteLine(_____);`
`  }`
`}`
	7
	0
	1
	2
	3
	4
	5
	6

แบบฝึกหัดที่ 4.3 : ทดลองคำสั่ง for: นับลง

เติมโปรแกรมต่อไปนี้ให้เป็นโปรแกรมที่รับจำนวนเต็ม n จากนั้นพิมพ์จำนวนตั้งแต่ n ถึง 1

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class counting`
`{`
`  public static void Main(string [] args)`
`  {`
`  int n = int.Parse(Console.ReadLine());`
`  for(_____; _____; _____)`
`  Console.WriteLine(_____);`
`  }`
`}`
	7
	7
	6
	5
	4
	3
	2
	1

แบบฝึกหัดที่ 4.4 : รับและแสดงข้อมูลแบบง่าย

จงเติมโปรแกรมด้านต่อไปนี้ให้สมบูรณ์เพื่อให้โปรแกรมต่อไปนี้ เป็นโปรแกรมที่รับข้อมูลเป็นเลขจำนวนเต็ม 5 จำนวนจากผู้ใช้ จากนั้นแสดงข้อมูลทั้ง 5 จำนวนนี้ทางหน้าจอ

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class Program`
`{`
`  static void Main()`
`  {`
`  int i;`
`  int [] data = _____;`
 	
`  for (i = 0; i < 5; i++)`
`  data[_____] = int.Parse(Console.ReadLine());`
`  Console.WriteLine(_____);`
 	
`  for(_____; _____; _____)`
`  Console.WriteLine("data[{0}] = {1}",_____,_____);`
`  }`
`}`
	30
	20
	15
	0
	9
	Your data:
	data[0] = 30
	data[1] = 20
	data[2] = 15
	data[3] = 0
	data[4] = 9

แบบฝึกหัดที่ 4.5 : พิมพ์ค่าในอาร์เรย์ 1

จงเติมโปรแกรมต่อไปนี้ให้สมบูรณ์เพื่อให้โปรแกรมต่อไปนี้ แสดงผลลัพธ์ดังตัวอย่างการทำงานที่กำหนดให้

***คำแนะนำ** ในส่วน while แรก จะเป็นการกำหนดค่าให้กับอาร์เรย์ x ที่สร้างขึ้นมาก ในส่วน while ที่สองจะเป็นการพิมพ์ค่าของข้อมูลในอาร์เรย์

โปรแกรมที่เขียนได้คือ

```
using System;
class example
{
  public static void Main(string [] args)
  {
    int [] x = new int[10];

    /*  initialize array x   */
    int i = 0;
    int a = 1;
    while(__________)
    {
      ______________
      a += 20;
      i++;
    }

    /*  print array x  */
    i = _____________;
    while(_______________)
    {
      Console.WriteLine(_______________);
      i++;
    }
  }
}
```

ตัวอย่างการทำงาน

```
1
21
41
61
81
101
121
141
161
181
```

แบบฝึกหัดที่ 4.6 : พิมพ์ค่าในอาร์เรย์ 2

จงเติมโปรแกรมต่อไปนี้ให้สมบูรณ์เพื่อให้โปรแกรมต่อไปนี้ แสดงผลลัพธ์ดังตัวอย่างการทำงานที่กำหนดให้

***คำแนะนำ** การทำงานของโปรแกรมนี้คล้ายกับแบบฝึกหัดที่ 4.5 อย่างไรก็ตามให้พิจารณาลำดับของการพิมพ์ข้อมูลในอาร์เรย์จากตัวอย่าง

โปรแกรมที่เขียนได้คือ

```
using System;
class example
{
  public static void Main(string [] args)
  {
    int [] x = new int[10];

    /*  initialize array x   */
    int i = 0;
    int a = 1;
    while(__________)
    {
      ______________
      a += 20;
      i++;
    }

    /*  print array x  */
    i = _____________;
    while(_______________)
    {
      Console.WriteLine(_______________);
      ___________________
    }
  }
}
```

ตัวอย่างการทำงาน

```
181
161
141
121
101
81
61
41
21
1
```

แบบฝึกหัดที่ 4.7 : พิมพ์ค่าในอาร์เรย์ 3

จงเติมโปรแกรมต่อไปนี้ให้สมบูรณ์เพื่อให้โปรแกรมต่อไปนี้ แสดงผลลัพธ์ดังตัวอย่างการทำงานที่กำหนดให้

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class example`
`{`
`  public static void Main(string [] args)`
`  {`
`  int [] x = new int[10];`
 	
`  /*  initialize array x  */`
`  int a = 1;`
`  for(int i = 0; _____; _____)`
`  {`
`  _____`
`  a += 20;`
`  }`
 	
`  /*  print array x  */`
`  for( _____;  _____; _____)`
`  Console.WriteLine(_____);`
`  }`
`}`
	181
	161
	141
	121
	101
	81
	61
	41
	21
	1

แบบฝึกหัดที่ 4.8 : คำนวณค่าเฉลี่ย

ให้เขียนโปรแกรมที่คำนวณค่าเฉลี่ยของคะแนนของนักเรียนทั้งหมด โดยโปรแกรมจะสอบถามจำนวนนักเรียนก่อนในลำดับแรก จากนั้นจึงเริ่มถามคะแนนของนักเรียนแต่ละคน เพื่อนำมาคำนวณหาค่าเฉลี่ยของคะแนนของนักเรียนทั้งหมด ดังตัวอย่าง

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class ArrayAVG`
`{`
`  static void Main()`
`  {`
`  int n;`
`  Console.Write("Input numbers of students : ");`
`  n = int.Parse(Console.ReadLine());`
`  _____`
`  _____`
`  _____`
`  _____`
`  }`
`}`
	Input numbers of students : **4**
	Score of student1 = **80.2**
	Score of student2 = **67.5**
	Score of student3 = **70**
	Score of student4 = **68.5**
	Average score = 71.55

แบบฝึกหัดที่ 4.9 : เมธอดอ่านอาร์เรย์

จงเขียนเติมข้อความที่เหมาะสมลงในช่องว่าง เพื่อให้เมธอด ReadArray เป็นเมธอดสำหรับอ่านอาร์เรย์จากผู้ใช้ โดยเมธอดดังกล่าวจะรับจำนวนเต็ม n แทนจำนวนข้อมูลในอาร์เรย์ จากนั้นจะอ่านจำนวนเต็มอีก n จำนวนมาเก็บในอาร์เรย์ และคืนอาร์เรย์นั้นออกมา

*หมายเหตุโปรแกรมนี้ได้เขียนเมธอด Main สำหรับทดสอบเมธอด ReadArray เรียบร้อยแล้ว

โปรแกรมที่เขียนได้คือ

```
using System;
class example
{
  static int [] ReadArray()
  {
    int n = int.Parse(____________________);

    int [] a =  ________________; /*Create new array of length n*/
    for(int i = 0; __________; i++)
      a[____] = ________________;
    return a;
  }

  public static void Main(string [] args)
  {
    int [] arr = ReadArray();
    Console.WriteLine("Array has {0} elements:", arr.Length);
    foreach(int x in arr)
      Console.WriteLine(x);
  }
}
```

ตัวอย่างการทำงาน

ตัวอย่างที่ 1

```
5
10
20
15
25
50
Array has 5 elements:
10
20
15
25
50
```

ตัวอย่างที่ 2

```
2
100
75
Array has 2 elements:
100
75
```

แบบฝึกหัดที่ 4.10 : เมธอดคำนวณค่าเฉลี่ย - for

จงเขียนเมธอด FindAverage ที่รับพารามิเตอร์เป็นอาร์เรย์ของจำนวนเต็ม และคืนค่าเป็นค่าเฉลี่ยของข้อมูลในอาร์เรย์ให้สมบูรณ์

โปรแกรมด้านล่างมีเมธอด Main เตรียมให้สำหรับทดสอบเมธอด FindAverage

คำแนะนำ: สามารถเรียกใช้คุณสมบัติ Length ของอารเรย์เพื่ออ่านจำนวนข้อมูลในอาร์เรย์ได้ เช่น ถ้าเราเรียก a.Length จะได้จำนวนข้อมูลในอาร์เรย์ a

โปรแกรมที่เขียนได้คือ

```
using System;
class example
{
  // คัดลอกเมธอด ReadArray จากแบบฝึกหัดที่ 4.9 มาวางไว้บริเวณนี้

  static double FindAverage(int [] a)
  {
    double total = 0;
    for(int i = 0; _________________; i++)
      ____________________________;
    return _________________;
  }

  public static void Main(string [] args)
  {
    int [] arr = ReadArray();
    double avr = FindAverage(arr);
    Console.WriteLine("Average is {0:f3}",avr);
  }
}
```

ตัวอย่างการทำงาน

ตัวอย่างที่ 1

```
5
10
20
15
25
50
Average is 24.000
```

ตัวอย่างที่ 2

```
2
100
75
Average is 87.500
```

5. คำสั่งวนซ้ำแบบ foreach

การเขียนโปรแกรมมีหลายครั้งที่เราจำเป็นต้องเข้าไปดูข้อมูลที่เป็นสมาชิกภายในอาร์เรย์ที่ละสมาชิกเพื่อใช้ประกอบการคำนวณ นอกจากคำสั่ง while และ for ที่สามารถใช้ประยุกต์ในการดูข้อมูลทุกตัวในอาร์เรย์แล้ว คำสั่ง foreach เป็นคำสั่งที่ออกแบบมาเพื่อทำหน้าที่นี้โดยเฉพาะ โดยมีรูปแบบดังนี้

```
foreach ( ประกาศตัวแปรให้เป็นชนิดเดียวกันกับข้อมูลในอาร์เรย์ที่ต้องการ  in  ชื่ออาร์เรย์ )
{
   คำสั่งที่ต้องการให้โปรแกรมทำงานในแต่ละรอบ
}
```

การทำงานของ foreach ... in ในภาษา C# จะคล้ายกับคำสั่ง for ... in ในภาษาไพทอน กล่าวคือตัวแปรที่เราประกาศขึ้นมาก่อนคำสำคัญ in จะไปคัดลอกข้อมูลของสมาชิกในอาร์เรย์ที่เราระบุไว้รอบละหนึ่งสมาชิก แล้วเราสามารถนำค่าในตัวแปรนั้นไปใช้ในการคำนวณอื่น ๆ ต่อไปได้ เช่นส่วนของโปรแกรมตัวอย่างที่ 5.1 ในรอบแรก รอบที่สอง และรอบที่สาม x จะมีค่าเท่ากับ 4, 9 และ 8 ตามลำดับ ทำให้คำสั่ง Console.WriteLine(x*10); ในบรรทัดที่ 4 จึงแสดงเลข 40, 90 และ 80 ตามลำดับ

ตัวอย่างส่วนของโปรแกรมที่ 5.1

```
1  int [ ]  numbers = {4, 9, 8}
2  foreach ( int x  in  numbers)
3  {
4      Console.WriteLine(x*10);
5  }
```

ตัวอย่างส่วนของโปรแกรมที่ 5.2.1 และ 5.2.2 เป็นส่วนของโปรแกรมที่ใช้คำนวณหาผลรวมของสมาชิกทุกตัวในอาร์เรย์ scores เก็บใส่ตัวแปร total โดยตัวอย่างส่วนของโปรแกรมที่ 5.2.1 ใช้โครงสร้าง for แต่ส่วนของโปรแกรมที่ 5.2.2 ใช้โครงสร้าง foreach

	ตัวอย่างส่วนของโปรแกรมที่ 5.2.1	ตัวอย่างส่วนของโปรแกรมที่ 5.2.2
1	`int total = 0;`	`int total = 0;`
2	`for(int i = 0; i < scores.Length; i++)`	`foreach(int  x  in  scores)`
3	`total += scores[i];`	`  total += x;`

แบบฝึกหัดที่ 5.1 : เมธอดคำนวณค่าเฉลี่ย - foreach

จงเขียนเมธอด FindAverage ที่รับพารามิเตอร์เป็นอาร์เรย์ของจำนวนเต็ม และคืนค่าเป็นค่าเฉลี่ยของข้อมูลในอาร์เรย์ให้สมบูรณ์ สำหรับข้อนี้แตกต่างจากแบบฝึกหัดที่ 4.10 ที่บังคับให้ใช้คำสั่ง foreach

***คำแนะนำ** สามารถเรียกใช้คุณสมบัติ Length ของอาร์เรย์เพื่ออ่านจำนวนข้อมูลในอาร์เรย์ได้ เช่น ถ้าเราเรียก a.Length จะได้จำนวนข้อมูลในอาร์เรย์ a

โปรแกรมที่เขียนได้คือ

```
using System;
class example
{
  // คัดลอกเมธอด ReadArray จากแบบฝึกหัดที่ 4.9 มาวางไว้บริเวณนี้

  static double FindAverage(int [] a)
  {
    double total = 0;
    foreach(_______________________________)
      ____________________________;
    return _______________;
  }

  public static void Main(string [] args)
  {
    int [] arr = ReadArray();
    double avr = FindAverage(arr);
    Console.WriteLine("Average is {0:f3}",avr);
  }
}
```

ตัวอย่างการทำงาน

ตัวอย่างที่ 1

```
5
10
20
15
25
50
Average is 24.000
```

ตัวอย่างที่ 2

```
2
100
75
Average is 87.500
```

แบบฝึกหัดที่ 5.2 : เมธอดนับเลขคี่

จงเขียนเมธอด CountOdd ที่รับพารามิเตอร์เป็นอาร์เรย์ของจำนวนเต็ม และคืนค่าเป็นจำนวนของข้อมูลในอาร์เรย์ที่เป็นเลขคี่

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class example`
`{`
`  // คัดลอกเมธอด ReadArray จากแบบฝึกหัดที่ 4.9 มาวางไว้บริเวณนี้`
`  _____`
`  _____`
`  _____`
`  _____`
`  _____`
`  _____`
`  public static void Main(string [] args)`
`  {`
`  int [] arr = ReadArray();`
`  Console.WriteLine("There are {0} odd numbers",CountOdd(arr));`
`  }`
`}`
	ตัวอย่างที่ 1
	5
	10
	20
	15
	25
	50
	`There are 2 odd numbers`
	_____
	ตัวอย่างที่ 2
	2
	100
	75
	`There are 1 odd numbers`

แบบฝึกหัดที่ 5.3 : เมธอดค้นหาในอาร์เรย์

จงเขียนเมธอด Find ที่รับพารามิเตอร์ดังนี้

- อาร์เรย์ a ของจำนวนเต็ม ไม่มีจำนวนเต็มที่ซ้ำกันในอาร์เรย์
- จำนวนเต็ม g

จากนั้นให้ค้นหาว่าค่า g อยู่ในอาร์เรย์หรือไม่ ให้เมธอดคืนค่าดัชนีในอาร์เรย์ที่มีค่า g เก็บอยู่ ถ้าไม่พบให้คืนค่า -1

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class example`
`{`
`  _____`
`  _____`
`  _____`
`  _____`
`  _____`
`  _____`
`  public static void Main(string [] args)`
`  {`
`  int [] array1 = {1,2,3,4,5,6,7,8,9,10};`
`  Console.WriteLine("Array 1");`
`  Console.WriteLine(Find(array1,1));`
`  Console.WriteLine(Find(array1,7));`
`  Console.WriteLine(Find(array1,10));`
`  Console.WriteLine(Find(array1,100));`
`  Console.WriteLine(Find(array1,-1));`
`  int [] array2 = {7,6,4,10,24,1,1200};`
`  Console.WriteLine("Array 2");`
`  Console.WriteLine(Find(array2,10));`
`  Console.WriteLine(Find(array2,100));`
`  Console.WriteLine(Find(array2,1000));`
`  Console.WriteLine(Find(array2,1200));`
`  }`
`}`
	Array 1
	0
	6
	9
	-1
	-1
	Array 2
	3
	-1
	-1
	6

6. string กับ อาร์เรย์

ตัวแปรชนิดข้อความ (string) นั้นจริง ๆ แล้วเป็นอาร์เรย์ชนิดหนึ่งแต่เป็นอาร์เรย์ของตัวอักษร (char) อีกทีนั้นเอง

ภาษา C# นั้นเราสามารถประมวลผลข้อมูลชนิดข้อความ (string) เสมือนว่าข้อมูลนั้นเป็นอาร์เรย์ของตัวอักษร (char) จึงสามารถอ้างอิงถึงตัวอักษรตำแหน่งใด ๆ ได้โดยใช้ดัชนีเช่นเดียวกับอาร์เรย์ทั่วไป ทั้งยังสามารถใช้คุณสมบัติ .Length อีกด้วย แต่ข้อจำกัดของการใช้ตัวแปรชนิดข้อความ (string) เป็นอาร์เรย์ของตัวอักษรคือ ตัวอักษรแต่ละตัวนั้นจะไม่สามารถแก้ไขข้อมูลได้

ตัวอย่างส่วนของโปรแกรมที่ 6.1	ผลลัพธ์ที่ได้จากตัวอย่างส่วนของโปรแกรมที่ 6.1

`string s = "Hello";`
`Console.WriteLine(s.Length);`
`Console.WriteLine(s[0]);`
`Console.WriteLine(s[1]);`
`Console.WriteLine(s[3]);`
	5
	H
	e
	l

จากตัวอย่างส่วนของโปรแกรมที่ 6.1 ถ้าเราสั่ง s[1] = 'G'; โปรแกรมจะคอมไพล์ไม่ผ่าน เพราะว่าการอ้างสตริงแบบอาร์เรย์ทำได้เฉพาะในการอ่านเท่านั้น

แบบฝึกหัดที่ 6.1 : สตริงกับอาร์เรย์

จงเขียนโปรแกรมที่รับสตริง 1 ตัว จากนั้นให้พิมพ์สตริงดังกล่าวทีละตัวอักษร อักษรละหนึ่งบรรทัด

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class example`
`{`
`  public static void Main(string[] args)`
`  {`
`  string s = Console.ReadLine();`
`  _____`
`  _____`
`  _____`
`  _____`
`  _____`
`  }`
`}`
	Cool!
	C
	o
	o
	l
	!

7. ข้อมูลประเภท char

ตัวอักษรทุกตัวจะมีเลขลำดับ (คล้ายกับที่นิสิตทุกคนต่างก็มีลำดับของรหัสนิสิตเป็นของตัวเอง) หากนักเรียนอยากทราบว่าตัวอักษรหนึ่ง ๆ มีเลขลำดับเป็นเท่าไหร่สามารถตรวจสอบได้ดังแบบฝึกหัดที่ 7.1

แบบฝึกหัดที่ 7.1 : เลขลำดับของตัวอักษร

ให้นักเรียนบันทึกผลลัพธ์ที่ได้เมื่อใช้คำสั่งต่อไปนี้

คำสั่ง	ผลลัพธ์
Console.WriteLine((int) 'A');	
Console.WriteLine((int) 'B');	
Console.WriteLine((int) 'C');	
Console.WriteLine((int) 'a');	
Console.WriteLine((int) 'b');	
Console.WriteLine((int) 'c');	
Console.WriteLine((int) '0');	
Console.WriteLine((int) '1');	

คำสั่ง	ผลลัพธ์
char c = 'D'; Console.WriteLine(c - 'A');	
char c = 'd'; Console.WriteLine(c - 'a');	
char c = '5'; Console.WriteLine(c - '0');	

ตัวอย่างที่ 7.1 : โปรแกรมพิมพ์ลำดับอักขระ

โปรแกรมต่อไปนี้ รับตัวอักษรภาษาอังกฤษพิมพ์ใหญ่ แล้วระบุว่าเป็นตัวอักษรตัวที่เท่าใด

โปรแกรม	ตัวอย่างการทำงาน

`using System;`
 	
`class teststring`
`{`
`  public static void Main(string [] args)`
`  {`
`  string s = Console.ReadLine();`
`  char c = s[0];`
`  int o = c - 'A' + 1;`
`  Console.WriteLine("{0} is the {1}-th character.", c, o);`
`  }`
`}`
	ตัวอย่างที่ 1
	A
	A is the 1-th character.
	
	ตัวอย่างที่ 2
	B
	B is the 2-th character.
	
	ตัวอย่างที่ 3
	H
	H is the 8-th character.
	
	ตัวอย่างที่ 4
	Z
	Z is the 26-th character.

แบบฝึกหัดที่ 7.2 : อันดับของตัวอักษร

จงแก้โปรแกรมจากตัวอย่างที่ 7.1 ข้างต้น ให้สามารถทำงานได้กับทั้งกรณีที่ข้อมูลป้อนเข้าเป็นตัวอักษรพิมพ์ใหญ่และตัวอักษรพิมพ์เล็ก

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน
	ตัวอย่างที่ 1 **A** A is the 1-th character. **ตัวอย่างที่ 2** **a** a is the 1-th character. **ตัวอย่างที่ 3** **c** c is the 3-th character. **ตัวอย่างที่ 4** **h** h is the 8-th character.

แบบฝึกหัดที่ 7.3 : เมธอดนับอักษรในสตริง

จงเขียนเมธอด CountChar ที่รับพารามิเตอร์ดังนี้

- สตริง st
- อักขระ c

จากนั้นคืนค่าเป็นจำนวนครั้งที่อักขระ c ปรากฏใน st

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`using System;`
`class countchar`
`{`
`  static int CountChar(string st, char c)`
`  {`
`  _____`
`  _____`
`  _____`
`  _____`
`  }`
 	
`  public static void Main(string [] args)`
`  {`
`  string input = Console.ReadLine();`
`  string query = Console.ReadLine();`
`  char q = query[0];`
`  Console.WriteLine(CountChar(input,q));`
`  }`
`}`
	ตัวอย่างที่ 1
	Hello world
	l
	3
	_____
	
	ตัวอย่างที่ 2
	Welcome to the new world
	w
	2

แบบฝึกหัดที่ 7.4 : เมธอดนับอักษรในสตริง

จงเขียนโปรแกรมรับสตริง s จากนั้นพิมพ์สตริง s ย้อนกลับ

*คำแนะนำ: ถ้าต้องการพิมพ์โดยไม่ขึ้นบรรทัดใหม่ สามารถใช้ Console.Write ได้

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน
	ตัวอย่างที่ 1 **Hello** olleH _____ **ตัวอย่างที่ 2** **This is the beginning** gninnigeb eht si sihT