

คู่มืออบรมการเขียนโปรแกรมด้วยภาษา C#



คำนำ

สถาบันส่งเสริมการสอนวิทยาศาสตร์และเทคโนโลยี (สสวท.) ได้ตระหนักถึงความสำคัญในการพัฒนาครูมาโดยตลอด โดยมีการจัดประชุมปฏิบัติการอบรมครูและจัดกิจกรรมต่างๆ ซึ่งช่วยให้ครูนำความรู้และประสบการณ์ที่ได้รับไปปรับปรุงและพัฒนาการจัดการเรียนการสอนให้เข้ากับสภาพของท้องถิ่นตามความเหมาะสม

ปีงบประมาณ ๒๕๕๒ ได้ดำเนินโครงการจัดการแข่งขันคอมพิวเตอร์โอลิมปิกระหว่างประเทศ ครั้งที่ ๒๓ ปี พ.ศ. ๒๕๕๔ โดยได้ดำเนินกิจกรรมต่างๆ เพื่อรองรับการเป็นเจ้าภาพจัดการแข่งขันฯ โดยเฉพาะอย่างยิ่งการพัฒนาศักยภาพของครูและนักเรียนทางด้านคอมพิวเตอร์ และการเขียนโปรแกรมสาขาคอมพิวเตอร์ จึงได้ร่วมกับคณาจารย์จากมหาวิทยาลัยต่างๆ จัดให้มีการอบรมครูหลักสูตรการโปรแกรมภาษาซีชาร์ปขึ้น เพื่อพัฒนาคุณภาพการเรียนการสอนด้านการเขียนโปรแกรมในโรงเรียน โดยได้จัดทำเอกสารสำหรับการอบรมครูหลักสูตรการเขียนโปรแกรมภาษาซีชาร์ปให้กับครูในโรงเรียนที่สนใจได้เพิ่มพูนทักษะ และนำความรู้ไปประยุกต์ใช้ในการจัดการเรียนการสอนวิชาการโปรแกรมต่อไป

การจัดทำเอกสารได้รับความร่วมมืออย่างดียิ่งจากผู้ทรงคุณวุฒิและนักวิชาการจากหน่วยงานต่างๆ ของรัฐ วิทยากรแกนนำคอมพิวเตอร์ และผู้สอนวิชาเทคโนโลยีสารสนเทศ ระดับมัธยมศึกษา รวมทั้งนักวิชาการสาขาคอมพิวเตอร์ สสวท. จึงขอขอบคุณไว้ ณ ที่นี้

สสวท. หวังเป็นอย่างยิ่งว่าเอกสารฉบับนี้จะเป็นประโยชน์แก่สถานศึกษาและผู้เกี่ยวข้องทุกฝ่ายที่จะช่วยพัฒนาการจัดการเรียนการสอนด้านการโปรแกรมให้มีประสิทธิภาพ หากมีข้อเสนอแนะใดที่จะทำให้เอกสารนี้สมบูรณ์ยิ่งขึ้น โปรดแจ้ง สาขาคอมพิวเตอร์ สสวท. ทราบด้วยจักขอบพระคุณยิ่ง

สาขาคอมพิวเตอร์

สถาบันส่งเสริมการสอนวิทยาศาสตร์และเทคโนโลยี

๒๐ เมษายน ๒๕๕๒

สารบัญ

คำนำ.....	2
สารบัญ.....	3
กิจกรรมที่ 1 แนะนำภาษา C#.....	6
ใบงานที่ 1.1 รู้จักกับภาษา C#	8
ใบงานที่ 1.2 นิพจน์ทางคณิตศาสตร์และคำสั่งในการแสดงผล.....	12
ใบความรู้ที่ 1.1 การสร้างแอปพลิเคชันด้วย MS Visual C#	14
ใบความรู้ที่ 1.2 ภาษา C# และเครื่องมือพัฒนาโปรแกรม.....	16
ใบความรู้ที่ 1.3 โครงสร้างโปรแกรมภาษา C# และข้อมูลพื้นฐานและตัวดำเนินการ	18
ใบความรู้ที่ 1.4 การขอความช่วยเหลือ.....	29
กิจกรรมที่ 2 ฟังก์ชันทางคณิตศาสตร์และการติดต่อกับผู้ใช้	32
ใบงานที่ 2.1 ฟังก์ชันทางคณิตศาสตร์	34
ใบงานที่ 2.2 แก๊สมการกำลังสอง.....	36
ใบงานที่ 2.3 คุณสมบัติของวงกลม	37
ใบความรู้ที่ 2.1 ฟังก์ชันทางคณิตศาสตร์.....	38
ใบความรู้ที่ 2.2 คำสั่งสำหรับรับข้อมูลจากผู้ใช้.....	39
กิจกรรมที่ 3 คำสั่งแบบมีเงื่อนไข	41
ใบงานที่ 3.1 การตรวจสอบค่าความจริง.....	44
ใบงานที่ 3.2 ระบบจตุภาค.....	45
ใบงานที่ 3.3 ดัชนีมวลกาย	47
ใบงานที่ 3.4 สัมผัสสี่เหลี่ยม	49
ใบงานที่ 3.5 เครื่องคิดเลขรุ่นใหม่.....	51
ใบความรู้ที่ 3.1 นิพจน์ทางตรรกศาสตร์.....	52
ใบความรู้ที่ 3.2 โครงสร้าง if และ if...else	53
ใบความรู้ที่ 3.3 โครงสร้าง if หลายชั้น	57
ใบความรู้ที่ 3.4 โครงสร้าง switch...case.....	59
กิจกรรมที่ 4 คำสั่งวนซ้ำ.....	63
ใบงานที่ 4.1 ผลรวมตัวเลข.....	66
ใบงานที่ 4.2 สถิติคะแนนสอบ	67
ใบงานที่ 4.3 ลำดับตัวเลข.....	70
ใบงานที่ 4.4 จำนวนเฉพาะ.....	71



ใบงานที่ 4.5 กลุ่มดาวสามเหลี่ยม	73
ใบความรู้ที่ 4.1 โครงสร้าง while ลูป	74
ใบความรู้ที่ 4.2 โครงสร้าง do...while ลูป	77
ใบความรู้ที่ 4.3 โครงสร้าง for ลูป	78
กิจกรรมที่ 5 เมทีอดเบื้องต้น.....	81
ใบงานที่ 5.1 แก๊ไขเมทีอดพิมพ์ดาว.....	84
ใบงานที่ 5.2 ทดลองส่งค่าไปยังเมทีอด	85
ใบงานที่ 5.3 ทดลองวาดแผนภาพ.....	86
ใบงานที่ 5.4 คำนวนพื้นที่วงกลม	87
ใบงานที่ 5.5 อนุกรมฮาโมนิค	88
ใบงานที่ 5.6 วาดกราฟของฟังก์ชันอย่างง่าย	89
ใบความรู้ที่ 5.1 การประกาศและเรียกใช้เมทีอด.....	91
ใบความรู้ที่ 5.2 การส่งค่าไปยังเมทีอด	95
ใบความรู้ที่ 5.3 เมทีอดแบบคืนค่า.....	96
กิจกรรมที่ 6 อาร์เรย์.....	99
ใบงานที่ 6.1 สร้างอาร์เรย์ของคะแนน	101
ใบงานที่ 6.2 หาค่าเฉลี่ยของคะแนน.....	102
ใบงานที่ 6.3 รับส่งอาร์เรย์ไปยังเมทีอด.....	103
ใบงานที่ 6.4 วิเคราะห์อักขระ.....	105
ใบงานที่ 6.5 ถอดรหัสลับจรรยาชน	106
ใบงานที่ 6.6 แผนภูมิแท่ง	108
ใบความรู้ที่ 6.1 ชนิดข้อมูลแบบอาร์เรย์.....	110
ใบความรู้ที่ 6.2 การหาขนาดของอาร์เรย์.....	114
ใบความรู้ที่ 6.3 คำสั่ง foreach	115
ใบความรู้ที่ 6.4 การส่งอาร์เรย์ไปยังเมทีอด	116
ใบความรู้ที่ 6.5 การการอ้างอิงถึงสตริงในรูปแบบอาร์เรย์.....	117
กิจกรรมที่ 7 อาร์เรย์หลายมิติ.....	118
ใบงานที่ 7.1 แสดงค่าในเมตริกซ์.....	120
ใบงานที่ 7.2 กำหนดค่าให้เมตริกซ์	121
ใบงานที่ 7.3 เมตริกซ์ทรานสโพส	123
ใบงานที่ 7.4 ดีเทอร์มิแนนต์ของเมตริกซ์.....	125
ใบงานที่ 7.5 การคูณเมตริกซ์	126



ใบความรู้ที่ 7.1 อาร์เรย์สองมิติ.....	128
ใบความรู้ที่ 7.2 การหาขนาดของอาร์เรย์.....	131
แบบฝึกหัดเพิ่มเติมเรื่องอาร์เรย์สองมิติ	132
กิจกรรมที่ 8 การพัฒนาโปรแกรมติดต่อกับผู้ใช้แบบกราฟิก	133
ใบงานที่ 8.1 ออกแบบโปรแกรมเครื่องคิดเลข	135
ใบงานที่ 8.2 โปรแกรมเครื่องคิดเลขอย่างง่าย	136
ใบความรู้ที่ 8.1 การโปรแกรมเชิงวัตถุ	138
ใบความรู้ที่ 8.2 การสร้างโปรเจกต์แบบ Windows.....	145
กิจกรรมที่ 9 พัฒนาโครงงาน	159
ใบงานที่ 9.1 พัฒนาโครงงานแบบ Windows Application.....	161
ภาคผนวก.....	163
การใช้งานโปรแกรม Microsoft Visual C# 2008 Express.....	164
กิจกรรมที่ 10 การจัดกลุ่มข้อมูลด้วยโครงสร้าง.....	177
ใบงานที่ 10.1 รู้จักกับข้อมูลแบบโครงสร้าง	179
ใบงานที่ 10.2 เวกเตอร์.....	181
ใบงานที่ 10.3 ฐานข้อมูลนักเรียน.....	184
ใบความรู้ที่ 10.1 ชนิดข้อมูลแบบโครงสร้าง	186
ใบความรู้ที่ 10.2 การใช้งานโครงสร้างร่วมกับอาร์เรย์	189
คณะผู้พัฒนาเอกสาร การโปรแกรมภาษาซีชาร์ป.....	191



กิจกรรมที่ 1

แนะนำภาษา C#

1. จุดประสงค์ ให้ผู้เรียนสามารถ

- 1.1 สร้างโปรเจกต์แบบ Console Application ด้วย MS Visual C#
- 1.2 ตั้งชื่อตัวระบุได้ตามกฎเกณฑ์การตั้งชื่อ
- 1.3 เขียนคำสั่งในภาษา C# เพื่อแสดงผลข้อมูล

2. แนวคิด

ในสภาพแวดล้อมของ MS Visual C# การพัฒนาแอปพลิเคชันขึ้นมาหนึ่งชิ้นเรียกว่าการสร้างโซลูชัน (solution) ซึ่งประกอบไปด้วยโปรเจกต์ (project) ตั้งแต่หนึ่งหรือมากกว่า โดยแต่ละโปรเจกต์เป็นส่วนหนึ่งของซอฟต์แวร์ที่อาจเป็นส่วนหนึ่งของโปรแกรมหลัก หรือส่วนไลบรารี (library) ที่ถูกเรียกใช้โดยโปรเจกต์อื่น ๆ

การใช้งานโปรแกรมภาษา C# มักจะมีการใช้งานตัวระบุ (identifier) อยู่ทั่วไปภายในโปรแกรม เช่น ชื่อของเนมสเปซ คลาส ตัวแปร ค่าคงที่ โดยการตั้งชื่อตัวระบุจะต้องตั้งชื่อตามกฎเกณฑ์ที่กำหนดไว้

อย่างไรก็ตาม การตรวจสอบว่าโปรแกรมที่เขียนขึ้นทำงานได้ถูกต้องตามที่ต้องการจำเป็นจะต้องมีการแสดงผลลัพธ์ให้เห็นในรูปแบบใดรูปแบบหนึ่ง ในที่นี้เราอาศัยคำสั่งที่ใช้ในการพิมพ์ข้อความและค่าของนิพจน์ต่าง ๆ ออกทางจอภาพ ได้แก่ คำสั่ง *Write* และ *WriteLine* ซึ่งถูกนิยามไว้ในคลาสชื่อ *Console* และเนมสเปซชื่อ *System* สองคำสั่งนี้มีการใช้งานในลักษณะเดียวกัน แตกต่างกันที่คำสั่ง *WriteLine* จะพิมพ์ข้อความตามด้วยการขึ้นบรรทัดใหม่

3. สื่ออุปกรณ์

3.1 ใบงาน

ใบงานที่	เรื่อง	เวลา (นาที)
1.1	รู้จักกับภาษา C#	25
1.2	นิพจน์ทางคณิตศาสตร์และคำสั่งในการแสดงผล	30

3.2 ใบความรู้

- ใบความรู้ที่ 1.1 เรื่องสร้างแอปพลิเคชันด้วย MS Visual C#



- ใบความรู้ที่ 1.2 เรื่องภาษา C# และเครื่องมือพัฒนาโปรแกรม
- ใบความรู้ที่ 1.3 เรื่องโครงสร้างของโปรแกรมภาษา C# และข้อมูลพื้นฐานและตัวดำเนินการ
- ใบความรู้ที่ 1.4 เรื่องการขอความช่วยเหลือ

3.3 อื่นๆ

- เครื่องคอมพิวเตอร์เท่ากับจำนวนกลุ่ม
- โปรแกรมวิชวลซีชาร์ป เอ็กเพรส (MS Visual C# Express)

4. วิธีดำเนินการ

4.1 การจัดเตรียม

- 4.1.1 แบ่งผู้เรียนเป็นกลุ่ม กลุ่มละ 2 คน
- 4.1.2 เตรียมใบงานที่ 1.1 - 1.2 ตามจำนวนกลุ่มและใบความรู้ที่ 1.1 - 1.3 ตามจำนวนผู้เรียน

4.2 ขั้นตอนการดำเนินการ

- 4.2.1 ผู้สอนกล่าวถึงภาษา C# และโปรแกรม MS Visual C#
- 4.2.2 ผู้เรียนแต่ละคนศึกษาใบความรู้ที่ 1.1 เรื่องสร้างแอปพลิเคชันด้วย MS Visual C# ใบความรู้ที่ 1.2 เรื่องภาษา C# และเครื่องมือพัฒนาโปรแกรม ใบความรู้ที่ 1.3 เรื่องโครงสร้างของโปรแกรมภาษา C# และข้อมูลพื้นฐานและตัวดำเนินการ และใบความรู้ที่ 1.4 เรื่องการขอความช่วยเหลือ
- 4.2.3 ผู้เรียนทำใบงานที่ 1.1 เรื่องรู้จักกับภาษา C# จากนั้นผู้สอนกลุ่มผู้เรียนออกมานำเสนอ
- 4.2.4 ผู้เรียนทำใบงานที่ 1.2 เรื่องนิพจน์ทางคณิตศาสตร์และคำสั่งในการแสดงผล จากนั้นผู้สอนกลุ่มผู้เรียนออกมานำเสนอ
- 4.2.5 ผู้เรียนและผู้สอนร่วมกันอภิปรายและสรุปเกี่ยวกับ การใช้งานโปรแกรม MS Visual C# เบื้องต้น โครงสร้างของภาษา C# ข้อมูลพื้นฐานและตัวดำเนินการ

5. การวัดและประเมินผล

5.1 ตรวจคำตอบจากใบงาน

6. แหล่งความรู้เพิ่มเติม

- 6.1 [http://msdn.microsoft.com/th-th/vcsharp/default\(en-us\).aspx](http://msdn.microsoft.com/th-th/vcsharp/default(en-us).aspx)

7. ข้อเสนอแนะ

- 7.1 ผู้เรียนสามารถศึกษาวิธีการใช้งานโปรแกรม MS Visual C# Express ได้จากภาคผนวก



ใบงานที่ 1.1

รู้จักกับภาษา C#

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 1.1 ใบความรู้ที่ 1.2 และใบความรู้ที่ 1.3 และปฏิบัติตามขั้นตอนในใบความรู้ แล้วตอบคำถามต่อไปนี้

1. เปิดโปรแกรม MS Visual C# และสร้างโปรเจกต์ เขียนโปรแกรม ดังนี้

1.4 สร้างโปรเจกต์แบบ Console Application และตั้งชื่อว่า *First*

1.5 ลบโค้ดที่ MS Visual C# เตรียมไว้ให้ออกทั้งหมด และพิมพ์โค้ดต่อไปนี้ลงไป

```
namespace First
{
    class First
    {
        static void Main()
        {
            System.Console.WriteLine("Hello teacher!");
            System.Console.ReadLine();
        }
    }
}
```

1.6 ทดลองรันโปรแกรม เขียนผลลัพธ์ที่ได้ลงในช่องว่างด้านล่าง

1.7 นำบรรทัดที่มีคำสั่ง `System.Console.ReadLine()` ; ออกจากโปรแกรม ทดลองรันโปรแกรมอีกครั้งหนึ่ง สังเกตผลลัพธ์ที่ได้ และผู้เรียนคิดว่าคำสั่ง `ReadLine` ทำหน้าที่อะไร

- 1.8 คัดแปลงโปรแกรมข้างต้นใหม่เพื่อให้โปรแกรมทำงานเหมือนเดิมทุกประการแต่มีขนาดสั้นลง โดยตัดบรรทัดที่ระบุ namespace ออกและใช้คำสั่ง **using** ช่วย ทดสอบความถูกต้องและป้อนโค้ดใหม่ที่ได้ลงในช่องว่าง

2. เขียนโปรแกรมที่มีองค์ประกอบตามที่ระบุ

- 2.1 สร้างโปรเจกต์ใหม่ชื่อ *Second* และสร้างโปรแกรมภาษา C# ให้มีองค์ประกอบดังนี้
- โปรแกรมอยู่ใน namespace ชื่อ *MyNameSpace*
 - โปรแกรมหลักอยู่ในคลาสชื่อ *MyClass*
 - เมื่อรันโปรแกรมจะพิมพ์ข้อความต่อไปนี้ และหยุดรอให้ผู้ใช้เคาะแป้น Enter ก่อนปิดหน้าจอ Console

```
Live as if you were to die tomorrow.  
Learn as if you were to live forever.  
-Mahatma Gandhi
```

- 2.2 ทดสอบความถูกต้อง และกรอกโปรแกรมที่ได้ลงในช่องว่าง



3. การใช้งานตัวแปรและค่าคงที่

3.1 พิจารณาชื่อตัวระบุ (ที่จะนำมาใช้ประกาศเป็นชื่อตัวแปร ค่าคงที่ คลาส หรือเนมสเปส) ต่อไปนี้ว่า ถูกต้องตามกฎเกณฑ์การตั้งชื่อหรือไม่ พร้อมทั้งอธิบายเหตุผลหากชื่อนั้นถูกตั้งไม่ถูกต้อง

ชื่อตัวระบุ	ใช้ได้หรือไม่	เหตุผล
xxx	ได้	-
\$\$\$	ไม่ได้	ประกอบด้วยอักขระพิเศษ
_Y		
string		
i_j		
Student ID		
HelloWorld!		
first-time		
null		
123Class		
Section3		
w*h		
do		

3.2 เขียนคำสั่งประกาศตัวแปรชื่อ x ที่มีชนิดข้อมูลเป็น **float** โดยไม่มีการกำหนดค่าเริ่มต้น

3.3 เขียนคำสั่งประกาศตัวแปรชื่อ `myName` ที่มีชนิดข้อมูลเป็น **string** โดยไม่มีการกำหนดค่าเริ่มต้น

3.4 เขียนคำสั่งเพื่อประกาศค่าคงที่ชื่อ PI ที่มีชนิดข้อมูลเป็น **double** เพื่อใช้แทนค่า 3.1415926535

3.5 เขียนคำสั่งภาษา C# เพื่อประกาศตัวแปรหรือค่าคงที่ตามที่กำหนดให้ โดยเลือกใช้ชนิดของข้อมูลที่เหมาะสม

- ค่าคงที่ชื่อ `MY_AGE` เพื่อใช้แทนอายุปัจจุบันของท่านเอง

- ตัวแปรชื่อ *income* เพื่อใช้เก็บเงินเดือนของพอลล่า

- ตัวแปรชื่อ *temp_c* เพื่อเก็บค่าอุณหภูมิบริเวณขั้วโลกเหนือเป็นองศาเซลเซียส

- ตัวแปรชื่อ *temp_k* เพื่อเก็บค่าอุณหภูมิใจกลางดวงอาทิตย์เป็นเคลวิน

- ค่าคงที่ชื่อ *NAME* เพื่อใช้แทนชื่อเต็มของตนเอง

3.6 โปรแกรมด้านล่างถูกเขียนขึ้นมาเพื่อคำนวณหาพื้นที่ของวงกลมที่มีรัศมี 12.5 หน่วย

```
using System;
class CircleArea
{
    static void Main()
    {
        const double PI = 3.1415926535;

        radius = 12.5;
        area = PI * radius * radius;
        Console.WriteLine("Circle area = {0}", area);
    }
}
```

สร้างโปรเจกต์ชื่อ Circle จากนั้นคัดลอกโปรแกรมข้างต้นลงไปและทดลองคอมไพล์โปรแกรม การคอมไพล์พบข้อผิดพลาดใดบ้าง

3.7 แก้ไขโปรแกรมให้คอมไพล์ได้โดยไม่มีข้อผิดพลาด ทดสอบความถูกต้อง แสดงโปรแกรมที่แก้ไข แล้วลงในช่องด้านล่าง



ใบงานที่ 1.2

นิพจน์ทางคณิตศาสตร์และคำสั่งในการแสดงผล

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

1. การสร้างนิพจน์ทางคณิตศาสตร์จากนิพจน์ที่มีอยู่

1.1 เปิดโปรแกรม MS Visual C# และสร้างโปรเจกต์ชื่อ Expression พิมพ์โค้ดต่อไปนี้ลงไป

```
1: using System;
2: class Test
3: {
4:     static void Main()
5:     {
6:         double x = 4.0, y = 2.0;
7:         int a = 10, b = 4;
8:         Console.WriteLine(_____);
9:         Console.ReadLine();
10:    }
11: }
```

สังเกตว่าบรรทัดที่ 8 มีส่วนที่ถูกเว้นว่างไว้ให้ผู้เรียนเดาค่าของนิพจน์ต่อไปนี้และตรวจสอบคำตอบโดยแทนที่ช่องว่างด้วยนิพจน์ด้านล่างทีละตัว รันโปรแกรมเพื่อดูผลลัพธ์

นิพจน์	ค่าของนิพจน์ที่คาดเดาไว้	ผลลัพธ์ที่ได้
$x+a$		
a/b		
a/x		
y/x		
$(a+b)/b\%a$		
$9.0/5.0*(a-x)$		
$x+y-x*y$		
$57\%50/25$		

1.2 จากการทดลองข้างต้น ทั้งที่ค่า a/b และ a/x คือการนำ 10 ไปหารด้วย 4 เหมือนกัน เหตุใดผลลัพธ์ที่ได้จึงต่างกัน ผู้เรียนคิดว่าในภาษา C# มีการตีความนิพจน์ทั้งคู่แตกต่างกันอย่างไร



2. ศึกษาการใช้งานคำสั่งเกี่ยวกับการแสดงผล

พิจารณาโปรแกรมที่ไม่สมบูรณ์ต่อไปนี้

```
using System;

class SayHi
{
    static void Main()
    {
        string yourName = ____(a)____;
        uint yourAge = ____(b)____;
        Console.WriteLine("Hello {1}.  You are {0} years old.",
                           ____(c)____, ____(d)____);
    }
}
```

เติมส่วนที่เว้นไว้จาก (a) ถึง (d) เพื่อให้โปรแกรมกล่าวคำทักทายท่านแสดงอายุของท่าน เช่น

หากท่านชื่อ Arthur และมีอายุ 18 ปี ผลลัพธ์ของโปรแกรมควรจะเป็น

Hello Arthur. You are 18 years old.

จากนั้นลอกสิ่งเติมในช่องว่างลงในตาราง

ช่องว่าง	สิ่งที่เติมลงไป
__(a)__	
__(b)__	
__(c)__	
__(d)__	

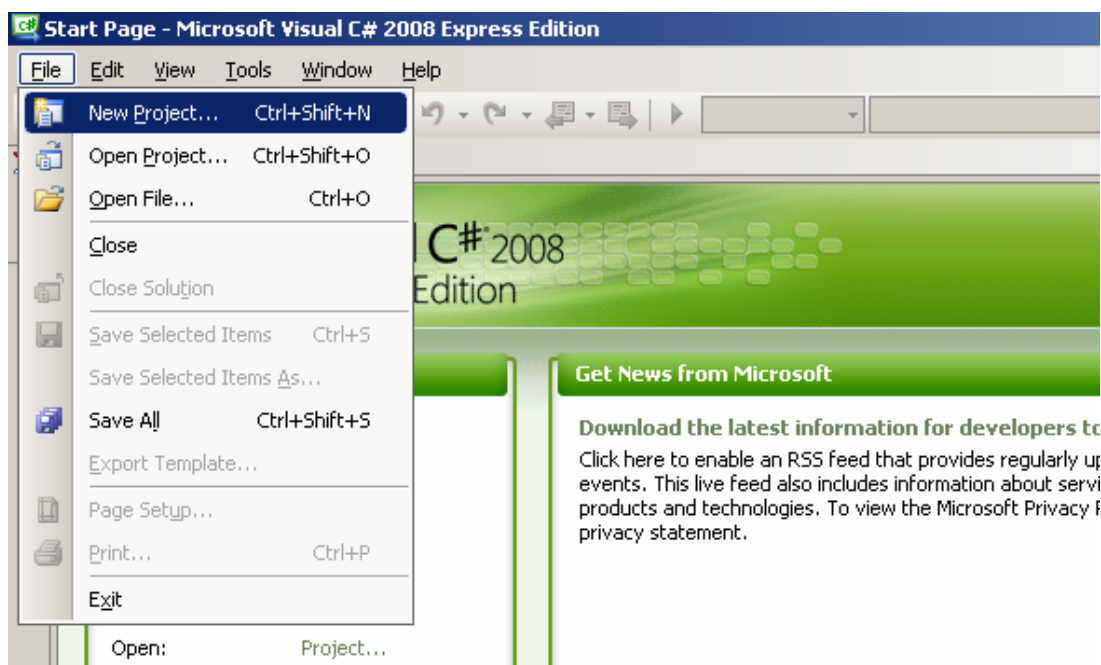


ใบความรู้ที่ 1.1

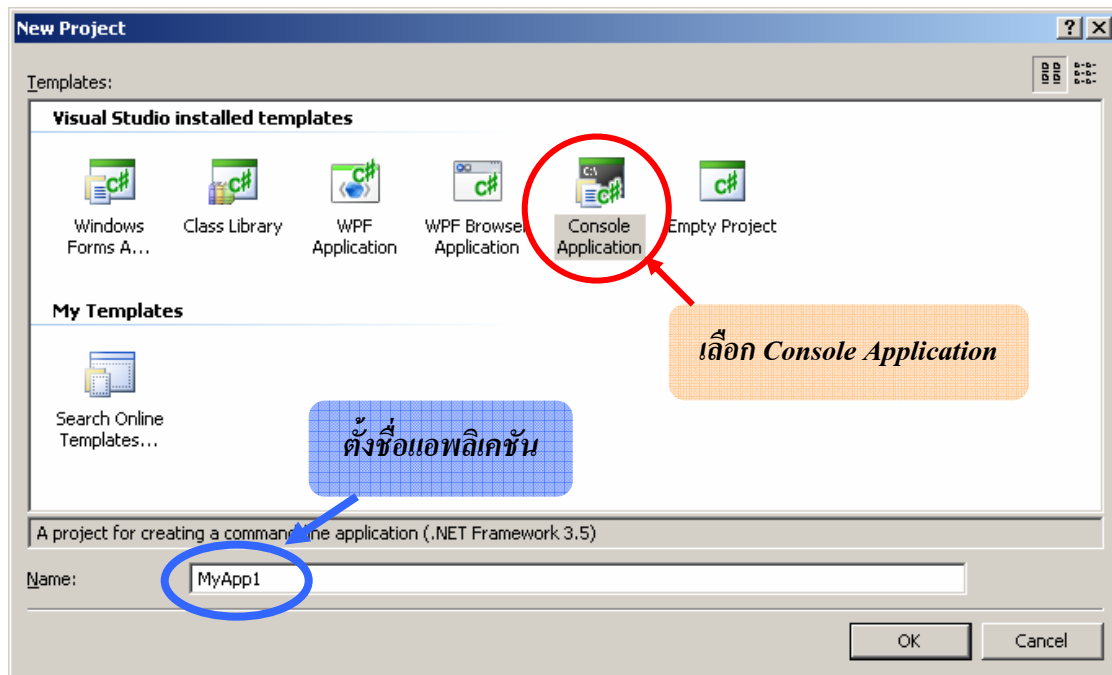
การสร้างแอปพลิเคชันด้วย MS Visual C#

ในสภาพแวดล้อมของ MS Visual C# การพัฒนาแอปพลิเคชันขึ้นมาหนึ่งชิ้นเรียกว่าการสร้างโซลูชัน (solution) ซึ่งประกอบไปด้วยโปรเจกต์ (project) ตั้งแต่หนึ่งหรือมากกว่า โดยแต่ละโปรเจกต์เป็นส่วนหนึ่งของซอฟต์แวร์ที่อาจเป็นส่วนหลักของโปรแกรม หรือส่วนไลบรารี (library) ที่ถูกเรียกใช้โดยโปรเจกต์อื่นๆ ดังนั้นแอปพลิเคชันแต่ละตัวจะประกอบไปด้วยโปรเจกต์อย่างน้อยหนึ่งโปรเจกต์เสมอ ใบความรู้นี้จะอธิบายถึงกระบวนการสร้างโปรเจกต์ใน MS Visual C# สำหรับพัฒนาแอปพลิเคชันแบบคอนโซล โดยมีขั้นตอนต่อไปนี้

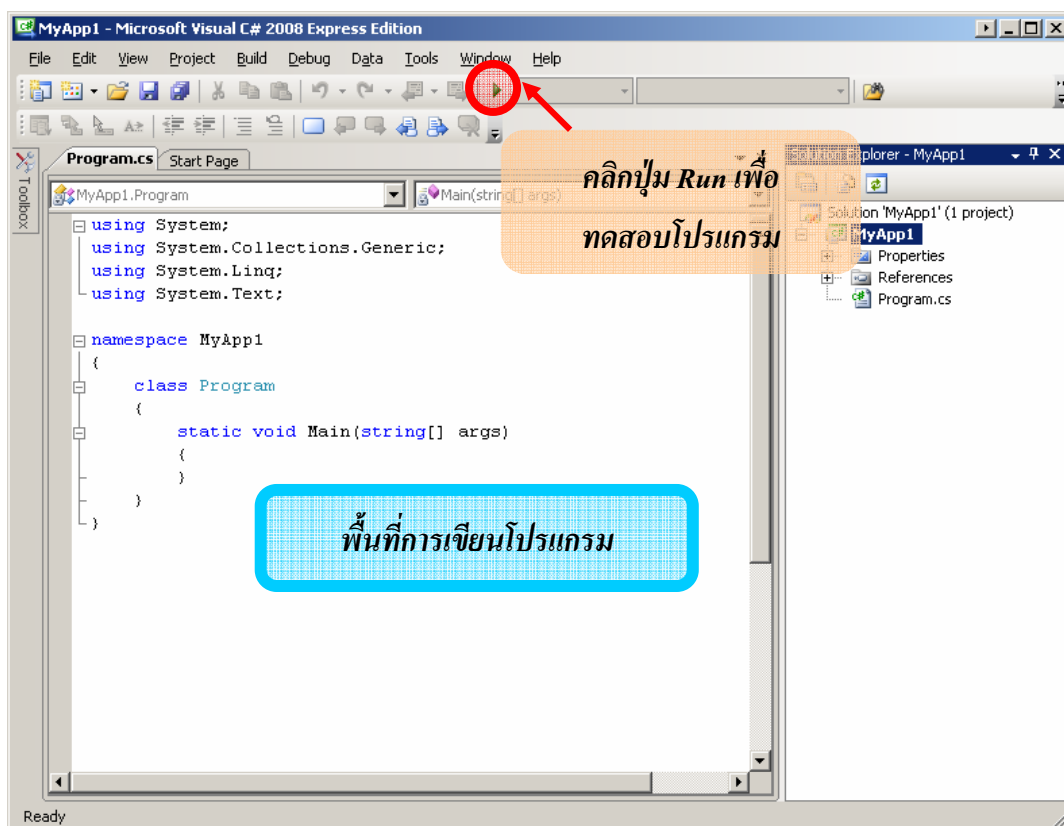
1. เลือกรายการเมนู New Project จากเมนู File ดังแสดงในรูปที่ 1.1
2. ในไดอะล็อกซ์ New Project เลือกชนิดของโปรเจกต์เป็น Console Application และตั้งชื่อให้กับโปรเจกต์ในกล่องข้อความด้านล่าง ดังแสดงตัวอย่างในรูปที่ 1.2 จากนั้นกดปุ่ม OK
3. โปรเจกต์ใหม่จะถูกสร้างขึ้นมาพร้อมทั้งไฟล์ชื่อ Program.cs ซึ่งมีโปรแกรมภาษา C# มาให้บางส่วนเพื่อให้่ายต่อการเริ่มต้น ดังแสดงในรูปที่ 1.3 ในที่นี้เราอาจลบโปรแกรมที่ให้มาทั้งหมดทิ้งไปแล้วเริ่มเขียนโปรแกรมของเราเองก็ได้ เมื่อต้องการทดสอบการทำงานของโปรแกรมให้คลิกที่ปุ่ม Run (▶) เพื่อสั่งให้ MS Visual C# คอมไพล์โปรแกรมและให้โปรแกรมเริ่มทำงาน



รูปที่ 1.1 รายการเมนูสำหรับสร้างโปรเจกต์ใหม่



รูปที่ 1.2 ไดอะล็อกซ์ในสร้างโปรเจกต์ใหม่และการตั้งค่าสำหรับคอนโซลแอปพลิเคชัน



รูปที่ 1.3 หน้าจอโดยรวมของ MS Visual C# หลังกระบวนการสร้างโปรเจกต์



ใบความรู้ที่ 1.2

ภาษา C# และเครื่องมือพัฒนาโปรแกรม

ภาษา C# เป็นภาษาโปรแกรมเชิงวัตถุ (object-oriented programming language) ที่ถูกพัฒนาขึ้นมาโดยบริษัทไมโครซอฟต์ การพัฒนาโปรแกรมคอมพิวเตอร์ด้วยภาษา C# นั้นจะประกอบด้วยขั้นตอนดังนี้

- วิเคราะห์ปัญหาและความต้องการในการพัฒนาโปรแกรม เช่น โปรแกรมจะติดต่อกับผู้ใช้อย่างไร ข้อมูลที่ผู้ใช้จะป้อนให้กับโปรแกรมเป็นอย่างไร และผลลัพธ์จะถูกแสดงผลอย่างไร
- ออกแบบขั้นตอนวิธี โดยแสดงการทำงานของโปรแกรมในภาพรวมออกมาเป็นลำดับขั้นตอน แต่ละขั้นตอนมีความชัดเจนและสามารถเปลี่ยนให้อยู่ในรูปคำสั่งภาษา C# ได้โดยง่าย
- นำขั้นตอนวิธีที่ออกแบบไว้มาสร้างเป็นไฟล์โปรแกรมรหัสต้นฉบับ (source code) ที่ถูกต้องตามโครงสร้างและไวยากรณ์ของตัวภาษา C# ทั้งนี้ไฟล์รหัสต้นฉบับต้องมีนามสกุล .cs เสมอ เช่น prog1.cs
- แปลงรหัสต้นฉบับให้อยู่ในรูปรหัสภาษาเครื่องที่คอมพิวเตอร์เข้าใจและทำงานตามคำสั่งได้ ขั้นตอนนี้ต้องใช้โปรแกรมที่เรียกว่า คอมไพเลอร์ (compiler) ไฟล์รหัสภาษาเครื่องที่ถูกสร้างขึ้นจากคอมไพเลอร์จะมีนามสกุล .exe ซึ่งย่อมาจาก executable หมายถึงไฟล์ที่ถูกเรียกทำงานได้
- ทดสอบการทำงานของโปรแกรม หากพบข้อผิดพลาดให้ตรวจสอบความถูกต้องในขั้นตอนที่ผ่านมา ซึ่งอาจหมายถึงการแก้ไขโปรแกรม ขั้นตอนวิธี หรือแม้กระทั่งวิเคราะห์ปัญหาและความต้องการใหม่

ขั้นตอนการพัฒนาโปรแกรมหากล่าวนี้ นอกจากจะสามารถใช้กับภาษา C# แล้วยังสามารถนำไปประยุกต์ใช้กับภาษาอื่น ๆ ได้ด้วย

เครื่องมือสำหรับพัฒนาโปรแกรมด้วย C#

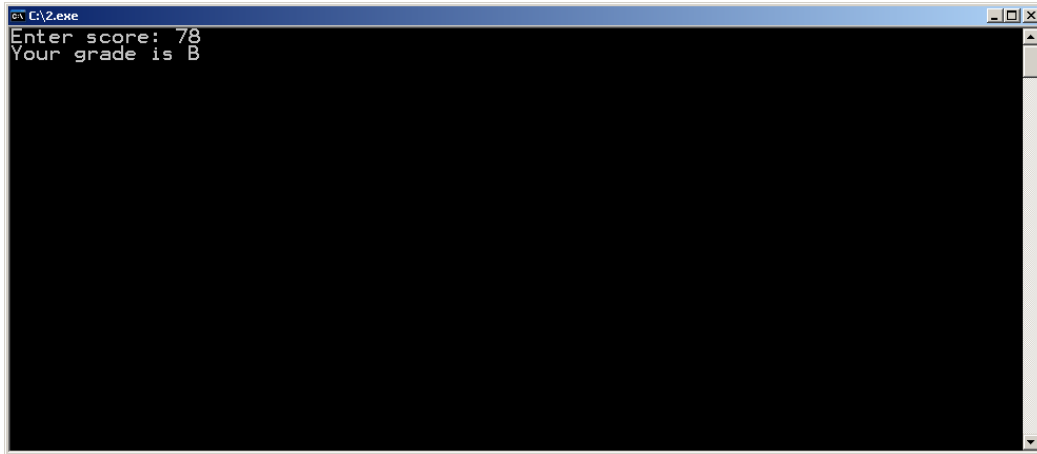
ขั้นตอนในการพัฒนาโปรแกรมที่กล่าวไปแล้วในข้างต้นอาจดูซับซ้อนสำหรับผู้ที่ยังไม่มีประสบการณ์ อย่างไรก็ตาม ในปัจจุบันได้มีซอฟต์แวร์สำหรับช่วยพัฒนาโปรแกรมภาษา C# อยู่มากมาย ให้เลือกใช้ซึ่งเพิ่มความสะดวกและลดข้อผิดพลาดลงได้เป็นอย่างมาก ซอฟต์แวร์หลายตัวถูกแจกจ่ายให้นำไปใช้งานได้โดยไม่ต้องเสียค่าใช้จ่าย หนึ่งในซอฟต์แวร์เหล่านั้นคือ Microsoft Visual C# 2008 Express Edition (ในที่นี้ขอเรียกย่อ ๆ ว่า MS Visual C#)

MS Visual C# มีคุณสมบัติคร่าว ๆ ดังต่อไปนี้

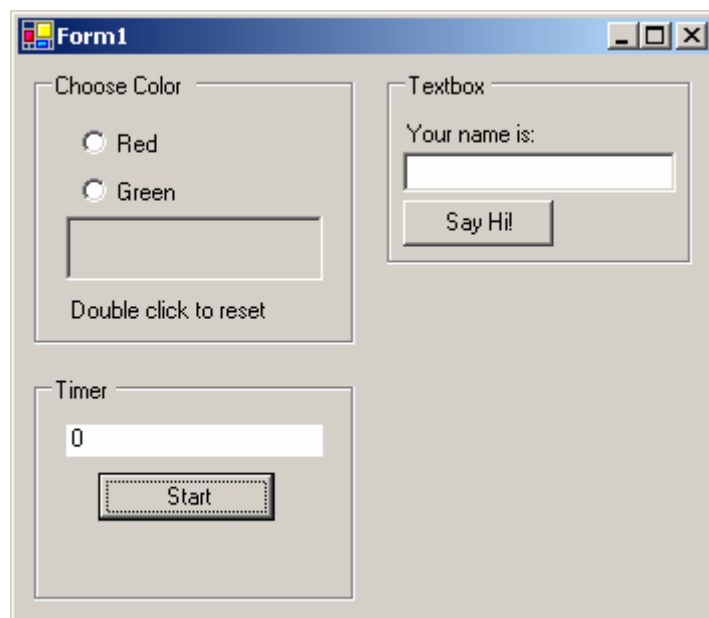
- สร้าง/แก้ไข/บันทึก โปรแกรมที่เขียนด้วยภาษา C# ได้



- คอมไพล์และทดสอบโปรแกรมที่เขียนขึ้นได้ทันที
- เพิ่มความง่ายในการเขียนโปรแกรมด้วยคุณสมบัติการเติมเต็มคำสั่ง (Code Completion)
- สนับสนุนการพัฒนาโปรแกรมทั้งประเภทที่ติดต่อกับผู้ใช้ผ่านคอนโซล (Console Application) ซึ่งรับข้อมูลผ่านแป้นพิมพ์และแสดงผลข้อมูลในรูปตัวอักษรเพียงอย่างเดียว ดังตัวอย่างในรูปที่ 1.4 และประเภทที่ใช้คุณสมบัติของวินโดวส์เต็มรูปแบบ (Windows Application) ซึ่งอนุญาตให้ผู้ใช้งานควบคุมโปรแกรมโดยใช้เมาส์และแสดงผลในแบบกราฟิกได้ ดังตัวอย่างในรูปที่ 1.5



รูปที่ 1.4 ตัวอย่างคอนโซลแอปพลิเคชัน (Console Application)



รูปที่ 1.5 ตัวอย่างวินโดวส์แอปพลิเคชัน (Windows Application)

ใบความรู้ที่ 1.3

โครงสร้างโปรแกรมภาษา C# และข้อมูลพื้นฐานและตัวดำเนินการ

การโปรแกรมภาษา C# ขึ้นพื้นฐานที่มีเฉพาะส่วนของโปรแกรมหลักและไม่มีโปรแกรมย่อย (subroutine) จะมีส่วนประกอบดังนี้

```
namespace ____ (A) ____  
{  
    class ____ (B) ____  
    {  
        static void Main()  
        {  
            ____ (C) ____  
        }  
    }  
}
```

ตามโครงสร้างข้างต้น ณ ตำแหน่ง (A), (B), and (C) มีความหมายดังต่อไปนี้

- ตำแหน่ง (A) ระบุชื่อของเนมสเปซ (namespace) ซึ่งใช้ในการกำหนดขอบเขตให้กับคลาสต่าง ๆ รวมถึงใช้ในการจัดโครงสร้างของโปรแกรมนำเข้าให้เป็นสัดส่วนอีกด้วย โดยเฉพาะอย่างยิ่งในการพัฒนาแอปพลิเคชันที่ซับซ้อนโดยผู้พัฒนาหลายคน การกำหนดเนมสเปซของตนเองสามารถป้องกันปัญหาการตั้งชื่อคลาสหรือค่าคงที่อื่น ๆ ซ้ำกันได้ ส่วนไวยากรณ์ของภาษา C# ที่มีให้เราเรียกใช้งานได้ ก็ถูกเตรียมเอาไว้ในเนมสเปซชื่อต่าง ๆ เช่นเดียวกัน
- ตำแหน่ง (B) ระบุชื่อของคลาส (class)
- ตำแหน่ง (C) เป็นพื้นที่สำหรับคำสั่งต่าง ๆ ที่ผู้เขียนโปรแกรมต้องการให้คอมพิวเตอร์ปฏิบัติตาม

นอกจากนี้โปรแกรมที่ไม่ซับซ้อนมากยังสามารถละส่วนที่ระบุเนมสเปซทิ้งไปได้ คลาสที่ถูกสร้างขึ้นมาโดยไม่อยู่ในขอบเขตของเนมสเปซใดจะถือว่าอยู่ในเนมสเปซกลาง (global namespace)

```
class ____ (B) ____  
{  
    static void Main()  
    {  
        ____ (C) ____  
    }  
}
```

โปรแกรมภาษา C# ซึ่งแสดงข้อความ Hello World! ออกทางจอภาพ จากนั้นรอกันว่าผู้ใช้จะกด Enter และจบการทำงาน โปรแกรมนี้อยู่ในเนมสเปซชื่อ HelloApp และคลาสชื่อ HelloClass

```
namespace HelloApp
{
    class HelloClass
    {
        static void Main()
        {
            System.Console.WriteLine("Hello World!");
            System.Console.ReadLine();
        }
    }
}
```

โปรแกรมภาษา C# ซึ่งให้ผลลัพธ์เช่นเดียวกับโปรแกรมข้างต้น แต่เขียนโดยไม่ระบุเนมสเปซ

```
class HelloClass
{
    static void Main()
    {
        System.Console.WriteLine("Hello World!");
        System.Console.ReadLine();
    }
}
```

กฎการตั้งชื่อตัวระบุในภาษา C#

โปรแกรมภาษา C# ที่นำไปใช้งานจริงมักจะมีการใช้งานตัวระบุ (identifier) อยู่ทั่วไปภายในโปรแกรม เช่นชื่อของเนมสเปซและคลาสที่ได้กล่าวไปข้างต้น ภาษา C# ได้จำกัดกฎเกณฑ์การตั้งชื่อให้ตัวระบุเอาไว้ดังต่อไปนี้

- ชื่อตัวระบุต้องประกอบด้วยตัวอักษรภาษาอังกฤษ (A-Z,a-z) ตัวเลข (0-9) หรือเครื่องหมายขีดเส้นใต้ (_) เท่านั้น
- ตัวอักษรตัวแรกของชื่อต้องเป็นตัวอักษรภาษาอังกฤษ หรือตัวขีดเส้นใต้
- ชื่อตัวระบุจะมีความยาวได้ไม่เกิน 63 ตัวอักษร
- ชื่อตัวระบุต้องไม่ซ้ำกับคำสงวน (reserved word) เช่น **class**, **namespace**, **int**, **void**, **static**



ชนิดข้อมูล (Data Types) ใน C#

C# กำหนดชนิดของข้อมูลไว้หลากหลายชนิดเพื่อรองรับการจัดเก็บข้อมูลหลาย ๆ ประเภท ตารางที่ 1 แสดงชนิดข้อมูลหลัก ๆ ที่พบบ่อยในโปรแกรมทั่วไป

ตารางที่ 1 ชนิดข้อมูลหลัก ๆ ของภาษา C#

ชนิดข้อมูล	คำอธิบาย
char	อักขระเดี่ยว เช่น a
bool	ค่าความจริง เป็นไปได้สองค่าคือ true หรือ false
byte	จำนวนเต็มไม่มีเครื่องหมาย ตั้งแต่ 0 ถึง 255
int	จำนวนเต็มมีเครื่องหมาย ตั้งแต่ - 2,147,483,648 ถึง 2,147,483,647
uint	จำนวนเต็มไม่มีเครื่องหมาย ตั้งแต่ 0 ถึง 4,294,967,295
long	จำนวนเต็มมีเครื่องหมาย ตั้งแต่ - 9,223,372,036,854,775,808 ถึง 9,223,372,036,854,775,807
ulong	จำนวนเต็มไม่มีเครื่องหมาย ตั้งแต่ 0 and 18,446,744,073,709,551,615
float	จำนวนจริง (มีทศนิยมได้) เช่น 3.14159
double	จำนวนจริงที่เก็บความละเอียดมากเป็นสองเท่า
string	ข้อความ (สายอักขระ) เช่น Hello

ตัวแปร (Variables)

ตัวแปร (variable) เป็นตัวระบุประเภทหนึ่งที่ใช้ในการอ้างถึงข้อมูล โดยค่าของมันสามารถถูกเปลี่ยนแปลงได้ตลอดเวลาที่โปรแกรมกำลังทำงานอยู่ ในภาษา C# ตัวแปรทุกตัวต้องถูกประกาศก่อนที่จะถูกนำมาใช้งาน โดยมีการระบุชนิดข้อมูลที่จะใช้กับตัวแปรนั้น ๆ ไว้ด้วยตามรูปแบบดังนี้

```
DataType variableName;
```

ในที่นี้ *variableName* คือชื่อของตัวแปร และ *DataType* คือชื่อของชนิดข้อมูลที่ตัวแปรนี้เก็บค่าได้ (ตามตัวอย่างในตารางที่ 1)

นอกจากการประกาศตัวแปรตามแบบข้างต้นแล้ว เรายังสามารถกำหนดค่าเริ่มต้นให้กับตัวแปรนั้น ๆ ได้อีกด้วย โดยมีรูปแบบดังนี้

```
DataType variableName = initialValue;
```

จะเห็นว่าส่วนที่เพิ่มขึ้นมาคือเครื่องหมาย = และ *initialValue* ซึ่งระบุค่าเริ่มต้นให้กับตัวแปรชื่อ *variableName*

คำสั่งด้านล่างประกาศตัวแปรชื่อ *distance* ที่มีชนิดข้อมูลเป็น **uint** และไม่มีการกำหนดค่าเริ่มต้น

```
uint distance;
```

คำสั่งด้านล่างประกาศตัวแปรชื่อ *salary* ที่มีชนิดข้อมูลเป็น **long** และค่าเริ่มต้นเป็น 30000

```
long salary = 30000;
```

ค่าคงที่ (Constants)

ค่าคงที่เป็นตัวระบุอีกประเภทหนึ่งที่น่าสนใจในการอ้างอิงข้อมูลเช่นเดียวกับตัวแปร สิ่งที่แตกต่างกันจากตัวแปรก็คือค่าของมันไม่สามารถเปลี่ยนแปลงได้อีกหลังจากการประกาศ

ในภาษา C# ค่าคงที่ต้องถูกประกาศโดยระบุชนิดข้อมูลและค่าตั้งต้นก่อนถูกนำมาใช้งานเสมอ การประกาศค่าคงที่จะคล้ายคลึงกับการประกาศตัวแปร แตกต่างกันตรงที่ต้องมีการระบุด้วยคีย์เวิร์ด **const** ดังแสดง

```
const DataType constantName = value;
```

ในที่นี้ *constantName* คือชื่อของค่าคงที่ ส่วน *DataType* คือชื่อชนิดข้อมูลที่ค่าคงที่นี้ใช้อ้างอิง และ *value* ระบุค่าที่ค่าคงที่นี้ถูกใช้เป็นตัวแทน

คำสั่งด้านล่างประกาศค่าคงที่ชื่อ *myconst* โดยมีชนิดข้อมูลเป็นแบบ **double** และใช้แทนค่า 2.71828

```
const double myconst = 2.71828;
```

การใช้งานค่าคงที่มีประโยชน์มากในกรณีที่เราต้องอ้างอิงถึงค่าใด ๆ ซ้ำกันอยู่บ่อยครั้งภายในโปรแกรม ดังเช่นตัวอย่างต่อไปนี้

โปรแกรมด้านล่างเป็นโปรแกรมที่ทำให้คอมพิวเตอร์กล่าวคำทักทายทั่ว ๆ ไปสองบรรทัดกับผู้ใช้ ซึ่งสมมติว่าชื่อ Harry จะเห็นว่าหากเราต้องการเปลี่ยนชื่อผู้ใช้เป็นชื่ออื่นนั้นก็สามารทำได้ง่าย ๆ โดยแก้ไขบรรทัด ที่ 5 ให้ค่าคงที่แทนด้วยค่าอื่นเท่านั้น

```
1: using System;
2:
3: class Intro {
4:     static void Main() {
5:         const string MY_NAME = "Harry";
6:
7:         Console.WriteLine("Hello {0}, how are you?", MY_NAME);
8:         Console.WriteLine("And how can I help you, {0}?",
9:             MY_NAME);
10:        Console.ReadLine();
11:    }
12: }
```



ตัวแปร)รวมถึงค่าคงที่ (ที่มีชนิดข้อมูลแบบเดียวกันสามารถถูกประกาศร่วมกันภายในคำสั่งเดียวกันได้โดยกันด้วยเครื่องหมายคอมม่า

```
const int FreezingPoint = 32;
int x, y;
int wd = 5, ht = 8;
const double PI = 3.1424;
char ch = 'A';
string mynote = "Hello, Kitty";
int j = 5;
```

นิพจน์ทางคณิตศาสตร์ (Arithmetic Expressions)

นิพจน์ (expression) ภายในโปรแกรมภาษา C# หมายถึงส่วนของโปรแกรมที่สามารถถูกตีความเป็นค่าต่าง ๆ ได้ โดยนิพจน์อาจประกอบด้วยเทอมเพียงเทอมเดียว หรือเกิดจากการผสมกันของนิพจน์อื่นที่เล็กกว่า ด้านล่างเป็นตัวอย่างของนิพจน์ที่ประกอบด้วยเทอมเพียงเทอมเดียว

- ตัวเลขโคด เช่น 3000, 1.414
- ข้อความ เช่น "Hello, World"
- ค่าความจริง ได้แก่ **true** และ **false**
- ตัวแปรหรือค่าคงที่เดี่ยว ๆ ที่ผ่านการกำหนดค่าไว้แล้ว เช่น *myName*, *salary*

ในที่นี้จะเน้นการศึกษาเกี่ยวกับนิพจน์ที่ให้ค่าเป็นตัวเลข เรียกว่า นิพจน์ทางคณิตศาสตร์ (arithmetic expression) โดยเราสามารถผสมนิพจน์ต่าง ๆ ให้เป็นนิพจน์ที่ซับซ้อนขึ้นโดยอาศัยตัวดำเนินการทางคณิตศาสตร์ที่มีให้ในภาษา C# ดังตารางที่ 2

ตารางที่ 2 ตัวดำเนินการทางคณิตศาสตร์

ตัวดำเนินการ	สัญลักษณ์	จำนวนนิพจน์ที่นำมาผสม	ตัวอย่าง	ค่าของนิพจน์เมื่อ $x = 20$
บวก (add)	+	2	$x+4$	24
ลบ (subtract)	-	2	$32-x$	12
คูณ (multiply)	*	2	$x*2$	40
หาร (divide)	/	2	$x/2$	10
หารเอาเศษ (modulo)	%	2	$x\%6$	2

ตัวดำเนินการ	สัญลักษณ์	จำนวนนิพจน์ที่นำมาผสม	ตัวอย่าง	ค่าของนิพจน์เมื่อ $x = 20$
กลับเครื่องหมาย (negate)	-	1	$-x$	-20
จัดกลุ่ม	()	N/A	$(x+2) * 3$	66

หากนิพจน์ที่สร้างขึ้นใหม่ประกอบด้วยตัวดำเนินการมากกว่าหนึ่งตัว C# จะคำนวณค่าเรียงตามลำดับก่อนหลังดังนี้

- ()
- *, / และ %
- + และ -
- หากตัวดำเนินการมีลำดับเท่าเทียมกัน คำนวณจากซ้ายไปขวา

คำสั่งที่ใช้ในการแสดงผล

คำสั่งหลักที่ใช้ในการแสดงข้อความออกทางจอภาพได้แก่คำสั่ง *Write* และ *WriteLine* ซึ่งถูกนิยามไว้ในคลาสชื่อ *Console* และเนมสเปซชื่อ *System* สองคำสั่งนี้มีการใช้งานในลักษณะเดียวกัน แตกต่างกันที่คำสั่ง *WriteLine* จะพิมพ์ข้อความตามด้วยการขึ้นบรรทัดใหม่หลังจากพิมพ์เสร็จ ลองพิจารณาการใช้งานจากตัวอย่างต่อไปนี้

การเรียกใช้งานคำสั่ง *Write* และ *WriteLine* โดยระบบเนมสเปซเต็มรูปแบบ

```
class Hello
{
    static void Main()
    {
        System.Console.Write("Hello, ");
        System.Console.WriteLine("everybody");
    }
}
```

สังเกตว่าการเรียกใช้คำสั่ง *Write* และ *WriteLine* นั้นต้องระบุถึงคลาส *Console* ซึ่งเป็นคลาสที่นิยามคำสั่งนี้เอาไว้ และเนื่องจากคลาส *Console* เป็นคลาสที่อยู่ในเนมสเปซ *System* จึงต้องระบุชื่อเนมสเปซไว้ด้วยตามตัวอย่าง อย่างไรก็ตามเราสามารถทำโปรแกรมให้สั้นลงได้โดยใช้คำสั่ง **using** ซึ่งเป็นการสั่งให้คอมไพเลอร์ค้นหาคลาสที่เรียกใช้งานในเนมสเปซที่ระบุ ดังตัวอย่างต่อไปนี้

การเรียกใช้งานคำสั่ง *Write* และ *WriteLine* โดยใช้คำสั่ง **using**

```
using System;
```



```

class Hello
{
    static void Main()
    {
        Console.Write("Hello, ");
        Console.WriteLine("everybody");
    }
}

```

คำสั่ง *Write* และ *WriteLine* นั้นยังสามารถเรียกใช้งานในรูปแบบที่ซับซ้อนกว่านี้อีกมาก ดังแสดงในตัวอย่างต่อไปนี้

การเรียกใช้งานคำสั่ง *Write* และ *WriteLine* โดยใช้สตริงกำหนดรูปแบบ (*formatting string*)

```

using System;

class Hello
{
    static void Main()
    {
        int width = 80, height = 30;
        Console.WriteLine("Area of {0}x{1} rectangle = {2}",
            width, height, width*height);
    }
}

```

จะเห็นว่าสตริงที่เป็นพารามิเตอร์ตัวแรกของคำสั่ง *WriteLine* นั้นเป็นสตริงที่ใช้สำหรับกำหนดรูปแบบในการแสดงผลของนิพจน์ที่ตามมาอีกสามตัว สัญลักษณ์ {0} {1} และ {2} คือตำแหน่งที่ C# จะนำค่าของนิพจน์ *width*, *height* และ *width*height* มาแทนที่

ความรู้เพิ่มเติม

สตริงกำหนดรูปแบบ (Formatting String)

การแสดงผลลัพธ์ทางจอภาพด้วยคำสั่ง *Write* หรือ *WriteLine* ที่มีพารามิเตอร์มากกว่าหนึ่งตัวนั้นบางครั้งเราต้องการแสดงผลลัพธ์ของนิพจน์พร้อมกับการจัดรูปแบบการแสดงผล เช่น การระบุตำแหน่งการแสดงผลนิพจน์ในข้อความหรือการระบุความกว้าง) จำนวนตัวอักษร (สำหรับการแสดงผลหรือการจัดรูปแบบชิดซ้ายหรือการจัดชิดขวา ในภาษา C# เราสามารถทำสิ่งเหล่านี้ได้โดยใช้วิธีการกำหนดรูปแบบ ซึ่งในภาษา C# จะมี สตริงกำหนดรูปแบบ (*formatting string*) เพื่อใช้สำหรับแสดงผลลัพธ์ของนิพจน์ตามที่เรต้องการ โดยทั่วไปแล้วสตริงกำหนดรูปแบบจะมีรูปแบบดังนี้

`"{index [,alignment] [:formatSpecifier]}"`

จากนั้นตามด้วยลำดับของอาร์กิวเมนต์ เช่น

`Console.WriteLine(" Two sample integers are {0} and {1}. ", 3, 9)`

ถูกนำไปแสดงผล

จะแสดงผลลัพธ์เป็น "Two sample integers are 3 and 9"

โดยที่ *index* เป็นจำนวนเต็มเริ่มต้นที่ศูนย์ใช้สำหรับระบุอาร์กิวเมนต์ที่ต้องการจะจัดรูปแบบ เช่น ถ้า *index* มีค่าเป็น 0 จะหมายถึงค่าของอาร์กิวเมนต์ตัวแรกจะถูกแสดงในตำแหน่งที่ *index* ปรากฏอยู่ และถ้า *index* มีค่าเป็น 1 จะหมายถึงค่าของอาร์กิวเมนต์ตัวที่สองจะถูกแสดงในตำแหน่งที่ *index* ปรากฏอยู่ เป็นต้น ส่วน *alignment* เป็นจำนวนเต็มที่ใช้สำหรับระบุความกว้างหรือจำนวนตัวอักษรที่จะแสดงผลพร้อมกับการจัดการแสดงผลแบบชิดซ้ายหรือชิดขวา ถ้า *alignment* มีค่าเป็นลบจะจัดชิดซ้ายและถ้า *alignment* มีค่าเป็นบวกจะจัดชิดขวา และตัวสุดท้าย *formatSpecifier* คือตัวกำหนดรูปแบบ (*format specifier*) ใช้สำหรับกำหนดรูปแบบการแสดงผลของนิพจน์ เช่น แสดงผลลัพธ์ของนิพจน์ในรูปแบบทศนิยมเป็นต้น ส่วนเครื่องหมาย `[]` หมายถึงจะมีหรือไม่มีส่วนนี้ก็ได้ ตัวกำหนดรูปแบบพื้นฐานสำหรับตัวเลขที่สำคัญแสดงในตารางต่อไปนี้



อักขระกำหนดรูปแบบ (<i>Format Character</i>)	ความหมาย
<i>E</i> หรือ <i>e</i>	<i>Exponential</i> (แสดงผลในรูปแบบตัวเลขทางวิทยาศาสตร์)
<i>F</i> หรือ <i>f</i>	<i>Fixed-point</i> (แสดงผลในรูปแบบทศนิยม)
<i>G</i> หรือ <i>g</i>	<i>General</i> (แสดงผลในรูปแบบทั่วไป เช่นตัวเลขจะถูกแสดงผลในรูปแบบสั้นที่สุด)
<i>N</i> หรือ <i>n</i>	<i>Number</i> (แสดงผลในรูปแบบตัวเลขเหมือนกับ <i>Fixed-point</i> แต่จะใส่เครื่องหมาย <i>comma</i> คั่นทุก ๆ 3 หลัก)
<i>P</i> หรือ <i>p</i>	<i>Percentage</i> (ตัวเลขจะถูกเปลี่ยนอยู่ในรูปแบบของเปอร์เซ็นต์)
<i>X</i> หรือ <i>x</i>	<i>Hexadecimal</i> (แสดงผลในรูปแบบของเลขฐานสิบหก)

ตัวอย่าง การเรียกใช้งานคำสั่ง *Write* และ *WriteLine* โดยใช้สตริงกำหนดรูปแบบ

```
using System;
class MainClass
{
    public static void Main(string[] args)
    {
        Console.WriteLine("x = {0} y = {1}", 123, 456);
        Console.WriteLine("123456789");
        Console.WriteLine("{0,9}", 123);
        Console.WriteLine("123456789");
        Console.WriteLine("{0,-9}", 123);
        Console.WriteLine("123456789123456789");
        Console.WriteLine("{0,-9}{0,9}", 123, 456);
    }
}
```

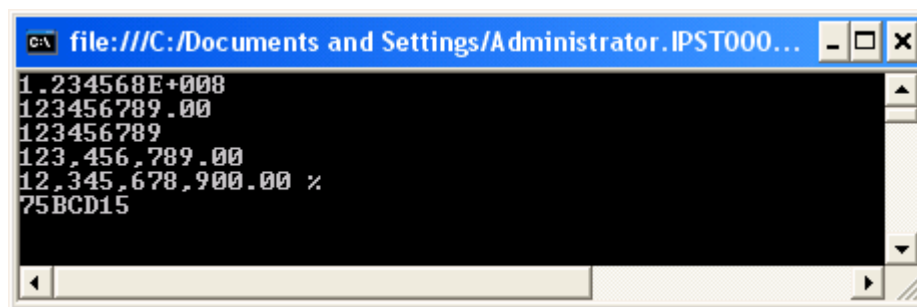
ผลลัพธ์จากการรันโปรแกรม

```
C:\ file:///C:/Documents and Settings/Administrator.IPST0001/Local Setti...
x= 123 y = 456
123456789
123
123456789
123
123456789123456789
123 123
```

ตัวอย่าง การเรียกใช้งานคำสั่ง *Write* และ *WriteLine* โดยใช้สตริงกำหนดรูปแบบพร้อมกับตัวกำหนดรูปแบบ

```
using System;
class MainClass
{
    public static void Main(string[] args)
    {
        int n = 123456789;
        Console.WriteLine("{0:E}", n);
        Console.WriteLine("{0:F}", n);
        Console.WriteLine("{0:G}", n);
        Console.WriteLine("{0:N}", n);
        Console.WriteLine("{0:P}", n);
        Console.WriteLine("{0:X}", n);
    }
}
```

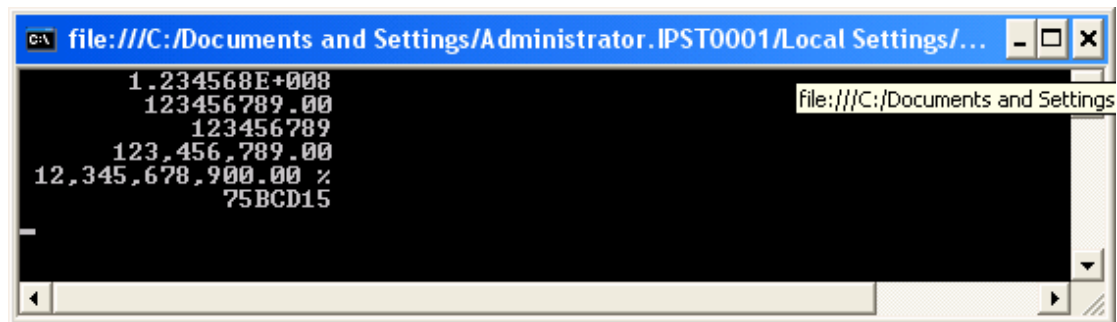
ผลลัพธ์จากการรันโปรแกรม



ตัวอย่าง การเรียกใช้งานคำสั่ง *Write* และ *WriteLine* โดยใช้สตริงกำหนดรูปแบบพร้อมกับการกำหนดความกว้าง และใช้ตัวกำหนดรูปแบบพร้อมกัน

```
using System;
class MainClass
{
    public static void Main(string[] args)
    {
        int n = 123456789;
        Console.WriteLine("{0,20:E}", n);
        Console.WriteLine("{0,20:F}", n);
        Console.WriteLine("{0,20:G}", n);
        Console.WriteLine("{0,20:N}", n);
        Console.WriteLine("{0,20:P}", n);
        Console.WriteLine("{0,20:X}", n);
    }
}
```

จากโปรแกรมจะกำหนดให้แสดงผลกว้าง 20 ตัว ผลลัพธ์จากการรันโปรแกรมจะเป็นดังนี้



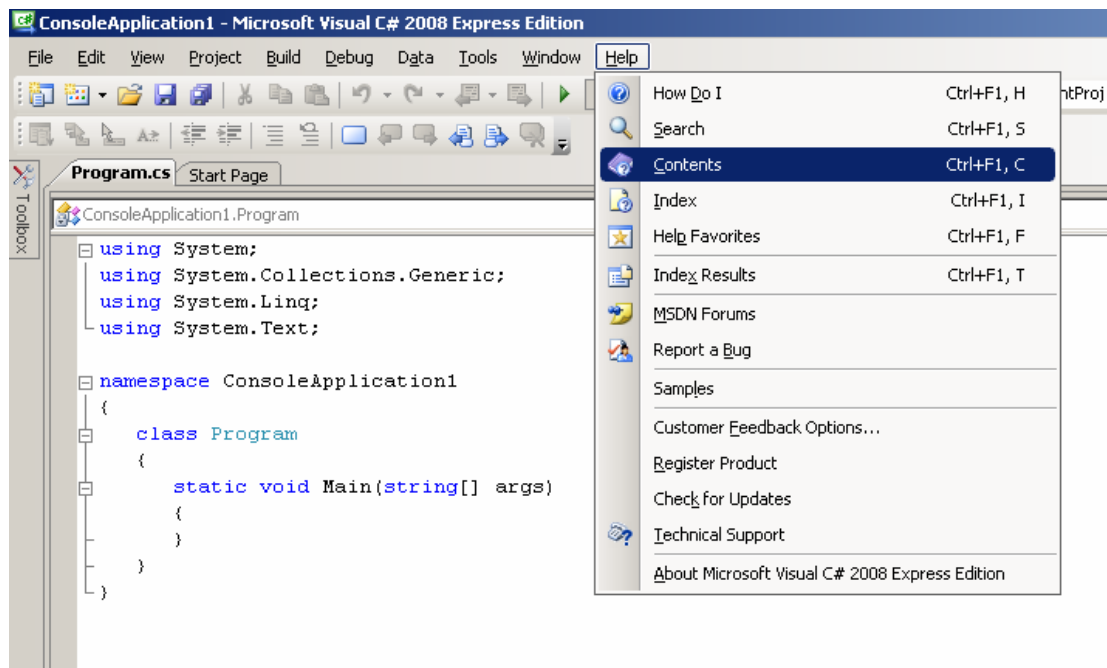
```
file:///C:/Documents and Settings/Administrator/IPST0001/Local Settings/...
1.234568E+008
123456789.00
123456789
123,456,789.00
12,345,678,900.00 %
75BCD15
```

นอกจากนี้เรายังสามารถระบุจำนวนของตัวเลขหลังจุดทศนิยมที่ต้องการแสดงออกทางจอภาพได้ โดยใส่ตัวเลขจำนวนเต็มหลังตัวกำหนดรูปแบบ เช่น `Console.WriteLine("{0:F2}", 123.4500)`; ตัวเลข 2 หลังตัวอักษร *F* หมายถึงให้แสดงผลตัวเลข 123.4500 เป็นทศนิยมสองตำแหน่งนั่นคือ 123.45

ใบความรู้ที่ 1.4

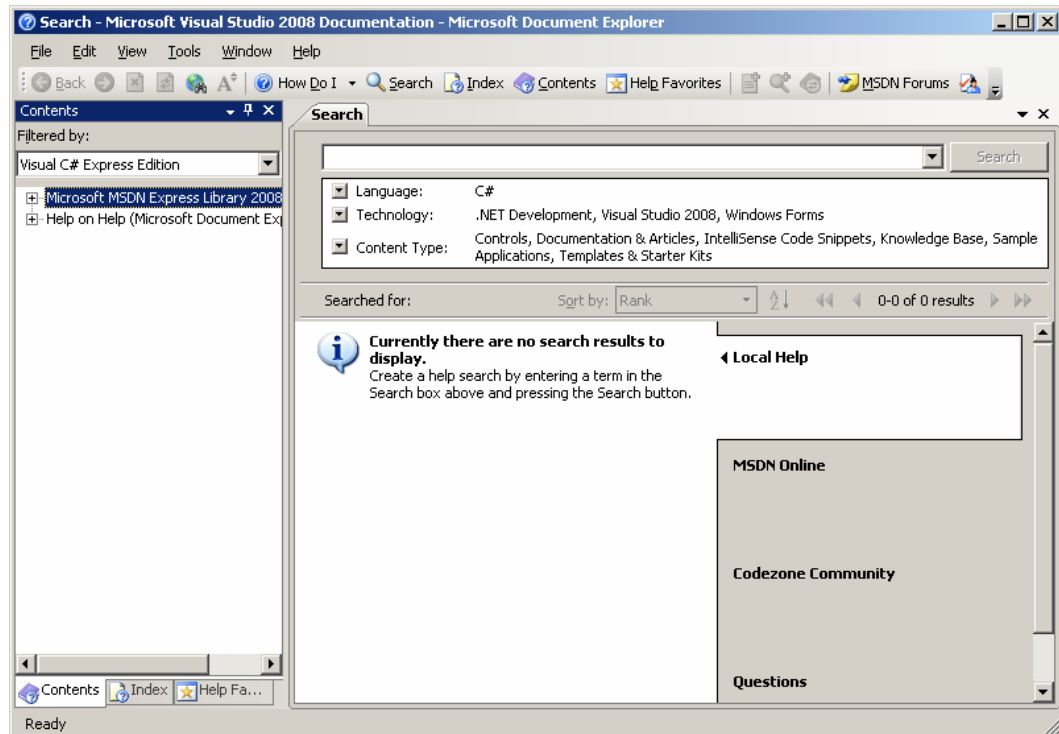
การขอความช่วยเหลือ

โปรแกรม MS Visual C# 2008 Express (รวมถึงโปรแกรมอื่นๆ ในชุด MS Visual Studio 2008 Express) มีการเชื่อมต่อกับระบบไลบรารีของ Microsoft Developer Network (MSDN Library) ซึ่งรวบรวมเอกสารเชิงเทคนิคสำหรับการพัฒนาซอฟต์แวร์บนไมโครซอฟต์วินโดวส์ไว้โดยละเอียด หาก MSDN Library ได้ถูกติดตั้งไว้พร้อมกับการติดตั้ง MS Visual C# ผู้พัฒนาโปรแกรมก็สามารถเรียก MSDN Library จากเมนู Help เพื่อเปิดเอกสารเพื่ออ่านรายละเอียดคุณสมบัติและการใช้งานคลาสต่าง ๆ ที่ถูกเตรียมไว้ใน .NET Framework ได้ รูปที่ 1.6 แสดงการเรียกใช้งาน MSDN Library จากหน้าจอหลักของ MS Visual C# ซึ่งจะได้หน้าจอคล้ายคลึงกับรูปที่ 1.7

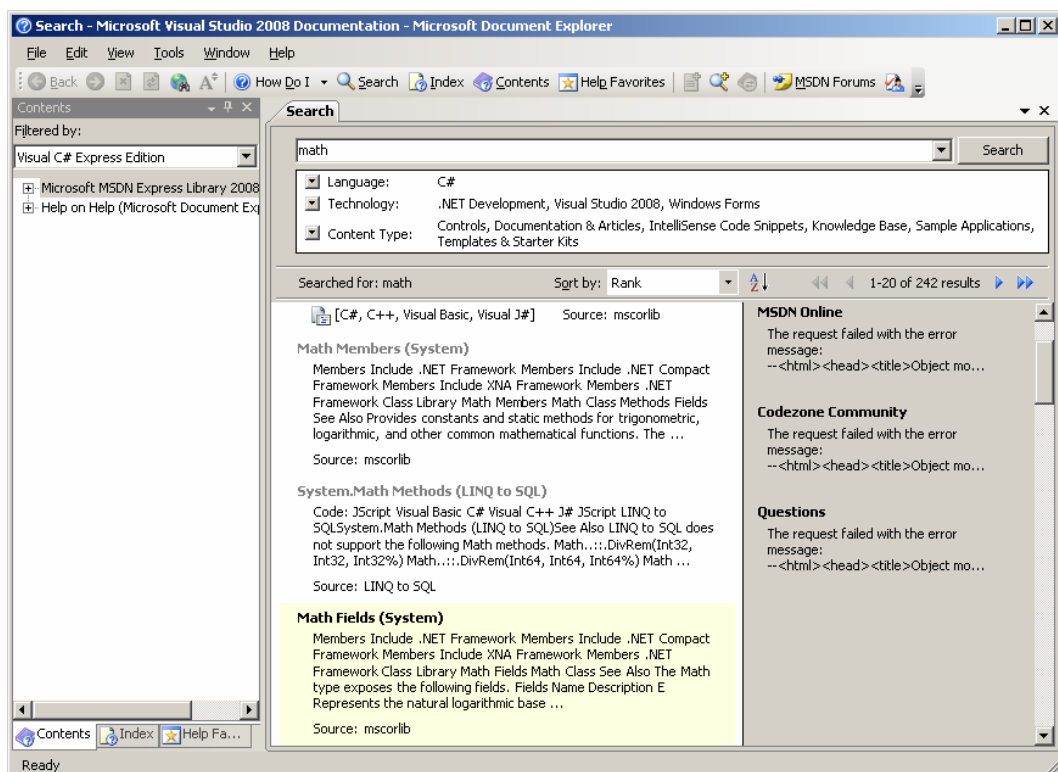


รูปที่ 1.6 การเรียกใช้งาน MSDN Library จากหน้าจอหลักของ MS Visual C#

การค้นเอกสารจาก MSDN Library นั้นสามารถทำได้หลายรูปแบบ ผู้พัฒนาโปรแกรมอาจเลือกหัวข้อจากสารบัญทางด้านซ้ายมือ หรือป้อนคำค้นในกล่อง Search เพื่อค้นหาหัวข้อที่เกี่ยวข้อง รูปที่ 1.8 แสดงผลลัพธ์จากการค้นหาด้วยคำค้น Math

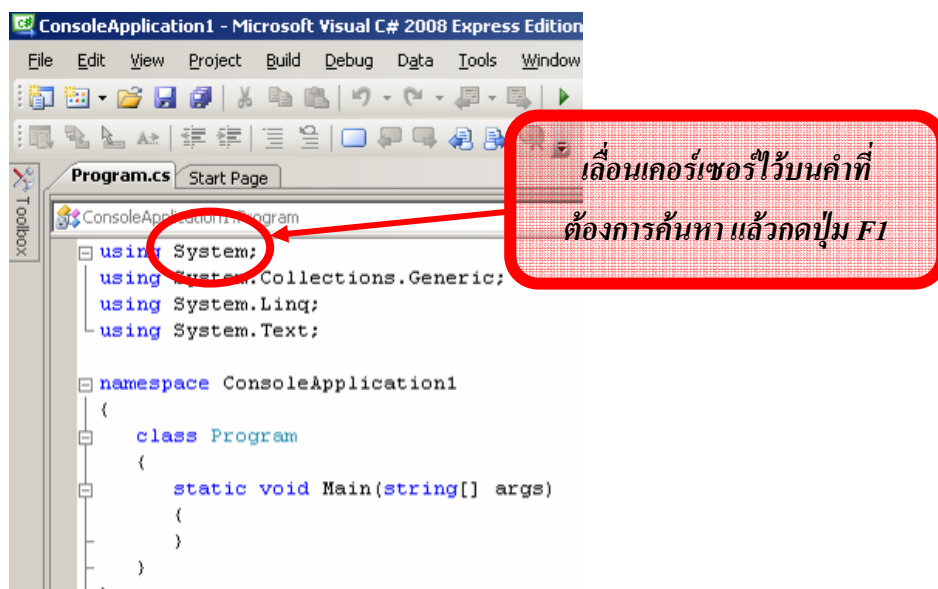


รูปที่ 1.7 หน้าจอของ MSDN Library ในส่วนของการค้นหา

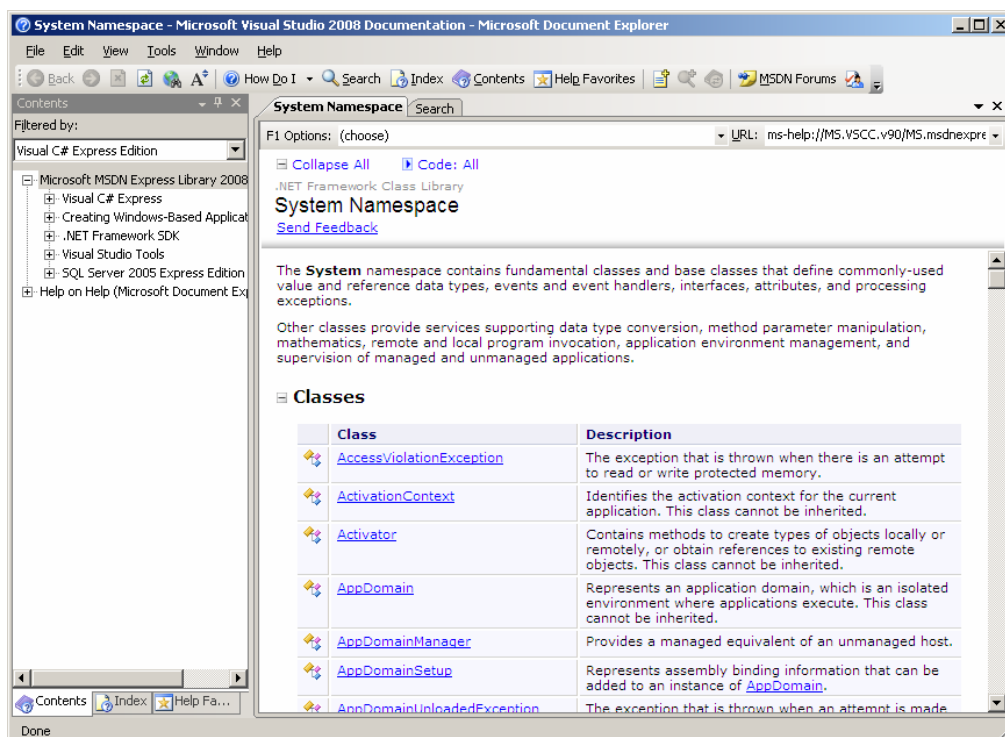


รูปที่ 1.8 ผลลัพธ์จากการป้อนคำค้น Math

นอกจากการเข้าสู่ MSDN Library ผ่านทางเมนู Help แล้ว ผู้พัฒนาโปรแกรมยังสามารถค้นหา รายละเอียดของชื่อเนมสเปส คลาส คีย์เวิร์ด หรือเมทอดที่ต้องการได้จากพื้นที่การเขียนโปรแกรมโดยตรง โดยเลื่อนเคอร์เซอร์มาวางไว้ในคำที่ต้องการสืบค้น แล้วกดปุ่ม F1 ดังแสดงตัวอย่างในรูปที่ 1.9 ซึ่งจะได้ผลลัพธ์เป็นรายการของคลาสที่อยู่ในเนมสเปส System ดังรูปที่ 1.10



รูปที่ 1.9 การขอความช่วยเหลือโดยใช้ปุ่ม F1



รูปที่ 1.10 รายชื่อคลาสในเนมสเปซ System หลังใช้คำสั่งค้นหาผ่าน MSDN Library



กิจกรรมที่ 2

ฟังก์ชันทางคณิตศาสตร์และการติดต่อกับผู้ใช้

1. จุดประสงค์ ให้ผู้เรียนสามารถ

- 1.1 เขียนโปรแกรมเพื่อตรวจสอบค่าของนิพจน์ของซีชาร์ป
- 1.2 แปลงสูตรทางคณิตศาสตร์ให้เป็นนิพจน์ของซีชาร์ป
- 1.3 เขียนโปรแกรมรับข้อมูล คำนวณ และแสดงผลลัพธ์

2. แนวคิด

คลาส Math อยู่ภายใต้เนมสเปส System ซึ่งเป็นคลาสที่รวบรวมเมทอด และค่าคงที่ทางคณิตศาสตร์ที่สำคัญ ผู้เรียนจะได้ทดลองนำฟังก์ชันต่าง ๆ เหล่านี้มาผสมผสานกันเพื่อให้เกิดการคำนวณทางคณิตศาสตร์ที่ต้องการได้ อาทิเช่นการคำนวณพื้นที่ของรูปทรงเรขาคณิต และการหารากของสมการกำลังสอง เป็นต้น

การรับข้อมูลจากผู้ใช้เข้ามาประมวลผลในภาษา C# ทำได้โดยการเรียกใช้เมทอด `Console.ReadLine` ซึ่งอยู่ในรูปของนิพจน์ที่ให้ค่าเป็นข้อความ (*String*) และสามารถนำค่าไปใส่ให้กับตัวแปรแบบสตริงหรือนำไปใช้ผสมกับนิพจน์อื่น ๆ ที่เกี่ยวข้องกับสตริงได้

ในภาษา C# ไม่มีคำสั่งที่รับข้อมูลชนิดตัวเลขโดยตรง แต่สามารถใช้เมทอด *Parse* ในการแปลงข้อมูลชนิดข้อความให้เป็นชนิดตัวเลขได้ โดยการใช้งานเมทอด *Parse* จะอยู่ในรูปของนิพจน์ที่ให้ชนิดข้อมูลตามที่ระบุไว้ใน `<numeric_datatype>` โดยมีค่าสอดคล้องกับค่าที่ระบุในนิพจน์ `<string_expression>` ดังนั้นค่าจากเมทอด *Parse* อาจนำไปกำหนดให้ตัวแปรโดยตรงหรือนำไปผสมกับนิพจน์อื่นๆ เพื่อสร้างเป็นนิพจน์ใหม่ก็ได้

3. สื่ออุปกรณ์

3.1 ใบงาน

ใบงานที่	เรื่อง	เวลา (นาที)
2.1	ฟังก์ชันทางคณิตศาสตร์	20
2.2	แก้สมการกำลังสอง	20
2.3	คุณสมบัติของวงกลม	20



3.2 ใบความรู้

- ใบความรู้ที่ 2.1 เรื่องฟังก์ชันทางคณิตศาสตร์
- ใบความรู้ที่ 2.2 เรื่องคำสั่งสำหรับรับข้อมูลเข้า

3.3 อื่นๆ

- เครื่องคอมพิวเตอร์เท่ากับจำนวนกลุ่ม
- โปรแกรมวิชวลซีชาร์ป เอ็กเพรส

4. วิธีดำเนินการ

4.1 การจัดเตรียม

- 4.1.1 แบ่งผู้เรียนเป็นกลุ่ม กลุ่มละ 2 คน
- 4.1.2 เตรียมใบงานที่ 2.1 - 2.3 ตามจำนวนกลุ่มและใบความรู้ที่ 2.1 - 2.2 ตามจำนวนผู้เรียน

4.2 ขั้นตอนการดำเนินการ

- 4.2.1 ผู้สอนกล่าวถึง Class Math ที่มีเมทอด และค่าคงที่ทางคณิตศาสตร์
- 4.2.2 ผู้เรียนแต่ละคนศึกษาใบความรู้ที่ 2.1 เรื่องฟังก์ชันทางคณิตศาสตร์ และใบความรู้ที่ 2.2 เรื่องคำสั่งสำหรับรับข้อมูลเข้า
- 4.2.3 ผู้เรียนทำใบงานที่ 2.1 เรื่องฟังก์ชันทางคณิตศาสตร์ จากนั้นผู้สอนสุ่มผู้เรียนออกมานำเสนอ
- 4.2.4 ผู้เรียนทำใบงานที่ 2.2 เรื่องแก้สมการกำลังสอง จากนั้นผู้สอนสุ่มผู้เรียนออกมานำเสนอ
- 4.2.5 ผู้เรียนทำใบงานที่ 2.3 เรื่องคุณสมบัติของวงกลม จากนั้นผู้สอนสุ่มผู้เรียนออกมานำเสนอ
- 4.2.6 ผู้เรียนและผู้สอนร่วมกันอภิปรายและสรุปเกี่ยวกับการแปลงสูตรทางคณิตศาสตร์ให้เป็นนิพจน์ของซีชาร์ป และการเขียนคำสั่งรับข้อมูล การคำนวณ การแสดงผลข้อมูล

5. การวัดและประเมินผล

5.1 ตรวจสอบคำตอบจากใบงาน

6. แหล่งความรู้เพิ่มเติม

- 6.1 [http://msdn.microsoft.com/th-th/vcsharp/default\(en-us\).aspx](http://msdn.microsoft.com/th-th/vcsharp/default(en-us).aspx)

7. ข้อเสนอแนะ

- 7.1 ผู้สอนอาจฝึกทักษะของผู้เรียนโดยเพิ่มเติมโจทย์ที่ใช้สูตรต่างๆ ในการคำนวณ



ใบงานที่ 2.1

ฟังก์ชันทางคณิตศาสตร์

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 2.1 และใบความรู้ที่ 2.2 แล้วตอบคำถามต่อไปนี้

- ให้เขียนโปรแกรมสั้น ๆ เพื่อตรวจสอบค่าของแต่ละนิพจน์ในตารางด้านล่าง จากนั้นบันทึกผลลัพธ์ลงในคอลัมน์ด้านขวามือ รวมทั้งระบุค่าที่คำนวณเป็นรูปแบบการคำนวณทางคณิตศาสตร์ ดังตัวอย่าง

นิพจน์	ค่าที่ถูกรคำนวณ	ผลลัพธ์
<code>Math.Abs(-1)</code>	$ -1 $	1
<code>Math.Sqrt(5)</code>		
<code>Math.Abs(9.5)</code>		
<code>Math.Pow(5,2)</code>		
<code>Math.Pow(2,-1)</code>		
<code>Math.Pow(5,0.5)</code>		
<code>Math.Pow(Math.Sqrt(2),8)</code>		
<code>Math.Log(10)</code>		
<code>Math.Sin(Math.PI/6)</code>		
<code>Math.Log10(100)</code>		

- แปลงสูตรคณิตศาสตร์ในช่องด้านซ้ายให้เป็นนิพจน์ของ C# ที่สอดคล้องกัน และตอบในช่องด้านขวา

สูตรคณิตศาสตร์	นิพจน์ในภาษา C#
$x^y + z$	<code>Math.Pow(x,y) + z</code>
$\cos 2\pi + \ln x$	
$ x + y $	
$\sqrt{x^2 + y^2 + z^2}$	
$\sin^2 x + \cos^2 x$	
$\sqrt[5]{a+b}$	
$e^{x \ln y}$	



3. ดัดแปลงโปรแกรมในตัวอย่างที่ 2.2 เพื่ออ่านเวกเตอร์แบบสามมิติ (x,y,z) จากผู้ใช้แทนที่จะเป็นเพียงเวกเตอร์สองมิติ โดยมีผลการทำงานตามตัวอย่าง

```
Enter x: 1.5  
Enter y: 7  
Enter z: 12.25  
Size of the vector (1.5,7,12.25) is 14.19.
```

จากนั้นกรอกโปรแกรมลงในช่องว่าง



ใบงานที่ 2.2

แก้สมการกำลังสอง

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

สมการกำลังสอง (Quadratic equation) เป็นสมการที่สามารถจัดให้อยู่ในรูป

$$ax^2 + bx + c = 0$$

โดยรากของสมการสามารถคำนวณได้จากสูตรดังนี้

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

จงเขียนโปรแกรมเพื่อแก้สมการกำลังสอง โดยโปรแกรมจะรับค่าสัมประสิทธิ์ a , b และ c จากผู้ใช้ และคำนวณหาราก x โดยใช้สูตรข้างต้น ทั้งนี้ให้รายงานค่าของ x ทั้งสองค่าเสมอ แม้ว่าทั้งคู่จะมีค่าเท่ากันก็ตาม

```
Enter a: 2
Enter b: 3
Enter c: -20
x = 2.5, -4
```

คัดลอกโปรแกรมลงในช่องว่าง

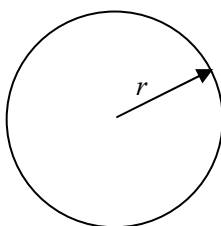
ใบงานที่ 2.3

คุณสมบัติของวงกลม

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

เป็นที่ทราบกันดีว่าวงกลมที่มีรัศมี r จะมีพื้นที่เท่ากับ πr^2 และเส้นรอบวงมีความยาวเท่ากับ $2\pi r$



จงเขียนโปรแกรมเพื่ออ่านค่าความยาวเส้นรอบวงของวงกลมวงหนึ่ง จากนั้นคำนวณและแสดงรัศมีและพื้นที่ของวงกลมนี้ออกทางหน้าจอโดยให้มีทศนิยมสองตำแหน่ง

Enter the circumference: 25.13274
The radius of the circle is 4.00
The area of the circle is 50.27

คัดลอกโปรแกรมลงในช่องว่าง



ใบความรู้ที่ 2.1

ฟังก์ชันทางคณิตศาสตร์

นอกเหนือจากตัวดำเนินการทางคณิตศาสตร์ (+ - * / ฯลฯ) ภาษา C# ได้เตรียมคลาส *Math* ที่รวบรวมเมทอด¹ และค่าคงที่ทางคณิตศาสตร์ที่สำคัญเอาไว้ใช้งาน ตารางด้านล่างแสดงตัวอย่างของฟังก์ชัน และค่าคงที่ที่เราสามารถเรียกใช้งานได้ผ่านคลาส *Math* คลาสนี้ถูกจัดไว้ภายใต้เนมสเปส *System* เช่นเดียวกันกับคลาส *Console*

เมทอด	ค่าที่คำนวณ
<i>Math.Abs(x)</i>	ค่าสัมบูรณ์ของ x ($ x $)
<i>Math.Ceiling(x)</i>	จำนวนเต็มที่น้อยที่สุดที่มากกว่าหรือเท่ากับ x
<i>Math.Floor(x)</i>	จำนวนเต็มที่มากที่สุดที่น้อยกว่าหรือเท่ากับ x
<i>Math.Log(x)</i>	ลอการิทึมฐานธรรมชาติของ x
<i>Math.Log10(x)</i>	ลอการิทึมฐานสิบของ x
<i>Math.Round(x)</i>	จำนวนเต็มที่ใกล้กับค่า x มากที่สุด
<i>Math.Pow(x,y)</i>	x ยกกำลัง y (x^y)
<i>Math.Sqrt(x)</i>	รากที่สองของ x (\sqrt{x})
<i>Math.Max(x)</i>	ค่าที่มากกว่าระหว่าง x และ y
<i>Math.Min(x)</i>	ค่าที่น้อยกว่าระหว่าง x และ y
<i>Math.Sin(x)</i>	ค่าไซน์ของ x (x มีหน่วยเป็นเรเดียน)}
<i>Math.Cos(x)</i>	ค่าโคไซน์ของ x (x มีหน่วยเป็นเรเดียน)}

ค่าคงที่	ความหมาย
<i>Math.PI</i>	ค่าคงที่ π ซึ่งมีค่าประมาณ 3.14159265358979323846
<i>Math.E</i>	ค่าคงที่ e ซึ่งมีค่าประมาณ 2.7182818284590452354

¹ เมทอดในภาษาเชิงวัตถุหมายถึงฟังก์ชันหรือโปรแกรมย่อยที่ถูกติดอยู่กับคลาสหรือวัตถุหนึ่ง ๆ



ใบความรู้ที่ 2.2

คำสั่งสำหรับรับข้อมูลจากผู้ใช้

โปรแกรมที่ใช้งานได้โดยทั่ว ๆ ไป นอกจากจะต้องมีการแสดงผลลัพธ์ออกทางหน้าจอแล้วยังควรต้องมีการรับข้อมูลจากผู้ใช้เข้ามาประมวลผลได้ด้วย ภาษา C# ได้เตรียมวิธีการติดต่อกับผู้ใช้ได้หลายวิธี แต่วิธีหลักที่เราจะใช้ในการอบรมนี้จะเป็นการเรียกใช้เมทอด `Console.ReadLine` การเรียกใช้งานเมทอดนี้จะอยู่ในรูปของนิพจน์ที่ให้ค่าเป็นข้อความ (สตริง) ดังนั้นจึงสามารถนำค่าไปใส่ให้กับตัวแปรแบบสตริงหรือนำไปใช้ผสมกับนิพจน์อื่น ๆ ที่เกี่ยวข้องกับสตริงได้

ตัวอย่างที่ 2.1 โปรแกรมด้านล่างจะถามชื่อจากผู้ใช้ และกล่าวคำทักทายตามชื่อนั้น ๆ

```
using System;
class SayHello
{
    static void Main()
    {
        string name;
        Console.Write("What is your name? ");
        name = Console.ReadLine();
        Console.WriteLine("Hello {0}, how are you?", name);
        Console.ReadLine();
    }
}
```

นอกจากการรับข้อมูลแบบข้อความแล้วโปรแกรมส่วนใหญ่ยังต้องการรับข้อมูลที่เป็นตัวเลขเพื่อนำมาคำนวณอีกด้วย อย่างไรก็ตาม ภาษา C# ไม่มีคำสั่งที่รับข้อมูลชนิดตัวเลขโดยตรง แต่ได้เตรียมเมทอด `Parse` สำหรับแต่ละชนิดข้อมูลแบบตัวเลขเพื่อแปลงข้อมูลชนิดข้อความให้เป็นชนิดตัวเลขได้ การใช้งานเมทอด `Parse` เป็นดังนี้

```
<numeric_datatype>.Parse(<string_expression>)
```

การใช้งานเมทอด `Parse` จะอยู่ในรูปของนิพจน์ที่ให้ชนิดข้อมูลตามที่ระบุไว้ใน `<numeric_datatype>` โดยมีค่าสอดคล้องกับค่าที่ระบุในนิพจน์ `<string_expression>` ดังนั้นค่าจากเมทอด `Parse` อาจนำไปกำหนดให้ตัวแปรโดยตรงหรือนำไปผสมกับนิพจน์อื่น ๆ เพื่อสร้างเป็นนิพจน์ใหม่ก็ได้



ตัวอย่างที่ 2.2 เขียนโปรแกรมเพื่ออ่านเวกเตอร์แบบสองมิติ (x,y) จากผู้ใช้แล้วคำนวณขนาดของเวกเตอร์ และแสดงผลพร้อมออกทางหน้าจอโดยมีทศนิยมสองตำแหน่ง (ใช้ตัวกำหนดรูปแบบ {2:f2} แทนที่จะเป็น {2} เพียงอย่างเดียว)

```
using System;
class Vector2D
{
    static void Main()
    {
        double x, y, size;
        Console.Write("Enter x: ");
        x = double.Parse(Console.ReadLine());
        Console.Write("Enter y: ");
        y = double.Parse(Console.ReadLine());
        size = Math.Sqrt(x*x + y*y);
        Console.WriteLine("Size of the vector ({0},{1}) is {2:f2}.",
            x, y, size);
        Console.ReadLine();
    }
}
```


กิจกรรมที่ 3

คำสั่งแบบมีเงื่อนไข

1. จุดประสงค์ ให้ผู้เรียนสามารถ

- 1.1 เขียนโปรแกรมคำสั่งแบบมีเงื่อนไขโดยใช้นิพจน์ทางตรรกศาสตร์
- 1.2 เขียนโปรแกรมโครงสร้าง if และ if...else
- 1.3 เขียนโปรแกรมโครงสร้าง if หลายชั้น
- 1.4 เขียนโปรแกรมโครงสร้าง switch...case

2. แนวคิด

โปรแกรมส่วนใหญ่มักประกอบไปด้วยชุดคำสั่งที่ถูกเรียกทำงานก็ต่อเมื่อเงื่อนไข (condition) หนึ่ง ๆ เป็นจริงหรือเท็จเท่านั้น ในภาษา C# มีโครงสร้างคำสั่งที่รองรับการกำหนดเงื่อนไขให้กับการทำงานของโปรแกรม ซึ่งได้แก่โครงสร้าง **if** โครงสร้าง **if...else** และโครงสร้าง **switch...case** การกำหนดค่าจริงเท็จให้กับเงื่อนไขที่ใช้ในโครงสร้างเหล่านี้อาศัยนิพจน์ทางตรรกศาสตร์ (boolean expressions) ซึ่งเป็นนิพจน์ที่ถูกตีความเป็นค่าความจริงและให้ค่าที่เป็นไปได้เพียงสองค่าคือ **true** (จริง) และ **false** (เท็จ) นิพจน์ทางตรรกศาสตร์มีบทบาทอย่างมากในการกำหนดเงื่อนไขให้กับคำสั่งแบบมีเงื่อนไข

โครงสร้าง **if** เป็นโครงสร้างที่ใช้ควบคุมการทำงานของคำสั่งอื่น ๆ ภายใต้อำนาจเงื่อนไข (condition) ที่กำหนด การใช้งานนั้นมีสองรูปแบบคร่าว ๆ ได้แก่ โครงสร้าง **if** ที่คำสั่ง *statement* จะถูกเรียกทำงานก็ต่อเมื่อนิพจน์ทางตรรกศาสตร์ที่กำหนดเป็น *condition* มีค่าเป็นจริง และ โครงสร้าง **if...else** ที่คำสั่ง *statement1* จะถูกเรียกทำงานเมื่อนิพจน์ในตำแหน่ง *condition* มีค่าเป็นจริง หากนิพจน์ดังกล่าวมีค่าเป็นเท็จ คำสั่ง *statement2* จะถูกเรียกทำงานแทน

ในบางโปรแกรมเราจำเป็นต้องเขียนโครงสร้าง **if** หลายชั้นโดยใช้ **if** (หรือ **if...else**) ซ้อนไว้ภายใต้อำนาจโครงสร้างของคำสั่ง **if** อีกอันหนึ่ง โดยเฉพาะอย่างยิ่งโปรแกรมที่ต้องตรวจสอบเงื่อนไขมากกว่าสองเงื่อนไข

ภาษา C# ได้เตรียมโครงสร้าง **switch...case** เพื่อใช้ในการจัดการเงื่อนไขหลายเงื่อนไข โดยเฉพาะ โดยทำได้เหมือนโครงสร้าง **if** แบบหลายชั้น โดยการตรวจสอบค่าของ *expression* และโปรแกรมจะกระโดดไปทำงาน ณ คำสั่ง **case** ที่ระบุค่าของ *constant-expression* ไว้ตรงกับค่าของ *expression* ที่ประเมินได้ คำสั่งต่าง ๆ ที่อยู่ถัดจากคำสั่ง **case** นั้น ๆ จะถูกเรียกใช้งานตามลำดับไปเรื่อย ๆ จนกว่าโปรแกรมจะพบคำสั่ง **break** ซึ่งมีผลทำให้โปรแกรมหยุดการทำงาน



3. สื่ออุปกรณ์

3.1 ใบงาน

ใบงานที่	เรื่อง	เวลา (นาที)
3.1	การตรวจสอบค่าความจริง	10
3.2	ระบบจตุภาค	15
3.3	ดัชนีมวลกาย	15
3.4	สั่งหนังสือ	30
3.5	เครื่องคิดเลขรุ่นใหม่	30

3.2 ใบความรู้

- ใบความรู้ที่ 3.1 เรื่องนิพจน์ทางตรรกศาสตร์
- ใบความรู้ที่ 3.2 เรื่องโครงสร้าง if และ if...else
- ใบความรู้ที่ 3.3 เรื่องโครงสร้าง if หลายชั้น
- ใบความรู้ที่ 3.4 เรื่องโครงสร้าง switch...case

3.3 อื่นๆ

- เครื่องคอมพิวเตอร์เท่ากับจำนวนกลุ่ม
- โปรแกรมวิชวลซีชาร์ป เอ็กเพรส

4. วิธีดำเนินการ

4.1 การจัดเตรียม

- 4.1.1 แบ่งผู้เรียนเป็นกลุ่ม กลุ่มละ 2 คน
- 4.1.2 เตรียมใบงานที่ 3.1 - 3.5 ตามจำนวนกลุ่มและใบความรู้ที่ 3.1 – 3.3 ตามจำนวนผู้เรียน

4.2 ขั้นตอนการดำเนินการ

- 4.2.1 แบ่งผู้เรียนเป็นกลุ่ม กลุ่มละ 2 คน
- 4.2.2 ผู้สอนชักชวนผู้เรียนพูดคุยถึงการแก้ปัญหาที่ผู้เรียนคุ้นเคยเพื่อโยงเข้าสู่การคิดตัดสินใจ เชื่อมโยงไปยังคำสั่งในคอมพิวเตอร์ โดยผู้สอนใช้นิพจน์ทางตรรกศาสตร์ เป็นแนวทางอภิปรายเพื่อช่วยให้ผู้เรียนเกิดข้อสรุปของการทำกิจกรรม
- 4.2.3 ผู้เรียนศึกษาใบความรู้ที่ 3.1 เรื่องนิพจน์ทางตรรกศาสตร์ จากนั้นให้ผู้เรียนแต่ละกลุ่มช่วยกันทำใบงานที่ 3.1 เรื่องการตรวจสอบค่าความจริง ซึ่งให้อธิบายเงื่อนไขที่จะทำให้ นิพจน์ทางตรรกศาสตร์ในแต่ละข้อมีค่าเป็นจริง

- 4.2.4 ผู้เรียนศึกษาไปความรู้ที่ 3.2 เรื่องโครงสร้าง if และ if...else และทำไปงานที่ 3.2 เรื่องระบบจตุภาค โดยให้เติมนิพจน์ทางตรรกศาสตร์ในคำสั่งเงื่อนไขเพื่อให้โปรแกรมทำงานได้ตามที่กำหนด
- 4.2.5 ผู้เรียนและผู้สอนร่วมกันเฉลยไปงานที่ 3.1 และ 3.2 พร้อมตรวจให้คะแนน
- 4.2.6 ผู้เรียนแต่ละกลุ่มศึกษาไปความรู้ที่ 3.3 เรื่องโครงสร้าง if หลายชั้น จากนั้นให้ทำไปงานที่ 3.3 เรื่องดัชนีมัลกาย ซึ่งให้เติมนิพจน์ทางตรรกศาสตร์ในคำสั่งเงื่อนไขเพื่อให้โปรแกรมคำนวณดัชนีมัลกาย ทำงานได้ตามที่กำหนด และไปงานที่ 3.4 เรื่องส่งหนังสือตามลำดับ โดยในระหว่างการเขียนโปรแกรมในไปงานให้ผู้สอนคอยให้คำแนะนำปรึกษา หลังจากนั้นให้ผู้สอนตรวจสอบโปรแกรมที่ได้จากทั้ง 2 ไปงาน ทางหน้าจอคอมพิวเตอร์ของแต่ละกลุ่ม พร้อมให้คะแนน และคำแนะนำ
- 4.2.7 ผู้เรียนแต่ละกลุ่มศึกษาไปความรู้ที่ 3.4 เรื่องโครงสร้าง switch...case และช่วยกันทำไปงานที่ 3.5 เรื่องเครื่องคิดเลขรุ่นใหม่ โดยให้ฝึกเขียนโปรแกรมโดยแก้ไขโปรแกรมเครื่องคิดเลขจากตัวอย่างที่ 3.6 เพื่อให้โปรแกรมรองรับตัวดำเนินการเพิ่มอีกสามตัวคือ *, / ,และ ^ ผู้สอนตรวจสอบการฝึกเขียนโปรแกรมจากหน้าจอคอมพิวเตอร์ของแต่ละกลุ่ม พร้อมให้คะแนน และคำแนะนำ
- 4.2.8 ผู้สอนสุ่มกลุ่มผู้เรียนมานำเสนอวิธีคิด ออกแบบโปรแกรม พร้อมอภิปรายถึงข้อดีข้อเสียของผลที่ได้

5. การวัดและประเมินผล

5.1 ตรวจคำตอบจากไปงาน

6. แหล่งความรู้เพิ่มเติม

6.1 [http://msdn.microsoft.com/th-th/vcsharp/default\(en-us\).aspx](http://msdn.microsoft.com/th-th/vcsharp/default(en-us).aspx)

7. ข้อเสนอแนะ

-



ใบงานที่ 3.1

การตรวจสอบค่าความจริง

รายชื่อสมาชิกในกลุ่มที่.....

- | | |
|---------|---------|
| 1. | 2. |
| 3. | 4. |

ให้ผู้เรียนศึกษาใบความรู้ที่ 3.1 แล้วตอบคำถามต่อไปนี้

กำหนดให้ x, y และ z เป็นตัวแปรชนิด **int**,

c เป็นชนิด **char**

และ s เป็นชนิด **string**

จงอธิบายเงื่อนไขที่จะทำให้นิพจน์ทางตรรกศาสตร์ในแต่ละบรรทัดมีค่าเป็นจริง

นิพจน์	เงื่อนไขที่ทำให้นิพจน์นี้เป็นจริง
$x > 2$	x มีค่ามากกว่า 2
$x \% 2 == 0$	x เป็นจำนวนคู่
$(x \% 5 == 0)$	
$(x \% y == 0)$	
$((x \% y == 0) \ \&\& \ (z \% y == 0))$	
$ch == 'a'$	
$((c \geq 'a') \ \&\& \ (c \leq 'z'))$	
$((c \geq 'A') \ \&\& \ (c \leq 'Z'))$	
$((c \geq '0') \ \&\& \ (c \leq '9'))$	c เป็นอักขระระหว่าง '0' ถึง '9'
$(s != \text{"Hello"})$	
$!(s != \text{"Arthur"})$	

ใบงานที่ 3.2

ระบบจตุภาค

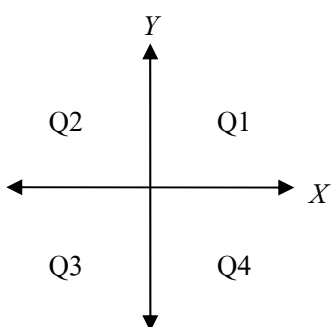
รายชื่อสมาชิกในกลุ่มที่.....

1. 2.

3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 3.2 และเขียนโปรแกรมจากโจทย์ที่กำหนดให้ต่อไปนี้

พิจารณาระบบจตุภาค (quadrant) ในระนาบสองมิติดังรูป



โปรแกรม (ที่ยังไม่สมบูรณ์) ด้านล่างจะรับข้อมูลเป็นตัวเลขจำนวนจริงสองค่าเพื่อระบุพิกัด (x,y) และรายงานว่าพิกัดนี้ตกอยู่ในจตุภาค (quadrant) ใด หากพิกัดที่ป้อนเข้ามาตกอยู่บนแกน x หรือแกน y โปรแกรมจะแสดงข้อความ I don't know.

```
using System;
class Quadrant {
    static void Main() {
        Console.Write("Enter X: ");
        int x = int.Parse(Console.ReadLine());
        Console.Write("Enter Y: ");
        int y = int.Parse(Console.ReadLine());

        if (__(a)__)
            Console.WriteLine("{0},{1} is in Q1.", x, y);
        if (__(b)__)
            Console.WriteLine("{0},{1} is in Q2.", x, y);
        if (__(c)__)
            Console.WriteLine("{0},{1} is in Q3.", x, y);
        if (__(d)__)
            Console.WriteLine("{0},{1} is in Q4.", x, y);
        if (__(e)__)
            Console.WriteLine("I don't know.");
    }
}
```



ตัวอย่างผลการทำงาน

```
Please input X: -50
Please input Y: 10
(-50, 10) is in Q2.
```

```
Please input X: 0
Please input Y: 50
I don't know.
```

จงเติมนิพจน์ทางตรรกศาสตร์ลงในช่องว่างที่เว้นไว้เพื่อให้โปรแกรมทำงานได้อย่างถูกต้อง

ช่องว่าง	นิพจน์ทางตรรกศาสตร์
__(a)__	
__(b)__	
__(c)__	
__(d)__	
__(e)__	

ใบงานที่ 3.3

ดัชนีมวลกาย

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 3.3 แล้วเขียนโปรแกรมจากโจทย์ที่กำหนดให้ต่อไปนี้

ดัชนีมวลกาย (BMI : Body mass index) เป็นค่าดัชนีที่คำนวณความสมดุลของน้ำหนักและส่วนสูงเพื่อนำมาเป็นตัวชี้วัดระดับความอ้วนผอมของมนุษย์ โดยคำนวณจากการนำน้ำหนัก (กิโลกรัม) มาหารด้วยกำลังสองของส่วนสูง (เมตร)

โปรแกรมด้านล่างจะสอบถามน้ำหนักและส่วนสูงจากผู้ใช้และรายงานผู้ใช้ให้ทราบถึงสุขภาพด้านน้ำหนักของตนเองตามตารางต่อไปนี้

ค่าดัชนีมวลกาย	การประเมินค่า
น้อยกว่า 18.5	น้ำหนักต่ำกว่าเกณฑ์ (Underweight)
ตั้งแต่ 18.5 แต่น้อยกว่า 25	ปกติ (Normal)
ตั้งแต่ 25 แต่น้อยกว่า 30	น้ำหนักมากกว่าเกณฑ์ (Overweight)
ตั้งแต่ 30 ขึ้นไป	โรคอ้วน (Obese)

เติมคำตอบที่ถูกต้องลงในช่องว่างที่เว้นไว้ เพื่อให้โปรแกรมทำงานได้อย่างสมบูรณ์

```
using System;
class BMICalc {
    static void Main() {
        Console.WriteLine("Enter your weight (in kg): ");
        double w = double.Parse(Console.ReadLine());
        Console.WriteLine("Enter your height (in m): ");
        double h = double.Parse(Console.ReadLine());
        double bmi = w/(h*h);
        Console.WriteLine("Your BMI is {0:f2}.", bmi);
        if (__(a)__)
            Console.WriteLine("You are underweight.");
        else if (__(b)__)
            Console.WriteLine("You are normal.");
        else if (__(c)__)
            Console.WriteLine("You are overweight.");
        else
            Console.WriteLine("You are obese.");
    }
}
```



ตัวอย่างผลการทำงาน

Enter your weight: 65
Enter your height: 1.75
Your BMI is 21.22.
You are normal.

Enter your weight: 100
Enter your height: 1.60
Your BMI is 39.06.
You are obese.

ช่องว่าง	คำตอบ
___(a)___	
___(b)___	
___(c)___	

ใบงานที่ 3.4

สั่งหนังสือ

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

จงเขียนโปรแกรมเพื่อแก้ปัญหาโจทย์ต่อไปนี้

ท่านได้รับมอบหมายจากร้านหนังสือออนไลน์แห่งหนึ่งเพื่อเขียนโปรแกรมคำนวณค่าส่งหนังสือไปยังลูกค้า โดยค่าส่งนั้นขึ้นอยู่กับน้ำหนักหนังสือและชนิดของบริการที่ลูกค้าเลือกตามตารางข้างล่าง

ประเภทบริการ	น้ำหนัก	อัตราค่าส่ง (บาท/กรัม)
ธรรมดา (Regular)	2000 กรัมแรก	0.25
ธรรมดา (Regular)	ส่วนที่เกิน 2000 กรัม	0.35
ด่วน (Express)	ใช้อัตราเดียวกับบริการแบบธรรมดา แต่เพิ่มค่าธรรมเนียมอีก 50 บาท	

สมมติว่าลูกค้าสั่งหนังสือที่มีน้ำหนัก 4.5 กก. และเลือกบริการการส่งแบบธรรมดาเราสามารถคำนวณค่าส่งได้ดังนี้

$$\begin{aligned}\text{ค่าส่ง} &= (2000 \text{ กรัม} \times 0.25 \text{ บาท/กรัม}) + (2500 \text{ กรัม} \times 0.35 \text{ บาท/กรัม}) \\ &= 500 \text{ บาท} + 875 \text{ บาท} = 1375 \text{ บาท}\end{aligned}$$

ตัวอย่างผลการทำงาน

Choose service (R-Regular, X-Express): R
Enter the package's weight (kilograms): 4.5
Your shipping cost is 1375.00 baht.

Choose service (R-Regular, X-Express): Y
Invalid service!!



จากนั้นคัดลอกโปรแกรมลงในช่องว่าง



ใบงานที่ 3.5

เครื่องคิดเลขรุ่นใหม่

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 3.4 แล้วเขียนโปรแกรมจากโจทย์ที่กำหนดให้ต่อไปนี้

แก้ไขโปรแกรมเครื่องคิดเลขจากตัวอย่างที่ 3.6 เพื่อให้โปรแกรมรองรับตัวดำเนินการเพิ่มอีกสามตัวคือ *, /, และ ^ ซึ่งคำนวณการคูณ ($x \times y$) การหาร ($x \div y$) และยกกำลัง (x^y) ตามลำดับ ดังตัวอย่างผลลัพธ์

```
Enter the first number: 5.2  
Enter the second number: 3  
Enter the operator: ^  
5.2^3 = 140.608
```

```
Enter the first number: 1  
Enter the second number: 2  
Enter the operator: @  
Invalid operator!
```

จากนั้นคัดลอกโปรแกรมลงในช่องว่าง



ใบความรู้ที่ 3.1

นิพจน์ทางตรรกศาสตร์

นิพจน์ทางตรรกศาสตร์ (boolean expressions) เป็นนิพจน์ที่ถูกตีความเป็นค่าความจริง ซึ่งให้ค่าที่เป็นไปได้เพียงสองค่าคือ **true** (จริง) และ **false** (เท็จ) นิพจน์ทางตรรกศาสตร์มีบทบาทอย่างมากในการกำหนดเงื่อนไขให้กับคำสั่งแบบมีเงื่อนไข ซึ่งภาษา C# อนุญาตให้เราสร้างนิพจน์เหล่านี้ได้จากการผสมนิพจน์ทางคณิตศาสตร์ (หรือแม้แต่นิพจน์แบบอักขระและนิพจน์แบบข้อความ) เข้าด้วยกันโดยอาศัยตัวดำเนินการเปรียบเทียบดังนี้

การเปรียบเทียบ	สัญลักษณ์ใน C#	ตัวอย่าง	ชนิดข้อมูลที่ใช้ได้	ความหมาย
=	==	x == y	ตัวเลขทุกชนิด, char, string	x เท่ากับ y
≠	!=	x != y	ตัวเลขทุกชนิด, char, string	x ไม่เท่ากับ y
<	<	x < y	ตัวเลขทุกชนิด, char	x น้อยกว่า y
≤	<=	x <= y	ตัวเลขทุกชนิด, char	x น้อยกว่าหรือเท่ากับ y
>	>	x > y	ตัวเลขทุกชนิด, char	x มากกว่า y
≥	>=	x >= y	ตัวเลขทุกชนิด, char	x มากกว่าหรือเท่ากับ y

นอกจากนี้เรายังสามารถนำเอานิพจน์ทางตรรกศาสตร์ตั้งแต่หนึ่งนิพจน์หรือมากกว่ามาผสมกันเพื่อสร้างนิพจน์ทางตรรกศาสตร์ที่ซับซ้อนขึ้นอีกโดยอาศัยตัวเชื่อมดังต่อไปนี้

- **&&** เชื่อมนิพจน์ทางตรรกศาสตร์สองนิพจน์เข้าด้วยกันโดยใช้ตรรกแบบ "และ" (AND) ตัวอย่างเช่น (x>1) && (x<10) จะให้ค่าจริงเมื่อตัวแปร x มีค่าอยู่ระหว่าง 1 ถึง 10
- **||** เชื่อมนิพจน์ทางตรรกศาสตร์สองนิพจน์เข้าด้วยกันโดยใช้ตรรกแบบ "หรือ" (OR) ตัวอย่างเช่น (x<1) || (x>10) จะให้ค่าจริงเมื่อตัวแปร x มีค่าน้อยกว่า 1 หรือมากกว่า 10
- **!** กลับค่าความจริงของนิพจน์ทางตรรกศาสตร์ ตัวอย่างเช่น !(x==1) จะเป็นจริงเมื่อตัวแปร x มีค่าไม่เท่ากับ 1

ใบความรู้ที่ 3.2

โครงสร้าง if และ if...else

โครงสร้าง **if** เป็นโครงสร้างที่ใช้ควบคุมการทำงานของคำสั่งอื่น ๆ ภายใต้เงื่อนไข (condition) ที่กำหนด การใช้งานนั้นมีสองรูปแบบคร่าว ๆ ได้แก่

- **รูปแบบที่ 1: โครงสร้าง if**

จากการใช้งานด้านล่าง คำสั่ง *statement* จะถูกเรียกทำงานก็ต่อเมื่อเงื่อนไขทางตรรกศาสตร์ที่กำหนดเป็น *condition* มีค่าเป็นจริง

```
if (condition)
    statement; // executed if the condition is true
```

เนื่องจากโครงสร้างข้างต้นอนุญาตให้เรากำหนดเงื่อนไขให้กับคำสั่งเพียงคำสั่งเดียวเท่านั้น อย่างไรก็ตาม หากมีคำสั่งมากกว่าหนึ่งภายใต้เงื่อนไขเดียวกัน คำสั่งเหล่านี้สามารถถูกจัดกลุ่มให้เป็นเสมือนคำสั่งเดียวได้โดยการครอบคำสั่งทั้งหมดด้วยวงเล็บปีกกา ({...})

```
if (condition) {
    statement1; // executed if the condition is true
    statement2; // executed if the condition is true
    statement3; // executed if the condition is true
    :
}
```

- **รูปแบบที่ 2: โครงสร้าง if...else**

คำสั่ง *statement1* จะถูกเรียกทำงานเมื่อเงื่อนไขในตำแหน่ง *condition* มีค่าเป็นจริง หากนิพจน์ดังกล่าวมีค่าเป็นเท็จ คำสั่ง *statement2* จะถูกเรียกทำงานแทน

```
if (condition)
    statement1; //executed if the condition is true
else
    statement2; //executed if the condition is false
```

และเช่นเคย เราสามารถใช้งานโครงสร้าง **if...else** ร่วมกับวงเล็บปีกกาหากมีคำสั่งที่ต้องการให้ทำงานภายใต้เงื่อนไขมากกว่าหนึ่ง

```
if (condition) {
    statementT1; //executed if the condition is true
    statementT2; //executed if the condition is true
}
else {
    statementF1; //executed if the condition is false
    statementF2; //executed if the condition is false
}
```



ตัวอย่างที่ 3.1 รหัสจำลอง (pseudo-code) ด้านล่างอธิบายขั้นตอนวิธีสำหรับให้โปรแกรมพิมพ์คำว่า Passed หากคะแนนของนักเรียนมีค่ามากกว่าหรือเท่ากับ 60 ไม่เช่นนั้นให้พิมพ์คำว่า Failed

```
if student's score is greater than or equal to 60
    Print "Passed"
otherwise
    Print "Failed"
```

เราสามารถนำรหัสจำลองข้างต้นมาเขียนเป็นโปรแกรมภาษา C# ได้ดังนี้ (แสดงเพียงส่วนสำคัญเท่านั้น)

```
if (score >= 60)
    Console.WriteLine("Passed");
else
    Console.WriteLine("Failed");
```

เนื่องจากคีย์เวิร์ด **else** เป็นตัวกำหนดให้การพิมพ์คำว่า Failed ทำงานเมื่อเงื่อนไข `score >= 60` เป็นเท็จ ดังนั้นหากเราแทนที่ **else** ด้วยคำสั่ง **if** และใช้เงื่อนไขที่ตรงข้ามกันคือ `score < 60`

โปรแกรมก็จะมีการทำงานเหมือนกับโปรแกรมข้างบนทุกประการ

```
if (score >= 60)
    Console.WriteLine("Passed");
if (score < 60)
    Console.WriteLine("Failed");
```

ตัวอย่างที่ 3.2 โปรแกรมต่อไปนี้จะรอรับตัวเลขจากผู้ใช้และให้คำตอบว่าตัวเลขนั้น ๆ เป็นเลขคู่ (even) หรือ เลขคี่ (odd)

- ใช้รูปแบบ **if**

```
using System;
class OddOrEven {
    static void Main() {
        int N;
        Console.Write("Please input N: ");
        N = int.Parse(Console.ReadLine());
        if (N%2 == 0)
            Console.WriteLine("{0} is even", N); //true
        if (N%2 != 0)
            Console.WriteLine("{0} is odd", N); //true
    }
}
```

- ใช้รูปแบบ **if...else**

```
using System;
class OddOrEven {
    static void Main() {
        int N;
        Console.Write("Please input N: ");
        N = int.Parse(Console.ReadLine());
        if (N%2 == 0)
            Console.WriteLine("{0} is even", N); //true
        else
            Console.WriteLine("{0} is odd", N); //false
    }
}
```

ตัวอย่างที่ 3.3 บริษัทโทรศัพท์มือถือแห่งหนึ่งเสนอโปรโมชั่นให้กับลูกค้าโดยมีการคำนวณค่าธรรมเนียมการใช้งานดังนี้

- สองนาทีแรก คิดนาทีละห้าบาท
- นาทีถัดมาคิดนาทีละสองบาท

โปรแกรมด้านล่างจะรับค่าจำนวนนาที่จากผู้ใช้ และคำนวณค่าธรรมเนียมการใช้งาน นอกจากนี้ภายในโปรแกรมยังมีคอมเมนต์กำกับเอาไว้หลายจุดเพื่ออธิบายการทำงานของโปรแกรมในส่วนต่าง ๆ

```
using System;
class Cellphone {
    static void Main() {
        // Step 1: Take the number of minutes input
        Console.Write("Enter the number of minutes: ");
        int minutes = int.Parse(Console.ReadLine());

        // Step 2: Split the number of minutes into two parts,
        // the first two and the remaining
        int first, remaining;
        if (minutes > 2) {
            // Step 2.1: If the call takes more than two minutes,
            // then the first two minutes is used entirely and the
            // remaining is minutes subtracted by two
            first = 2;
            remaining = minutes - 2;
        }
        else {
            // Step 2.2: If the call takes less than 2 minutes,
            // these minutes are considered part of the first two
            first = minutes;
            remaining = 0;
        }

        // Step 3: Compute the fee based on the number of minutes
        // during the first two minutes, and the number of minutes
        // after the first two minutes
        int fee = (first*5) + (remaining*2);
        Console.WriteLine("The air time fee is {0} baht.", fee);
    }
}
```



ตัวอย่างผลการทำงาน

Enter the number of minutes: 1
The air time fee is 5 baht.

Enter the number of minutes: 5
The air time fee is 16 baht.



ใบความรู้ที่ 3.3

โครงสร้าง if หลายชั้น

ในบางโปรแกรม เราจำเป็นต้องเขียนโครงสร้าง **if** (หรือ **if...else**) ซ้อนไว้ภายใต้โครงสร้างของคำสั่ง **if** อีกอันหนึ่ง โดยเฉพาะอย่างยิ่งโปรแกรมที่ต้องตรวจสอบเงื่อนไขมากกว่าสองเงื่อนไข ซึ่งมีรูปแบบการใช้งานโดยทั่วไปดังนี้

```
if (condition1)
    statement1;
else if (condition2)
    statement2;
else if (condition3)
    statement3;
:
else
    statementN;
```

จากรูปแบบด้านบน *statement1* จะถูกเรียกทำงานเมื่อเงื่อนไข *condition1* เป็นจริง ลองพิจารณา *statement2* จะเห็นว่ามันถูกควบคุมด้วยเงื่อนไข *condition2* และยังอยู่ภายใต้ **else** ของโครงสร้าง **if** อันบนสุด จึงทำให้ *statement2* นี้ถูกเรียกทำงานเมื่อเงื่อนไข *condition2* เป็นจริงและเงื่อนไข *condition1* เป็นเท็จเท่านั้น ในทำนองเดียวกัน คำสั่ง *condition3* จะถูกเรียกทำงานเมื่อเงื่อนไข *condition1* และ *condition2* ทั้งคู่เป็นเท็จ และเงื่อนไข *condition3* เป็นจริงเท่านั้น และสุดท้ายคือ *statementN* ซึ่งจะถูกเรียกทำงานเมื่อเงื่อนไขทั้งหมดข้างต้นเป็นเท็จ

ตัวอย่างที่ 3.4 พิจารณากระบวนการคัดเกรดนักเรียนโดยพิจารณาจากคะแนนสอบไล่ที่ได้ตามตาราง

เงื่อนไข	ระดับคะแนน
ได้คะแนนน้อยกว่า 50	F
ได้คะแนนตั้งแต่ 50 แต่น้อยกว่า 60	D
ได้คะแนนตั้งแต่ 60 แต่น้อยกว่า 70	C
ได้คะแนนตั้งแต่ 70 แต่น้อยกว่า 80	B
ได้คะแนนตั้งแต่ 80 ขึ้นไป	A



เราสามารถเขียนโปรแกรมเพื่อทำงานดังกล่าวโดยใช้โครงสร้าง **if** หลายชั้นดังนี้

```
if (point < 50)
    Console.WriteLine("Grade F");
else if (point < 60)
    Console.WriteLine("Grade D");
else if (point < 70)
    Console.WriteLine("Grade C");
else if (point < 80)
    Console.WriteLine("Grade B");
else
    Console.WriteLine("Grade A");
```

ใบความรู้ที่ 3.4

โครงสร้าง switch...case

ถึงแม้ว่าการจัดการเงื่อนไขหลาย ๆ เงื่อนไขในคราวเดียวกันจะสามารถทำได้โดยอาศัยโครงสร้าง **if** แบบหลายชั้น ภาษา C# ยังได้เตรียมโครงสร้าง **switch...case** เพื่อใช้ในการจัดการเงื่อนไขหลายเงื่อนไขโดยเฉพาะ การใช้งานโครงสร้าง **switch...case** อยู่ในรูปแบบดังนี้

```
switch (expression)
{
    case constant-expression-1:
        statements;
        break;
    case constant-expression-2:
        statements;
        break;
    case constant-expression-3:
        statements;
        break;
    :
    default:
        statements;
        break;
}
```

ภาษา C# ยอมให้นิพจน์ที่ใช้ในตำแหน่ง *expression* เป็นนิพจน์แบบจำนวนเต็ม (integer) แบบอักขระ (char) หรือแบบข้อความ (string) เท่านั้น หลังจากทีค่าของ *expression* ถูกตรวจสอบ โปรแกรมจะกระโดดไปทำงาน ณ คำสั่ง **case** ที่ระบุค่าของ *constant-expression* ไว้ตรงกับค่าของ *expression* ที่ประเมินได้ คำสั่งต่าง ๆ ที่อยู่ถัดจากคำสั่ง **case** นั้น ๆ จะถูกเรียกใช้งานตามลำดับไปเรื่อย ๆ จนกว่าโปรแกรมจะพบคำสั่ง **break** ซึ่งมีผลทำให้โปรแกรมหยุดการทำงานภายในโครงสร้าง **switch...case** นั้นและกระโดดไปยังคำสั่งที่ต่อไปนอกโครงสร้าง หากไม่พบ *constant-expression* ใดที่มีค่าตรงกับ *expression* โปรแกรมจะกระโดดไปยังจุดที่มีการระบุด้วยคำสั่ง **default**



ตัวอย่างที่ 3.5 โปรแกรมต่อไปนี้แสดงแต้มคะแนนตามระดับคะแนน (A,B,C,D,F) ที่ป้อนโดยผู้ใช้

ระดับคะแนน (grade)	แต้มระดับคะแนน (grade point)
A	4.0
B	3.0
C	2.0
D	1.0
F	0.0

```
using System;
class GradePoint {
    static void Main() {
        string grade;
        Console.Write("Please input your grade: ");
        grade = Console.ReadLine();
        switch (grade) {
            case "A" : Console.WriteLine("Your point is 4.0."); break;
            case "a" : Console.WriteLine("Your point is 4.0."); break;
            case "B" : Console.WriteLine("Your point is 3.0."); break;
            case "b" : Console.WriteLine("Your point is 3.0."); break;
            case "C" : Console.WriteLine("Your point is 2.0."); break;
            case "c" : Console.WriteLine("Your point is 2.0."); break;
            case "D" : Console.WriteLine("Your point is 1.0."); break;
            case "d" : Console.WriteLine("Your point is 1.0."); break;
            case "F" : Console.WriteLine("Your point is 0.0."); break;
            case "f" : Console.WriteLine("Your point is 0.0."); break;
            default: Console.WriteLine("Invalid input!!"); break;
        }
    }
}
```

ตัวอย่างผลการทำงาน

Please input your grade: A
Your point is 4.0.

Please input your grade: B
Your point is 3.0.

Please input your grade: e
Invalid input!!

แม้ว่าโปรแกรมข้างต้นจะทำงานได้อย่างถูกต้อง โปรแกรมดังกล่าวยังค่อนข้างยาวอีกทั้งโปรแกรมยังมีคำสั่งที่ถูกใช้ซ้ำ ๆ กันอยู่หลายแห่งเนื่องจากการป้อนระดับคะแนนด้วยตัวอักษรตัวใหญ่และตัวเล็กจะให้ผลเหมือนกัน ภาษา C# อนุญาตให้คำสั่ง **case** หลาย ๆ คำสั่งควบคุมชุดคำสั่งร่วมกันได้ดังตัวอย่าง

```
using System;
class GradePoint {
    static void Main() {
        string grade;
        Console.Write("Please input your grade: ");
        grade = Console.ReadLine();
        switch (grade) {
            case "A" :
            case "a" : Console.WriteLine("Your point is 4.0."); break;
            case "B" :
            case "b" : Console.WriteLine("Your point is 3.0."); break;
            case "C" :
            case "c" : Console.WriteLine("Your point is 2.0."); break;
            case "D" :
            case "d" : Console.WriteLine("Your point is 1.0."); break;
            case "F" :
            case "f" : Console.WriteLine("Your point is 0.0."); break;
            default: Console.WriteLine("Invalid input!!"); break;
        }
    }
}
```

โปรแกรมนี้อย่างยังสามารถทำให้สั้นลงได้อีกโดยการใช้ตัวแปรเสริมอีกตัวเพื่อเก็บแต้มระดับคะแนน แล้วจึงนำค่าของตัวแปรพิมพ์ออกทางหน้าจอโดยใช้คำสั่ง *Console.WriteLine* ในคราวเดียวก่อนจบโปรแกรม ดังแสดง

```
using System;
class GradePoint {
    static void Main() {
        string grade;
        double point = -1;
        Console.Write("Please input your grade: ");
        grade = Console.ReadLine();
        switch (grade) {
            case "A" : case "a" : point = 4.0; break;
            case "B" : case "b" : point = 3.0; break;
            case "C" : case "c" : point = 2.0; break;
            case "D" : case "d" : point = 1.0; break;
            case "F" : case "f" : point = 0.0; break;
            default: Console.WriteLine("Invalid input!!"); break;
        }
        if (point >= 0)
            Console.WriteLine("Your point is {0:f1}.", point);
    }
}
```



ตัวอย่างที่ 3.6 โปรแกรมต่อไปนี้เป็นโปรแกรมเครื่องคิดเลขอย่างง่าย เมื่อเริ่มทำงานโปรแกรมจะให้ผู้ใช้ป้อนค่าตัวเลขจำนวนจริงสองค่า พร้อมระบุตัวดำเนินการทางคณิตศาสตร์ที่ต้องการโดยเป็นได้เพียง + หรือ - จากนั้นโปรแกรมจะแสดงผลลัพธ์จากการคำนวณ หากผู้ใช้ป้อนตัวดำเนินการอื่นนอกเหนือจาก + หรือ - โปรแกรมจะรายงานความผิดพลาด

```

1:  using System;
2:  class Calculator {
3:      static void Main() {
4:          double n1, n2, ans = 0;
5:          char op;
6:          Console.Write("Enter the first number: ");
7:          n1 = double.Parse(Console.ReadLine());
8:          Console.Write("Enter the second number: ");
9:          n2 = double.Parse(Console.ReadLine());
10:         Console.Write("Enter the operator: ");
11:         op = char.Parse(Console.ReadLine());
12:         switch(op) {
13:             case '+': ans = n1+n2; break;
14:             case '-': ans = n1-n2; break;
15:             default: op = ' '; break;
16:         }
17:         if (op == ' ')
18:             Console.WriteLine("Invalid operator!");
19:         else
20:             Console.WriteLine("{0}{1}{2} = {3}", n1, op, n2, ans);
21:     }
22: }

```

สังเกตบรรทัดที่ 15 ซึ่งใช้จัดการกรณีที่ผู้ใช้ป้อนตัวดำเนินการอื่นนอกเหนือจาก + หรือ - บรรทัดนี้จะเปลี่ยนค่าของตัวแปร `op` ให้เป็นช่องว่าง (' ') เพื่อนำไปเช็คที่ท้ายโปรแกรมอีกทีหนึ่ง

ตัวอย่างผลการทำงาน

```

Enter the first number: 8
Enter the second number: 10
Enter the operator: +
8+10 = 18

```

กิจกรรมที่ 4

คำสั่งวนซ้ำ

1. จุดประสงค์ ให้ผู้เรียนสามารถ

- 1.1 อธิบายโครงสร้างของการเขียนโปรแกรมแบบวนซ้ำ
- 1.2 เขียนโปรแกรมคำสั่งวนซ้ำโครงสร้างแบบ while
- 1.3 เขียนโปรแกรมคำสั่งวนซ้ำโครงสร้างแบบ do..while
- 1.4 เขียนโปรแกรมคำสั่งวนซ้ำโครงสร้างแบบ for

2. แนวคิด

การเขียนโปรแกรมเพื่อให้คอมพิวเตอร์ประมวลผลข้อมูลแบบเดียวกันหลาย ๆ รอบ โดยเขียนคำสั่งซ้ำ ๆ กันในโปรแกรมทำให้โปรแกรมมีขนาดใหญ่ ขาดความยืดหยุ่นและผิดพลาดได้ง่าย จึงจำเป็นต้องอาศัยการเขียนโปรแกรมที่มีโครงสร้างแบบวนซ้ำ ซึ่งจะช่วยให้การเขียนคำสั่งสั้นลง คำสั่งในการเขียนโปรแกรมแบบวนซ้ำมีหลายคำสั่ง แต่ละคำสั่งจะมีความเหมาะสมกับลักษณะงานหรือสถานการณ์ที่แตกต่างกัน

โครงสร้าง **while** ถือเป็นโครงสร้างแบบง่ายที่สุดที่สามารถนำมาใช้เขียนโปรแกรมเพื่อวนทำคำสั่ง) หรือกลุ่มของคำสั่ง (ซ้ำหลาย ๆ รอบ

โครงสร้างแบบ **do..while** เป็นอีกโครงสร้างหนึ่งที่สามารถนำมาใช้เขียนโปรแกรมเพื่อทำงานวนซ้ำได้ และมีการทำงานคล้ายคลึงกับโครงสร้างแบบ **while** มาก สิ่งที่แตกต่างกันก็คือ **do..while** จะตรวจสอบเงื่อนไขหลังจากทำคำสั่งภายในลูปไปแล้วหนึ่งครั้ง และจะวนซ้ำไปเรื่อย ๆ จนกระทั่งเงื่อนไขที่ระบุมีค่าเป็นเท็จ

เนื่องจากลูปแบบวนซ้ำมีการใช้งานบ่อยครั้งในโปรแกรมทั่ว ๆ ไป ภาษา C# (รวมถึงภาษาโปรแกรมอื่น ๆ ด้วย) จึงได้เตรียมโครงสร้างพิเศษเพื่อใช้จัดการลูปประเภทนี้ได้โดยสะดวกยิ่งขึ้น โครงสร้างนี้คือโครงสร้าง **for** โดยเงื่อนไขของลูปจะถูกตรวจสอบก่อนที่จะคำสั่งวนซ้ำคำสั่งแรกจะถูกเรียกใช้



3. สื่ออุปกรณ์

3.1 ใบงาน

ใบงานที่	เรื่อง	เวลา (นาที)
4.1	ผลรวมตัวเลข	20
4.2	สถิติคะแนนสอบ	30
4.3	ลำดับตัวเลข	20
4.4	จำนวนเฉพาะ	20
4.5	กลุ่มดาวสามเหลี่ยม	20

3.2 ใบความรู้

- ใบความรู้ที่ 4.1 เรื่องโครงสร้าง While ลูป
- ใบความรู้ที่ 4.2 เรื่องโครงสร้าง do...while ลูป
- ใบความรู้ที่ 4.3 เรื่องโครงสร้าง for ลูป

3.3 อื่นๆ

- เครื่องคอมพิวเตอร์เท่ากับจำนวนกลุ่ม
- โปรแกรมวิชวลซีชาร์ป เอ็กเพรส

4. วิธีดำเนินการ

4.1 การจัดเตรียม

- 4.1.1 แบ่งผู้เรียนเป็นกลุ่ม กลุ่มละ 2 คน
- 4.1.2 เตรียมใบงานที่ 4.1 - 4.5 ตามจำนวนกลุ่มและใบความรู้ที่ 4.1 – 4.3 ตามจำนวนผู้เรียน

4.2 ขั้นตอนการดำเนินการ

- 4.2.1 ผู้สอนกล่าวถึงการเขียนโปรแกรมโครงสร้าง While, do...while และ for
- 4.2.2 ผู้เรียนแต่ละคนศึกษาใบความรู้ที่ 4.1 เรื่องโครงสร้าง While ลูป และทำใบงานที่ 4.1 เรื่องผลรวมตัวเลข
- 4.2.3 ผู้สอนสุ่มผู้เรียนออกมานำเสนอคำตอบในใบงานที่ 4.1
- 4.2.4 ผู้เรียนแต่ละกลุ่มศึกษาใบความรู้ที่ 4.2 เรื่องโครงสร้าง do...while ลูป และทำใบงานที่ 4.2 เรื่องสถิติคะแนนสอบ จากนั้นผู้สอนสุ่มกลุ่มผู้เรียนออกมานำเสนอ
- 4.2.5 ผู้เรียนแต่ละคนทำศึกษาใบความรู้ที่ 4.3 เรื่องโครงสร้าง for ลูป และทำใบงานที่ 4.3 เรื่องลำดับตัวเลข จากนั้นให้ผู้เรียนตรวจสอบคำตอบกับเพื่อนในกลุ่ม
- 4.2.6 ผู้เรียนและผู้สอนร่วมกันอภิปราย ซักถามเกี่ยวกับคำสั่งวนซ้ำแต่ละแบบว่าเหมาะสมกับลักษณะงาน หรือสถานการณ์ใดบ้าง เพื่อการเลือกใช้คำสั่งที่เหมาะสมกับงาน



- 4.2.7 ผู้เรียนแต่ละกลุ่มทำใบงานที่ 4.4 เรื่องจำนวนเฉพาะ จากนั้นผู้สอนสุ่มกลุ่มผู้เรียนออกมานำเสนอ
- 4.2.8 ผู้เรียนแต่ละกลุ่มทำใบงานที่ 4.5 เรื่องกลุ่มดาวสามเหลี่ยม จากนั้นผู้สอนสุ่มผู้เรียนออกมานำเสนอ
- 4.2.9 ผู้เรียนและผู้สอนร่วมกันอภิปรายและสรุปเกี่ยวกับโครงสร้าง While, do...while และ for

5. การวัดและประเมินผล

- 5.1 ตรวจคำตอบจากใบงาน

6. แหล่งความรู้เพิ่มเติม

- 6.1 [http://msdn.microsoft.com/th-th/vcsharp/default\(en-us\).aspx](http://msdn.microsoft.com/th-th/vcsharp/default(en-us).aspx)

7. ข้อเสนอแนะ

- 7.1 ผู้สอนอาจหาใบงาน หรือ โจทย์สถานการณ์ต่างๆ เพิ่มเติม เพื่อให้ผู้เรียนทดลองเขียนโปรแกรมโดยใช้คำสั่งวนซ้ำ



ใบงานที่ 4.1

ผลรวมตัวเลข

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 4.1 แล้วเขียนโปรแกรมจากโจทย์ที่กำหนดให้ต่อไปนี้

โปรแกรมด้านล่างคัดแปลงมาจากโปรแกรมในตัวอย่างที่ 4.3 โดยนอกจากจะรับตัวเลขจากผู้ใช้ไปเรื่อย ๆ แล้ว โปรแกรมยังคำนวณผลรวมของตัวเลขทั้งหมด (ยกเว้นเลขติดลบตัวสุดท้ายที่ผู้ใช้ ป้อนเพื่อระบุจุดสิ้นสุดของข้อมูล) จงทำโปรแกรมให้สมบูรณ์โดยเติมคำสั่งที่เหมาะสมลงในช่องว่าง

```
using System;
class While5 {
    static void Main() {
        int N = 0, sum;
        ____(a)____;
        while (N >= 0) { //Exit while loop when N is negative
            Console.Write("Please input N: ");
            N = int.Parse(Console.ReadLine());
            _____(b)_____;
        }
        _____(c)_____;
        Console.WriteLine("Bye Bye!!!");
    }
}
```

ตัวอย่างผลการทำงาน

```
Please input N: 3
Please input N: 2
Please input N: 599
Please input N: 0
Please input N: -5
Sum = 604
Bye Bye!!!
```

ช่องว่าง	คำตอบ
____(a)____	
____(b)____	
____(c)____	

ใบงานที่ 4.2

สถิติคะแนนสอบ

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 4.1 และใบความรู้ที่ 4.2 แล้วเขียนโปรแกรมจากโจทย์ที่กำหนดให้ต่อไปนี้

1. ประมวลผลสถิติเบื้องต้น

งานของท่านคือเขียนโปรแกรมเพื่อช่วยคุณครูผู้สอนภาษา C# คำนวณคะแนนเฉลี่ยของนักเรียนในห้อง โปรแกรมของท่านจะอ่านคะแนนของนักเรียนทีละรายจนกว่าผู้ใช้จะป้อนค่า -1 เพื่อบ่งบอกจุดสิ้นสุดของข้อมูล จากนั้นให้รายงานจำนวนนักเรียน (นับเอาจากจำนวนคะแนน) และคะแนนเฉลี่ยด้วยทศนิยมสองตำแหน่ง

ตัวอย่างผลการทำงาน

```
Enter a score, or -1 to quit: 76  
Enter a score, or -1 to quit: 56.7  
Enter a score, or -1 to quit: 87.4  
Enter a score, or -1 to quit: 53.5  
Enter a score, or -1 to quit: 90.8  
Enter a score, or -1 to quit: 99  
Enter a score, or -1 to quit: -1  
Number of students is 6  
Average score is 77.23
```

คัดลอกโปรแกรมลงในช่องว่าง





2. สถิติขั้นสูง

แก้ไขโปรแกรมในขั้นตอนที่แล้วเพื่อให้โปรแกรมรายงานคะแนนสูงสุดและต่ำสุดเพิ่มเติมจากเดิมด้วยโดยใช้ทศนิยมสองตำแหน่งเช่นกัน

ตัวอย่างผลการทำงาน

```
Enter a score, or -1 to quit: 76
Enter a score, or -1 to quit: 56.7
Enter a score, or -1 to quit: 87.4
Enter a score, or -1 to quit: 53.5
Enter a score, or -1 to quit: 90.8
Enter a score, or -1 to quit: 99
Enter a score, or -1 to quit: -1
Number of students is 6
Average score is 77.23
Min score is 53.50
Max score is 99.00
```

คัดลอกโปรแกรมลงในช่องว่าง



ใบงานที่ 4.3

ลำดับตัวเลข

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 4.3 แล้วตอบคำถามต่อไปนี้

1. เติมเต็มโปรแกรมต่อไปนี้เพื่อให้โปรแกรมพิมพ์ตัวเลข 0, -1, -2, ..., -56 บนจอภาพ (แสดงตัวเลขบรรทัดละตัว)

```
class ForEx {
    static void Main() {
        int k;
        for ( __ (a) __ ; __ (b) __ ; __ (c) __ ) {
            ____ (d) ____;
        }
    }
}
```

เขียนส่วนที่เติมเต็มลงในตารางด้านล่าง

ช่องว่าง	คำตอบ
__ (a) __	
__ (b) __	
__ (c) __	
__ (d) __	

2. จากข้อ 1 โปรแกรมในตำแหน่ง (a),..., (d) ควรเปลี่ยนเป็นเช่นไรเพื่อให้โปรแกรมแสดงค่า 7, 14, 21,..., 70 บนจอภาพ

ช่องว่าง	คำตอบ
__ (a) __	
__ (b) __	
__ (c) __	
__ (d) __	

ใบงานที่ 4.4

จำนวนเฉพาะ

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

จากโจทย์ต่อไปนี้

จำนวนเฉพาะ (prime number หรือเรียกสั้น ๆ ว่า prime) เป็นจำนวนเต็มบวกซึ่งมีตัวหารเพียงสองตัวคือ 1 และตัวมันเอง ตัวอย่างเช่น 7 เป็นจำนวนเฉพาะเนื่องจากเลขที่หารมันลงตัวมีเพียง 1 และ 7 ส่วน 10 ไม่ใช่จำนวนเฉพาะเพราะมีตัวหารถึงสี่ตัวคือ 1, 2, 5 และ 10 จะเห็นว่าจากนิยามของจำนวนเฉพาะจะได้ว่า 1 ไม่ใช่จำนวนเฉพาะเนื่องจากมีตัวหารเพียงตัวเดียวคือ 1

เขียนโปรแกรมเพื่อรับจำนวนเต็มบวก N และตรวจสอบว่า N เป็นจำนวนเฉพาะหรือไม่

ตัวอย่างผลการทำงาน

Enter N: 10
10 is not a prime

Enter N: 19
19 is a prime

Enter N: 1
1 is not a prime





ใบงานที่ 4.5

กลุ่มดาวสามเหลี่ยม

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

เขียนโปรแกรมเพื่อวาดรูปสามเหลี่ยมด้วยเครื่องหมายดอกจัน (*) ที่มีขนาดตามที่ใช้กำหนด

ตัวอย่างผลการทำงาน

```
Enter triangle size: 2  
*  
**
```

```
Enter triangle size: 5  
*  
**  
***  
****  
*****
```

แนวทาง: ให้พิจารณาว่าสามเหลี่ยมแต่ละขนาดกินเนื้อที่กี่บรรทัดและแต่ละบรรทัดประกอบด้วยช่องว่างต้นบรรทัดกี่ช่อง และดาวกี่ดวง)



ใบความรู้ที่ 4.1

โครงสร้าง while ลูป

โครงสร้าง **while** ลูปเป็นโครงสร้างแบบง่ายที่สุดที่สามารถนำมาใช้เขียนโปรแกรมเพื่อวนทำคำสั่ง (หรือกลุ่มของคำสั่ง) ซ้ำหลาย ๆ รอบ รูปแบบการใช้งาน **while** ลูปเป็นดังนี้

```
while (condition)
    statement;
```

เช่นเดียวกับโครงสร้าง **if** เราสามารถกำหนดการวนซ้ำให้กับกลุ่มของคำสั่งได้โดยใช้วงเล็บปีกกา ({...})

```
while (condition) {
    statement1;
    statement2;
    :
    statementN;
}
```

ตัวอย่างที่ 4.1 จงเขียนโปรแกรมเพื่อพิมพ์ค่าตั้งแต่ 1 ถึง N โดยรับค่า N จากผู้ใช้

```
1: using System;
2: class OneToN {
3:     static void Main() {
4:         int i, N;
5:         Console.WriteLine("Enter N: ");
6:         N = int.Parse(Console.ReadLine());
7:         i = 1; // initialize variable i to 1
8:         while (i <= N) {
9:             Console.WriteLine(i);
10:            i++;
11:        }
12:    }
13: }
```

ตัวอย่างผลการทำงาน

```
Please input N: 5
1
2
3
4
5
```

ลองมาพิจารณาการทำงานของโปรแกรมทีละขั้นตอน เมื่อโปรแกรมเริ่มทำงาน ผู้ใช้จะป้อนตัวเลขเพื่อใช้เป็นค่าของ N ณ บรรทัดที่ 6 และโปรแกรมจะกำหนดให้ค่า 1 เป็นค่าเริ่มต้นของ i ที่บรรทัดถัดมา จากนั้นโครงสร้าง **while** จะมีผลทำให้โปรแกรมทำคำสั่งที่บรรทัดที่ 9 และ 10 จนกระทั่งนิพจน์ $i <=$

N มีค่าเป็นเท็จ ในการวนซ้ำแต่ละครั้งโปรแกรมจะพิมพ์ค่า i และเพิ่มค่า i ขึ้นทีละหนึ่ง ผลการทำงานของโปรแกรมจึงเป็นการพิมพ์ตัวเลขตั้งแต่ 1 (ซึ่งเป็นค่าเริ่มต้นของ i) จนถึงค่าของ N (ซึ่งเป็นค่าสุดท้ายของ i ที่ยังทำให้เงื่อนไข $i \leq N$ เป็นจริง)

ตัวอย่างที่ 4.2 เราสามารถเขียนโปรแกรมเพื่อคำนวณหาผลรวมตั้งแต่ 1 ถึง N ใด ๆ โดยแก้ไขโปรแกรมในตัวอย่างที่ 4.1 เพียงเล็กน้อยเท่านั้นดังต่อไปนี้

```
using System;
class Summing {
    static void Main() {
        int N, i, sum;
        Console.WriteLine("Please input N: ");
        N = int.Parse(Console.ReadLine());
        i = 1; sum = 0;
        while (i <= N) {
            sum = sum + i;
            i++;
        }
        Console.WriteLine("Sum from 1 to {0} = {1} ", N, sum);
    }
}
```

ตัวอย่างผลการทำงาน

```
Please input N: 4
1
2
3
4
Sum from 1 to 4 = 10
```

ในตัวอย่างที่ผ่านมามีการกำหนดเงื่อนไขการทำงานของลูปโดยการกำหนดตัวแปรตัวหนึ่งขึ้นมาใช้เป็นตัวนับ (counter) ซึ่งมักจะถูกตั้งค่าเริ่มต้นให้เป็นศูนย์ก่อนเริ่มเข้าลูป และเพิ่มค่าขึ้นทีละหนึ่งหลังจากคำสั่งภายในลูปถูกทำงานเสร็จสิ้นหนึ่งรอบ โปรแกรมจะทำซ้ำไปเรื่อย ๆ จนกระทั่งตัวนับมีค่าเกินกว่าค่าที่กำหนด การวนลูปในลักษณะนี้เรียกว่า *ลูปวนนับ* (counting loop)

อย่างไรก็ตามลูปวนนับไม่สามารถนำไปใช้ในสถานการณ์ที่เราไม่สามารถคาดการณ์ได้ว่าลูปจะต้องถูกเรียกทำงานกี่รอบจึงจะจบการทำงาน อาทิเช่นการวนรับค่าจากผู้ใช้เรื่อย ๆ จนกว่าผู้ใช้จะป้อนค่าพิเศษค่าหนึ่ง ในกรณีนี้เราจะไม่อาศัยตัวแปรที่ทำหน้าที่เป็นตัวนับ แต่จะใช้วิธีกำหนดเงื่อนไขของลูปให้มีการเฝ้าดูค่าในตัวแปร (ซึ่งมักจะได้รับค่าจากผู้ใช้) ว่ามีค่าเท่ากับค่าพิเศษที่กำหนดให้สิ้นสุดการวนลูปหรือไม่ การวนลูปในลักษณะนี้เรียกว่า *ลูปเฝ้ายาม* (sentinel loop)

ตัวอย่างที่ 4.3 จงเขียนโปรแกรมเพื่อวนรับตัวเลขจากผู้ใช้จนกว่าผู้ใช้จะป้อนค่าติดลบจึงจบการทำงาน

```
using System;
class While4 {
    static void Main() {
        int N = 0;
        while (N >= 0) { // stop when N is negative
            Console.Write("Please input N: ");
            N = int.Parse(Console.ReadLine());
        }
        Console.WriteLine("Bye Bye!!!");
    }
}
```

ตัวอย่างผลการทำงาน

```
Please input N: 3
Please input N: 2
Please input N: 3000
Please input N: 9999
Please input N: -50
Bye Bye!!!
```

ใบความรู้ที่ 4.2

โครงสร้าง do...while ลูป

โครงสร้างแบบ **do..while** เป็นอีกโครงสร้างหนึ่งที่น่าสนใจเขียนโปรแกรมเพื่อทำงานวนซ้ำได้ และมีการทำงานคล้ายคลึงกับโครงสร้างแบบ **while** มาก สิ่งที่แตกต่างกันก็คือ **do..while** จะตรวจสอบเงื่อนไขหลังจากทำคำสั่งภายในลูปไปแล้วหนึ่งครั้ง และจะวนซ้ำไปเรื่อย ๆ จนกระทั่งเงื่อนไขที่ระบุมีค่าเป็นเท็จ ดังนั้นโครงสร้างแบบ **do..while** จึงทำคำสั่งในลูปอย่างน้อย หนึ่งครั้ง แม้ว่าเงื่อนไขจะเป็นเท็จตั้งแต่แรกก็ตาม รูปแบบการใช้งาน โครงสร้าง **do..while** เป็นดังนี้

```
do statement; while (condition);
```

และเช่นเคย วงเล็บปีกกาถูกนำมาใช้เพื่อรวมหลาย ๆ คำสั่งให้เสมือนเป็นหนึ่งเอาไว้ภายในลูปได้

```
do {  
    statement1;  
    statement2;  
    :  
    statementN;  
} while (condition);
```

ตัวอย่างที่ 4.4 โปรแกรมด้านล่างมีการใช้งานโครงสร้าง **do..while** แต่ให้ผลการทำงานเหมือนกัน

โปรแกรมในตัวอย่างที่ 4.3 ทุกประการ

```
using System;  
class DoWhile1 {  
    static void Main() {  
        int N = 0;  
        do {  
            Console.WriteLine("Please input N: ");  
            N = int.Parse(Console.ReadLine());  
        } while (N >= 0);  
        Console.WriteLine("Bye Bye!!!");  
    }  
}
```



ใบความรู้ที่ 4.3

โครงสร้าง for ลูป

ในใบความรู้ก่อนหน้านี้เราได้เห็นโครงสร้างของโปรแกรมแบบวนซ้ำที่ใช้คำสั่ง **while** และ **do..while** เมื่อพิจารณาการใช้คำสั่งเหล่านี้ในการเขียนลูปแบบวนซ้ำ (counting loop) เราจะพบว่าโครงสร้างของลูปมักจะมีส่วนประกอบเหล่านี้เสมอ

- ส่วนกำหนดค่าเริ่มต้นให้ตัวนับ – เป็นคำสั่งให้ค่าเริ่มต้นกับตัวแปรที่นำมาใช้เป็นตัวนับ คำสั่งนี้มักถูกเขียนไว้ทันทีก่อนที่จะโปรแกรมจะเข้าทำงานในโครงสร้าง **while** หรือ **do..while**
- ส่วนคำสั่งที่ถูกทำซ้ำ – ส่วนหลักของลูปที่ประกอบด้วยคำสั่งที่ถูกเรียกทำงานในแต่ละรอบการวนซ้ำ
- ส่วนเงื่อนไข – ใช้สำหรับกำหนดเงื่อนไขว่าคำสั่งในลูปจะถูกเรียกทำต่อหรือไม่
- ส่วนปรับค่าตัวนับ – มักเป็นเพียงคำสั่งสั้น ๆ เพื่อเพิ่มหรือลดค่าตัวแปรที่นำมาใช้เป็นตัวนับ

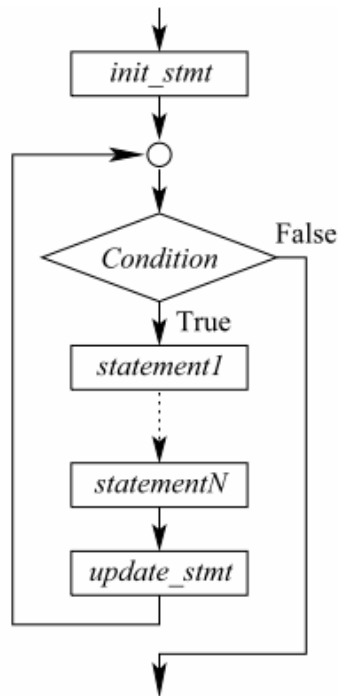
ตัวอย่างโปรแกรมต่อไปนี้แสดงให้เห็นส่วนประกอบต่าง ๆ ทั้งสี่ส่วน (แต่ละส่วนกำกับไว้ด้วยคอมเมนต์ท้ายบรรทัด)

```
using System;
class CountDown {
    static void Main() {
        int i;
        i = 10;                // (1) ส่วนกำหนดค่าเริ่มต้นให้ตัวนับ
        while (i >= 0) {        // (3) ส่วนเงื่อนไข
            Console.WriteLine(i); // (2) ส่วนคำสั่งที่ถูกทำซ้ำ
            i--;                // (4) ส่วนปรับค่าตัวนับ
        }
    }
}
```

เนื่องจากลูปแบบวนซ้ำมีการใช้งานบ่อยครั้งในโปรแกรมทั่ว ๆ ไป ภาษา C# (รวมถึงภาษาโปรแกรมอื่น ๆ ด้วย) จึงได้เตรียมโครงสร้างพิเศษเพื่อใช้จัดการลูปประเภทนี้ได้โดยสะดวกยิ่งขึ้น โครงสร้างนี้คือโครงสร้าง **for** ซึ่งมีรูปแบบการใช้งานดังนี้

```
for (init_stmt; condition; update_stmt) {
    statement1;
    statement2;
    :
    statementN;
}
```

ผังงานด้านล่างแสดงขั้นตอนการทำงานของ **for** ลูป สังเกตว่าเงื่อนไขของลูปจะถูกตรวจสอบก่อนที่คำสั่งวนซ้ำคำสั่งแรกจะถูกเรียกใช้ ดังนั้นโครงสร้าง **for** จึงมีการทำงานที่คล้ายคลึงกับโครงสร้าง **while** มากกว่าโครงสร้าง **do..while**



ตัวอย่างที่ 4.5 โปรแกรมต่อไปนี้นี้จะแสดงตัวเลข 1,2,3,...,20 บนหน้าจอ

```

using System;
class Counting {
    static void Main() {
        int i;
        for (i = 1; i <= 20; i++)
            Console.WriteLine(i);
    }
}
  
```

ตัวอย่างที่ 4.6 โปรแกรมต่อไปนี้รับตัวเลขอินพุต N จากผู้ใช้และแสดงตัวเลขทั้งหมดที่เป็นตัวประกอบของ N (นำไปหาร N แล้วลงตัว)

```
using System;
class Divisors {
    static void Main() {
        int i, N;

        Console.Write("Enter N: ");
        N = int.Parse(Console.ReadLine());
        for (i = 1; i <= N; i++) {
            if (N%i == 0) Console.WriteLine(i);
        }
    }
}
```

ตัวอย่างผลการทำงาน

```
Enter N: 100
1
2
4
5
10
20
25
50
100
```


กิจกรรมที่ 5

เมทอดเบื้องต้น

1. จุดประสงค์ ให้ผู้เรียนสามารถ

- 1.1 เขียนโปรแกรมโดยการประกาศและสร้างเมทอด
- 1.2 อธิบายวิธีการส่งค่าไปยังเมทอด
- 1.3 สร้างเมทอดแบบคืนค่า

2. แนวคิด

เมทอด (method) เป็นส่วนของโปรแกรมเพื่อจัดการงานย่อยหนึ่ง ๆ โดยการมองงานที่ซับซ้อนเป็นงานย่อย ๆ ที่เล็กลงทำให้เขียนโปรแกรมแก้ปัญหาได้ง่ายขึ้น ช่วยลดการเขียนโค้ดที่ซ้ำซ้อน เพิ่มความสะดวกในการดูแลและแก้ไขโปรแกรมในภายหลัง และลดความเสี่ยงในการเขียนโปรแกรมผิดพลาด

เราสามารถแบ่งชนิดของเมทอดเป็นสองประเภทคร่าว ๆ ได้แก่ **เมทอดแบบไม่คืนค่า (non-value returning methods)** และ **เมทอดแบบคืนค่า (value-returning methods)**

การส่งค่าไปยังเมทอดสามารถทำได้โดย สร้างเมทอดที่มีการรับค่าจากผู้เรียกเพื่อกำหนดพฤติกรรมการทำงานของเมทอดนั้น ๆ ค่าที่ถูกส่งไปนี้เรียกว่า **อาร์กิวเมนต์ (argument)** ส่วนเมทอดที่ถูกเรียกจะรับค่าเหล่านี้ผ่านมาทาง **พารามิเตอร์ (parameter)**

3. สื่ออุปกรณ์

3.1 ใบงาน

ใบงานที่	เรื่อง	เวลา (นาที)
5.1	แก้ไขเมทอดพิมพ์ดาว	20
5.2	ทดลองส่งค่าไปยังเมทอด	20
5.3	ทดลองวาดแผนภาพ	20
5.4	คำนวณพื้นที่วงกลม	20
5.5	อนุกรมฮาโมนิก	30
5.6	วาดกราฟของฟังก์ชันอย่างง่าย ๆ	30



3.2 ใบความรู้

- ใบความรู้ที่ 5.1 เรื่องการประกาศและเรียกใช้เมทอด
- ใบความรู้ที่ 5.2 เรื่องการส่งค่าไปยังเมทอด
- ใบความรู้ที่ 5.3 เรื่องเมทอดแบบคืนค่า

3.3 อื่นๆ

- เครื่องคอมพิวเตอร์เท่ากับจำนวนกลุ่ม
- โปรแกรมวิชวลชีชาร์ป เอ็กเพรส

4. วิธีดำเนินการ

4.1 การจัดเตรียม

- 4.1.1 แบ่งผู้เรียนเป็นกลุ่ม กลุ่มละ 2 คน
- 4.1.2 เตรียมใบงานที่ 5.1 – 5.6 ตามจำนวนกลุ่มและใบความรู้ที่ 5.1 – 5.3 ตามจำนวนผู้เรียน

4.2 ขั้นตอนการดำเนินการ

- 4.2.1 ผู้สอนกล่าวถึงการเขียนโปรแกรมด้วยการประกาศและเรียกใช้เมทอด
- 4.2.2 ผู้เรียนแต่ละกลุ่มศึกษาใบความรู้ที่ 5.1 เรื่องการประกาศและเรียกใช้เมทอด และทำใบงานที่ 5.1 เรื่องแก้ไขเมทอดพิมพ์ดาว
- 4.2.3 ผู้สอนสุ่มผู้เรียนออกมานำเสนอคำตอบในใบงานที่ 5.1
- 4.2.4 ผู้เรียนแต่ละกลุ่มศึกษาใบความรู้ที่ 5.2 เรื่องการส่งค่าไปยังเมทอด จากนั้นทำใบงานที่ 5.2 เรื่องทดลองส่งค่าไปยังเมทอด และใบงานที่ 5.3 เรื่องทดลองวาดแผนภาพ จากนั้นผู้สอนสุ่มกลุ่มผู้เรียนออกมานำเสนอ
- 4.2.5 ผู้เรียนแต่ละคนศึกษาใบความรู้ที่ 5.3 เรื่องเมทอดแบบคืนค่า แล้วทำใบงานที่ 5.4 เรื่องคำนวณพื้นที่วงกลม และใบงานที่ 5.5 เรื่องอนุกรมฮามอนิก จากนั้นให้ผู้เรียนตรวจสอบคำตอบกับเพื่อนในกลุ่ม
- 4.2.6 ผู้เรียนแต่ละกลุ่มทำใบงานที่ 5.6 เรื่องวาดกราฟของฟังก์ชันอย่างง่าย ๆ จากนั้นผู้สอนสุ่มกลุ่มผู้เรียนออกมานำเสนอ
- 4.2.7 ผู้เรียนและผู้สอนร่วมกันอภิปรายและสรุปเกี่ยวกับการสร้างและใช้งานเมทอดแบบต่างๆ

5. การวัดและประเมินผล

5.1 ตรวจคำตอบจากใบงาน

6. แหล่งความรู้เพิ่มเติม

- 6.1 [http://msdn.microsoft.com/th-th/vcsharp/default\(en-us\).aspx](http://msdn.microsoft.com/th-th/vcsharp/default(en-us).aspx)



7. ข้อเสนอแนะ

-



ใบงานที่ 5.1

แก้ไขเมทอดพิมพ์ดาว

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 5.1 จากนั้นสร้างโปรเจกต์และเขียนโปรแกรมตามโจทย์ที่กำหนดให้ต่อไปนี้

1. คัดแปลงโปรแกรมในตัวอย่างที่ 5.1 เพื่อให้มีการทำงานดังต่อไปนี้

1.1 ทดลองสร้างเมทอดอีกอันหนึ่งชื่อ *PrintLongerLine* ซึ่งมีการทำงานคล้ายกับเมทอด *PrintLine* แต่พิมพ์ดาวออกมา 20 ดวงแทนที่จะเป็น 10 จากนั้นเขียนเฉพาะการประกาศเมทอด *PrintLongerLine* ลงในช่องว่าง

1.2 หลังจากสร้างเมทอด *PrintLongerLine* สำเร็จแล้ว แก้ไขเมทอด *Main* เพื่อให้โปรแกรมพิมพ์ดาว 10 ดวงเป็นจำนวน 3 แถว และ 20 ดวงเป็นจำนวน 3 แถว ดังแสดง

```
*****  
*****  
*****  
*****  
*****  
*****
```

1.3 จากนั้นเขียนคำสั่งที่ใช้ภายในเมทอด *Main* ลงในช่องว่าง

ใบงานที่ 5.2

ทดลองส่งค่าไปยังเมท็อด

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.

3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 5.1 และใบความรู้ที่ 5.2 จากนั้นสร้างโปรเจกต์และเขียนโปรแกรมตามโจทย์ที่กำหนดให้ต่อไปนี้

1. ทดลองพิมพ์และรันโปรแกรมในตัวอย่างที่ 5.2

- ### 1.1 โปรแกรมแสดงผลอย่างไร

--

- 1.2 คัดแปลงเมทีอด *Main* เพื่อให้โปรแกรมแสดงอักขระ 'x' 10 ตัวในบรรทัดแรก อักขระ '*' 20 ตัวในบรรทัดถัดมา และอักขระ 'v' 30 ตัวในบรรทัดสุดท้าย จากนั้นคัดลอกเฉพาะคำสั่งที่ใช้ภายในเมทีอด *Main* ลงในช่องว่าง

ตัวอย่างผลการทำงาน

```
XXXXXXXXXXXXX  
*****  
VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV
```

--

ใบงานที่ 5.3

ทดลองวาดแผนภาพ

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 5.1 และใบความรู้ที่ 5.2 จากนั้นสร้างโปรเจกต์และเขียนโปรแกรมตามโจทย์ที่กำหนดให้ต่อไปนี้

เราจะทดลองสร้างเมทอดชื่อ *PlotChar* เพื่อนำไปใช้ในโจทย์ฝึกโปรแกรมในตอนท้าย เมทอดนี้รับพารามิเตอร์สองตัว คือ *c* เป็นชนิด **char** และ *dist* เป็นชนิด **int** ซึ่งคล้ายคลึงกับเมทอด *PrintCharLine* ที่ผ่านมา แต่เมทอด *PlotChar* จะพิมพ์ช่องว่างเป็นจำนวน *dist*-1 ตัวอักษร และพิมพ์อักขระในพารามิเตอร์ *c* ปิดท้ายเพียงตัวเดียวพร้อมทั้งขึ้นบรรทัดใหม่ ให้เติมคำสั่งลงในบรรทัดว่างที่เว้นไว้ในบรรทัดที่ 4--7 ของโปรแกรมต่อไปนี้ เพื่อให้โปรแกรมแสดงผลตามผลลัพธ์ที่แสดงไว้ด้านล่าง และห้ามแก้ไขส่วนของเมทอด *Main* โดยเด็ดขาด

```
1: using System;
2: class Stars {
3:     static void PlotChar(char c, int dist) {
4:         _____
5:         _____
6:         _____
7:         _____
8:     }
9:
10:    static void Main() {
11:        PlotChar('x', 1);
12:        PlotChar('-', 2);
13:        PlotChar('+', 3);
14:        PlotChar('o', 6);
15:    }
16: }
```

ตัวอย่างผลการทำงาน

```
x
-
+
o
```

ใบงานที่ 5.4

คำนวณพื้นที่วงกลม

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 5.3 จากนั้นสร้างโปรเจกต์และเขียนโปรแกรมตามโจทย์ที่กำหนดให้ต่อไปนี้

โปรแกรมต่อไปนี้ทำงานคล้ายกับโปรแกรมในตัวอย่างที่ 5.3 แต่โปรแกรมจะคำนวณพื้นที่ของวงกลมแทนที่จะเป็นสามเหลี่ยมโดยรับค่ารัศมีของวงกลมจากผู้ใช้ แต่โปรแกรมยังขาดส่วนสำคัญที่เว้นว่างเอาไว้คือการประกาศเมทอด *CircleArea* ที่จะคำนวณค่าพื้นที่จากรัศมี

```
using System;
class Circle {
    ____ (a) ____

    static void Main() {
        Console.Write("Enter radius: ");
        double radius = double.Parse(Console.ReadLine());
        Console.WriteLine("The area of the circle is {0:f2}.",
            CircleArea(radius));
    }
}
```

ตัวอย่างผลการทำงาน

```
Enter radius: 32
The area of the circle is 3216.99.
```

เขียนโปรแกรมเพิ่มเติมให้สมบูรณ์และคัดลอกส่วนของโปรแกรมที่เป็นการประกาศเมทอด *CircleArea* ในตำแหน่ง (a) ลงในช่องว่างด้านล่าง



ใบงานที่ 5.5

อนุกรมฮาร์โมนิค

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 5.3 จากนั้นสร้างโปรเจกต์และเขียนโปรแกรมตามโจทย์ที่กำหนดให้ต่อไปนี้

ให้คิดแปลงโปรแกรมในตัวอย่างที่ 5.4 ในส่วนของการประกาศเมทอด f เพื่อหาผลรวม n พจน์แรกของอนุกรมฮาร์โมนิค ดังนั้น $f(n)$ จึงมีนิยามดังนี้

$$f(n) = \sum_{i=1}^n \frac{1}{i}$$

จากนั้นลอกเมทอด f ลงในช่องว่าง

ตัวอย่างผลการทำงาน

n	f(n)
1	1.000
2	1.500
3	1.833
4	2.083
5	2.283
6	2.450
7	2.593
8	2.718
9	2.829
10	2.929
11	3.020
12	3.103
13	3.180
14	3.252
15	3.318



ใบงานที่ 5.6

วาดกราฟของฟังก์ชันอย่างง่าย ๆ

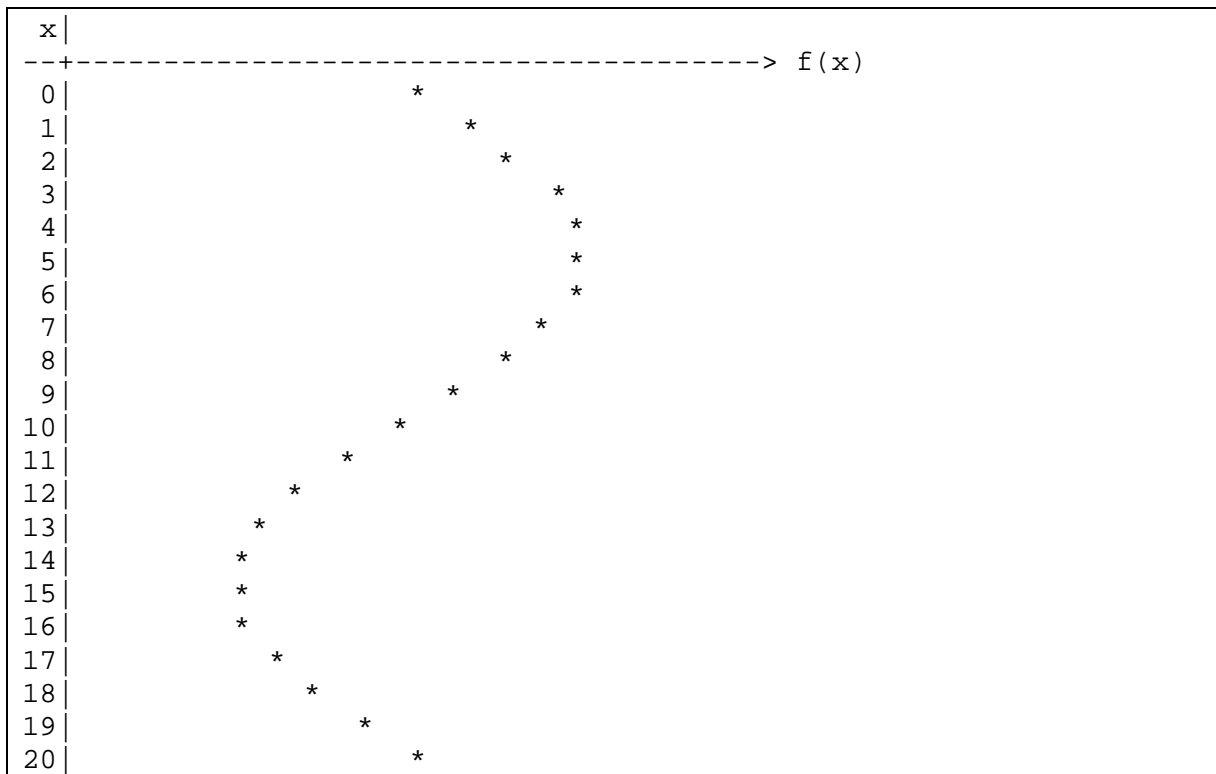
รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

จงเขียนโปรแกรมเพื่อวาดกราฟอย่างง่ายของฟังก์ชัน

$$f(x) = 20 + 10\sin\left(\frac{x}{\pi}\right)$$

โดยที่ x มีค่าตั้งแต่ 0 ถึง 20 และฟังก์ชัน $\sin(x)$ ให้ค่าชานซ์ของมุม x เรเดียน โปรแกรมที่สมบูรณ์ควรแสดงผลลัพธ์ดังนี้



ข้อมูลเพิ่มเติม

- เมทอด *Math.Sin* ในภาษา C# รับค่ามุมในหน่วยเรเดียนอยู่แล้ว
- เราสามารถปัดค่าทศนิยมให้เป็นจำนวนเต็มได้โดยใช้เมทอด *Math.Round* อย่างไรก็ตาม ค่าที่ได้จากเมทอดนี้จะยังคงเป็นชนิด **double** หากจะนำไปใช้ในรูปนิพจน์ชนิด **int** จะต้องมีการทำ type casting โดยการใส่ (**int**) ไว้ด้านหน้า เช่น

```
int a = (int) Math.Round(10.3293);
```

คัดลอกโปรแกรมลงในช่องว่าง

ใบความรู้ที่ 5.1

การประกาศและเรียกใช้เมทอด

ที่ผ่านมาเราได้เขียนโปรแกรมโดยใช้คำสั่งทั้งหมดไว้ในส่วนที่เรียกว่าเมทอด *Main* อย่างไรก็ตามการเขียนโปรแกรมในลักษณะนี้จะเหมาะสำหรับโปรแกรมสั้น ๆ ที่ใช้แก้ปัญหาง่าย ๆ เท่านั้น แต่สำหรับปัญหาที่มีขนาดใหญ่และมีความซับซ้อนมากขึ้นการเขียนโปรแกรมแบบนี้นอกจากจะทำให้โปรแกรมดูเอะอะไม่เป็นระเบียบแล้วยังทำให้เกิดความผิดพลาดภายในโปรแกรมได้ง่ายขึ้นอีกด้วย ดังนั้นนักเขียนโปรแกรมที่มีประสบการณ์จึงมักออกแบบโปรแกรมโดยการแบ่งปัญหาที่ต้องการแก้ออกเป็นงานย่อย ๆ หลาย ๆ งาน จากนั้นจึงเขียนโปรแกรมเป็นส่วน ๆ เพื่อจัดการงานย่อยในแต่ละงาน เราเรียกส่วนของโปรแกรมเพื่อจัดการงานย่อยหนึ่ง ๆ ว่า *ซับรูทีน* (subroutine) นอกจากนี้ยังมีคำอีกหลายคำที่ให้ความหมายในทำนองเดียวกัน เช่น *โปรแกรมย่อย* (subprogram) และ *ฟังก์ชัน* (function) สำหรับภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุเช่นภาษา C# นั้นส่วนของโปรแกรมหกกล่าวจะถูกเรียกว่า *เมทอด* (method)

การเขียนโปรแกรมเป็นส่วน ๆ โดยใช้เมทอดมีข้อดีหลายประการ อาทิเช่น:

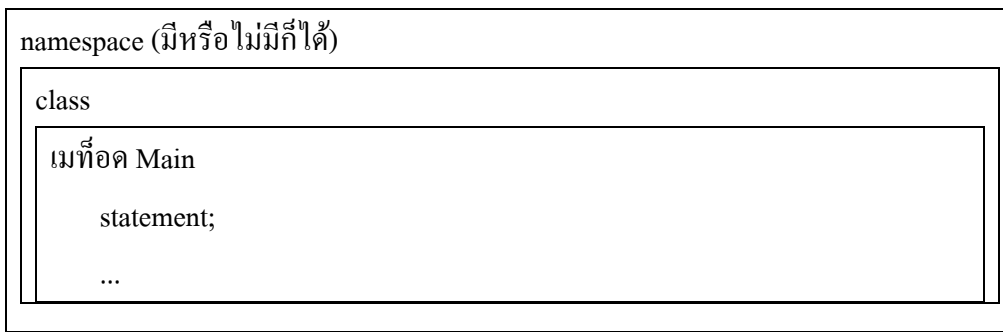
- เป็นการมองงานที่ซับซ้อนเป็นงานย่อย ๆ ที่เล็กลงและเขียนโปรแกรมแก้ปัญหาได้ง่ายขึ้น
- ช่วยลดการเขียนโค้ดที่ซ้ำซ้อน เช่นงานที่คล้ายคลึงกันที่ต้องถูกทำบ่อยครั้ง
- ช่วยซ่อนรายละเอียดของโปรแกรมไว้ที่ส่วนอื่น แทนที่จะใส่ทั้งหมดไว้ที่ Main
- ทำให้โปรแกรมดูมีระเบียบและง่ายต่อการเข้าใจ
- รวมการคำนวณทางคณิตศาสตร์ที่ซับซ้อนไว้เป็นฟังก์ชันใหม่ให้เรียกใช้ได้ง่าย
- เพิ่มความสะดวกในการดูแลและแก้ไขโปรแกรมในภายหลัง และลดความเสี่ยงในการเขียนโปรแกรมผิดพลาด
- เพิ่มความสะดวกในการนำโค้ดที่เขียนไว้แล้วไปใช้ในโปรแกรมอื่น ๆ

เราได้รู้จักวิธีการและได้เรียกใช้เมทอดมาแล้วก่อนหน้านี้หลายครั้งเช่นการใช้งานเมทอด *Console.WriteLine* และ *Math.Sqrt* เมทอดเหล่านี้เป็นเมทอดที่ถูกสร้างมาให้พร้อมกับตัวภาษา C# ซึ่งเราสามารถเรียกใช้งานได้ทันที แต่ในปฏิบัติการครั้งนี้เราจะรู้จักวิธีการสร้างเมทอดของเราขึ้นมาเอง

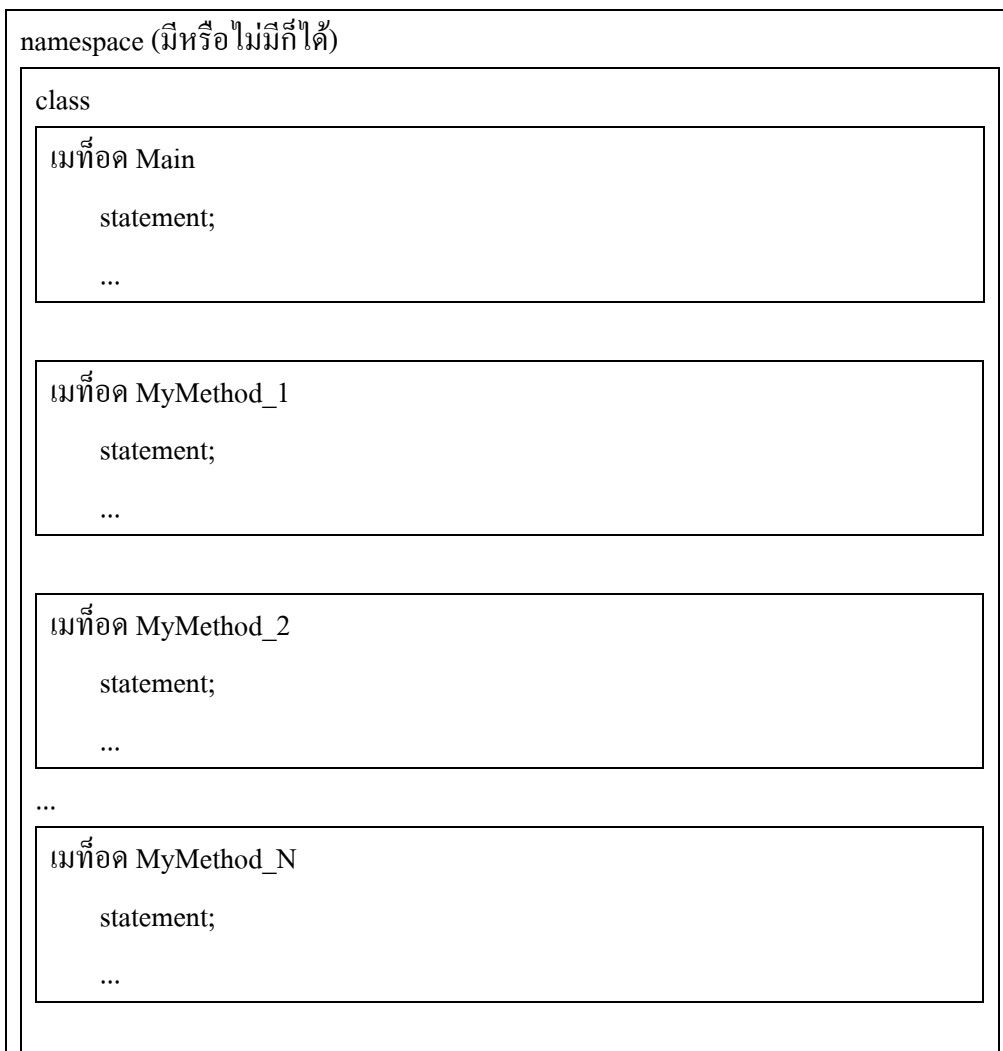
1. การประกาศเมทอด

ในความเป็นจริงเราเคยได้สร้างเมทอดขึ้นมาเองมาก่อนหน้านี้แล้ว ยิ่งไปกว่านั้นเราได้สร้างเมทอดลักษณะนี้เสมอในทุก ๆ โปรแกรมภาษา C# ที่เราเขียน เมทอดดังกล่าวก็คือเมทอด *Main* ที่ใช้สำหรับระบุจุดตั้งต้นของโปรแกรมนั่นเอง ลองพิจารณาโครงสร้างของโปรแกรมภาษา C# ที่เราก่อนเคยมาก่อนแล้วดังนี้





โครงสร้างโปรแกรมด้านบนเป็นโครงสร้างอย่างง่ายที่เราได้ใช้กันมาตั้งแต่แรกซึ่งประกอบด้วยเมที่อด *Main* เพียงเมที่อดเดียว หากเราเขียนโปรแกรมที่ประกอบด้วยเมที่อดอื่น ๆ นอกเหนือจาก *Main* โครงสร้างของโปรแกรมจะมีลักษณะดังนี้



สังเกตว่าเมที่อดแต่ละอันต้องถูกประกาศอยู่นอกเมที่อดอื่น ๆ แต่อยู่ภายในคลาสเสมอ น อ ก จ า ก นี้ น เมที่อด *Main* ไม่จำเป็นต้องถูกประกาศเป็นเมที่อดแรกของโปรแกรม การประกาศเมที่อดจะมีรูปแบบดังนี้

```
static return_type method_name(parameter_list)
{
    statement1;
    statement2;
    :
}
```

จะเห็นว่าเมทอด *Main* ที่เราใช้กันมาตั้งแต่แรกก็มีการเขียนอยู่ในรูปแบบข้างต้นเช่นเดียวกัน เราสามารถแบ่งชนิดของเมทอดเป็นสองประเภทคร่าว ๆ ได้แก่ *เมทอดแบบไม่คืนค่า* (non-value returning methods) และ *เมทอดแบบคืนค่า* (value-returning methods)

2. เมทอดแบบไม่คืนค่า

เมทอดแบบไม่คืนค่า (บางครั้งเรียกว่า subroutine หรือ procedure) เป็นเมทอดที่เขียนขึ้นมาเพื่อปฏิบัติงานบางอย่างและจบงานนั้นในตัวเองโดยไม่ส่งค่าคืนกลับไปยังผู้เรียก การประกาศเมทอดประเภทนี้ต้องมีการใช้คีย์เวิร์ด **void** ในตำแหน่ง *return_type* และการเรียกใช้เมทอดประเภทนี้จะปรากฏอยู่ในโปรแกรมเสมือนคำสั่งหนึ่งคำสั่ง

ตัวอย่างที่ 5.1 โปรแกรมต่อไปนี้ประกาศเมทอดชื่อ *PrintLine* เพื่อทำหน้าที่พิมพ์ดาว (*) 10 ดวงออกทางหน้าจอ จากนั้นภายในเมทอด *Main* จะเรียกใช้เมทอด *PrintLine* สองครั้ง ดังนั้นผลลัพธ์ของโปรแกรมจะปรากฏเป็นดาวสองแถว แถวละ 10 ดวง

```
1: using System;
2: class MyClass {
3:     static void PrintLine() {
4:         for (int i = 0; i < 10; i++)
5:             Console.Write('*');
6:         Console.WriteLine();
7:     }
8:
9:     static void Main() {
10:        MyClass.PrintLine();
11:        MyClass.PrintLine();
12:    }
13: }
```

จากการเรียกใช้งานเมทอดในบรรทัดที่ 10-11 จะเห็นว่าการใช้งานเมทอดนี้จะอยู่ในรูป *ClassName.MethodName* โดย *ClassName* ในที่นี้คือ *MyClass* ซึ่งเป็นชื่อคลาสที่เราใช้ในโปรแกรมนี้นี้ (บรรทัดที่ 2) และเป็นคลาสที่เมทอด *PrintLine* ถูกนิยามไว้ ดังนั้นการใช้งานเมทอดที่สร้างขึ้นมาเองจึงเป็นลักษณะเดียวกันกับการใช้งานเมทอดที่ภาษา C# เตรียมไว้ให้แล้ว เช่น *Console.WriteLine* ซึ่ง *WriteLine* คือชื่อเมทอดและ *Console* คือชื่อคลาสนั่นเอง



อย่างไรก็ตามหากการเรียกใช้งานเมธอดอยู่ภายในคลาสเดียวกันกับคลาสที่เมธอดนั้น ๆ ถูกประกาศไว้ ภาษา C# อนุญาตให้เราละส่วนของชื่อคลาสออกไปได้ ดังนั้นเมธอด *Main* ในตัวอย่างข้างต้นจึงเขียนให้สั้นลงได้ดังนี้

```
static void Main() {  
    PrintLine();  
    PrintLine();  
}
```

ใบความรู้ที่ 5.2

การส่งค่าไปยังเมทอด

เราสามารถสร้างเมทอดที่มีการรับค่าจากผู้เรียกเพื่อกำหนดพฤติกรรมการทำงานของเมทอดนั้น ๆ ค่าที่ถูกส่งไปนี้เรียกว่า อาร์กิวเมนต์ (argument) ส่วนเมทอดที่ถูกเรียกจะรับค่าเหล่านี้ผ่านมาทางพารามิเตอร์ (parameter) ซึ่งถูกนิยามไว้ในส่วน *parameter_list* ของการประกาศเมทอด (ลองย้อนกลับไปดูรูปแบบการประกาศเมทอดข้างต้น) เมทอดแต่ละอันสามารถถูกประกาศให้มีพารามิเตอร์ได้ตั้งแต่ศูนย์ตัวหรือมากกว่า โดยพารามิเตอร์แต่ละตัวจะต้องมีรูปแบบข้อมูลกำกับไว้เสมอ และอาร์กิวเมนต์ที่ใช้ในขณะเรียกใช้งานเมทอดจะต้องมีรูปแบบข้อมูลที่ตรงกัน

ตัวอย่างที่ 5.2 เมทอด *PrintCharLine* ด้านล่างนี้ดัดแปลงมาจากเมทอด *PrintLine* เมทอดใหม่นี้กำหนดให้มีพารามิเตอร์สองตัว คือ *c* เป็นชนิด **char** และ *len* เป็นชนิด **int** โดยที่พารามิเตอร์ *c* ใช้สำหรับระบุอักขระที่จะพิมพ์ออกทางจอภาพ) ไม่ได้พิมพ์เพียงแค่ตัวอีกต่อไป (และ *len* ใช้ระบุจำนวนอักขระที่ต้องการพิมพ์ในหนึ่งบรรทัด

```
1: using System;
2: class ParamNoRet {
3:     static void PrintCharLine(char c, int len) {
4:         for (int i = 0; i < len; i++)
5:             Console.Write(c);
6:         Console.WriteLine();
7:     }
8:
9:     static void Main() {
10:        PrintCharLine('o', 10);
11:        PrintCharLine('x', 20);
12:    }
13: }
```

จากโปรแกรมข้างต้น การเรียกใช้เมทอด *PrintCharLine* ครั้งแรกภายในเมทอด *Main* ในบรรทัดที่ 10 มีอาร์กิวเมนต์เป็น 'o' และ 10 ซึ่งเมทอด *PrintCharLine* จะรับค่าอาร์กิวเมนต์นี้เข้ามาอยู่ในพารามิเตอร์ *c* และ *len* ตามลำดับ ดังนั้นจะเห็นว่าบรรทัดที่ 4 และ 5 จะผลทำให้โปรแกรมพิมพ์อักขระ o ออกทางหน้าจอเป็นจำนวน 10 ตัว ในทำนองเดียวกัน การเรียกใช้เมทอด *PrintCharLine* ในบรรทัดที่ 11 จะมีผลทำให้โปรแกรมพิมพ์อักขระ x ออกทางหน้าจอเป็นจำนวน 20 ตัว



ใบความรู้ที่ 5.3

เมทอดแบบคืนค่า

เมทอดแบบคืนค่าเป็นเมทอดที่ส่งผลลัพธ์กลับไปให้ยังผู้เรียกหลังจากการทำงานในเมทอดเสร็จสิ้นลงตัวอย่างที่เราคุ้นเคยคืออยู่แล้วคือเมทอด `Console.ReadLine` ซึ่งทำหน้าที่รอรับข้อมูลจากผู้ใช้งานและคืนค่าที่ผู้ใช้ป้อนในรูปแบบของข้อความเพื่อให้โปรแกรมนำไปใช้ต่อไป หรือเมทอด `int.Parse` ที่มีการรับค่าพารามิเตอร์ในรูปแบบข้อความและคืนค่ากลับออกมาเป็นตัวเลขที่สามารถนำไปใช้ในการคำนวณต่อไปได้ ดังนั้นโดยทั่วไปแล้วเมทอดประเภทนี้จะปรากฏอยู่ในโปรแกรมในรูปแบบของนิพจน์ชนิดต่าง ๆ ขึ้นอยู่กับชนิดของข้อมูลที่เมทอดนั้น ๆ คืนกลับมาให้ อย่างไรก็ตามเราสามารถเรียกใช้เมทอดเหล่านี้เป็นคำสั่งโดด ๆ ได้ เช่นเดียวกับเมทอดที่ไม่คืนค่าหากเราไม่ต้องการนำค่าที่คืนกลับมาไปประมวลผลใด ๆ

การสร้างเมทอดแบบคืนค่าขึ้นมาเองนั้นต้องระบุชนิดของข้อมูลที่เมทอดจะส่งกลับเอาไว้ในส่วนที่เป็น `return_type` ของการประกาศเมทอดนั้น ๆ แทนที่จะใช้คีย์เวิร์ด **void** เหมือนที่เคยใช้กับเมทอดแบบไม่คืนค่า และภายในตัวเมทอดเองจะต้องมีการใช้คำสั่ง **return** เพื่อให้เมทอดสิ้นสุดการทำงานทันทีและส่งค่ากลับไปยังผู้เรียก รูปแบบการใช้งานคำสั่ง **return** เป็นดังนี้

```
return expression;
```

โดยแทนที่ `expression` ด้วยนิพจน์ที่จะถูกประเมินเป็นค่าที่ต้องการส่งกลับ

เมทอดแบบคืนค่านิยมใช้ในกรณีที่ต้องการแจ้งให้ผู้เรียกใช้งานเมทอดทราบถึงสถานะการทำงานในเมทอดเองว่าทำงานได้เสร็จสมบูรณ์หรือผิดพลาดอย่างไร โดยรายงานอยู่ในรูปของค่าความจริง **true** หรือ **false** หรือหากความผิดพลาดสามารถเกิดขึ้นได้ในหลายกรณีแล้วเมทอดนั้น ๆ สามารถรายงานความผิดพลาดกลับไปได้ในรูปแบบของรหัสความผิดพลาดที่กำหนดขึ้นมาเอง (เช่น ค่า 0 หมายถึงทำงานถูกต้องสมบูรณ์ 1 หมายถึงผู้ใช้ป้อนข้อมูลผิด หรือ 2 หมายถึงผู้ใช้ยกเลิกการทำงาน เป็นต้น)

อีกกรณีหนึ่งที่เมทอดแบบคืนค่าถูกนำมาใช้มากคือการนิยามฟังก์ชัน (function) ทางคณิตศาสตร์ขึ้นมาเองเพื่อชดเชยจากส่วนที่ขาดไปในคลาส `Math` อีกทั้งยังใช้ในการสร้างฟังก์ชันเพื่อคำนวณที่ซับซ้อนเพื่อซ่อนความซับซ้อนนั้นไม่ให้ปรากฏในส่วนของโปรแกรมหลักดังเช่นตัวอย่างต่อไปนี้

ตัวอย่างที่ 5.3 โปรแกรมต่อไปนี้เป็นนิยามเมทอดชื่อ *TriangleArea* ขึ้นมาเพื่อคำนวณพื้นที่ของสามเหลี่ยมโดยรับพารามิเตอร์เป็นความสูงและความกว้างของสามเหลี่ยม ส่วนภายในเมทอด *Main* จะเป็นเพียงการถามค่าความสูงและความกว้างจากผู้ใช้แล้วส่งค่าให้กับ *TriangleArea* เพื่อคำนวณพื้นที่

```

1: using System;
2: class Triangle {
3:     static double TriangleArea(double w, double h) {
4:         double area = w*h/2.0;
5:         return area;
6:     }
7:
8:     static void Main() {
9:         Console.Write("Enter width: ");
10:        double width = double.Parse(Console.ReadLine());
11:        Console.Write("Enter height: ");
12:        double height = double.Parse(Console.ReadLine());
13:        Console.WriteLine("The area of the triangle is {0:f2}.",
14:            TriangleArea(width, height));
15:    }
16: }

```

สังเกตว่าเนื่องจากเมทอด *TriangleArea* ถูกประกาศให้คืนค่าเป็นชนิด **double** ในบรรทัดที่ 3 ดังนั้นการเรียกใช้เมทอดนี้จึงอยู่ในรูปของนิพจน์แบบ **double** ทำให้สามารถใช้งานร่วมกับเมทอด *Console.WriteLine* ในบรรทัดที่ 13-14 ได้ทันที

ตัวอย่างที่ 5.4 ตัวอย่างนี้แสดงการนิยามฟังก์ชัน

$$f(n) = \sum_{i=1}^n \sqrt{i}$$

ให้อยู่ในรูปของเมทอดที่ถูกเรียกใช้จากเมทอด *Main* เพื่อแสดงตารางค่าของ $f(n)$ สำหรับแต่ละค่าของ n ตั้งแต่ 1 ถึง 15

```

1: using System;
2: class Function {
3:     static double f(int n) {
4:         double sum = 0.0;
5:         for (int i = 1; i <= n; i++)
6:             sum += Math.Sqrt(i);
7:
8:         return sum;
9:     }
10:
11:     static void Main() {
12:         Console.WriteLine(" n | f(n)");
13:         Console.WriteLine("----+-----");
14:         for (int n = 1; n <= 15; n++) {
15:             Console.WriteLine("{0,2} | {1:f3}", n, f(n));
16:         }
17:     }
18: }

```



ตัวอย่างผลการทำงาน

n	f(n)
1	1.000
2	2.414
3	4.146
4	6.146
5	8.382
6	10.832
7	13.478
8	16.306
9	19.306
10	22.468
11	25.785
12	29.249
13	32.855
14	36.596
15	40.469

หมายเหตุ: สัญลักษณ์ $\{0, 2\}$ ที่ใช้ในคำสั่ง `Console.WriteLine` ในบรรทัดที่ 15 คือการกำหนดให้ผลลัพธ์ในการพิมพ์ค่าในนิพจน์แรก (ซึ่งก็คือค่าในตัวแปร n) มีความยาวอย่างน้อยสองอักขระ เช่น หาก n มีค่าเท่ากับ 3 ผลลัพธ์ที่พิมพ์ออกทางหน้าจอคือ ' 3' คือมีช่องว่างนำหน้าหนึ่งช่อง (ไม่รวมเครื่องหมาย ') ทำให้ผลลัพธ์โดยรวมมีความยาวสองอักขระ แต่ถ้า n มีค่าเท่ากับ 15 โปรแกรมจะแสดงผล ' 15' โดยไม่มีช่องว่างเนื่องจาก 15 มีความยาวสองอักขระอยู่แล้ว เหตุผลที่ต้องทำแบบนี้เพื่อให้เส้นตารางในผลลัพธ์ดูสวยงาม ไม่เหลื่อมซ้อนกัน

กิจกรรมที่ 6

อาร์เรย์

1. จุดประสงค์ ให้ผู้เรียนสามารถ

- 1.1 สร้างอาร์เรย์หนึ่งมิติ
- 1.2 อ้างถึงข้อมูลในอาร์เรย์
- 1.3 เข้าถึงข้อมูลในอาร์เรย์โดยใช้คำสั่ง foreach
- 1.4 ส่งอาร์เรย์ไปยังเมทอด

2. แนวคิด

อาร์เรย์ (arrays) คือชุดหรือกลุ่มของข้อมูลชนิดเดียวกัน ที่แต่ละตัวเป็นส่วนประกอบหนึ่งของกลุ่มสมาชิกที่เรียงต่อกันไปเป็นลำดับโดยที่ข้อมูลแต่ละตัวจะถูกเรียกว่าสมาชิกของอาร์เรย์ (elements) ทำให้สามารถเก็บข้อมูลได้จำนวนมากๆ โดยไม่ต้องประกาศตัวแปรหลายตัวเพื่อเก็บข้อมูลเหล่านี้

การเข้าถึงสมาชิกของอาร์เรย์แต่ละตัว ทำได้โดยการกำหนดดัชนีซึ่งเป็นจำนวนเต็มที่เขียนอยู่ภายในวงเล็บก้ามปู “[]” ตามหลังชื่ออาร์เรย์ เช่น `x[0]` และ `y[0]` เป็นต้น สมาชิกของอาร์เรย์จะมีดัชนี (index) หรือตำแหน่งของสมาชิกแรกเป็นศูนย์ (0) และตัวต่อไปจะเพิ่มขึ้นทีละ 1 ตามลำดับจนถึงตัวสุดท้ายตัวที่ `n` จะมีดัชนีเป็น `n-1`

3. สื่อและอุปกรณ์

3.1 ใบงาน

ใบงานที่	เรื่อง	เวลา (นาที)
6.1	สร้างอาร์เรย์ของคะแนน	10
6.2	หาค่าเฉลี่ยของคะแนน	5
6.3	รับส่งอาร์เรย์ไปยังเมทอด	10
6.4	วิเคราะห์อักขระ	10
6.5	ถอดรหัสลับจารชน	20
6.6	แผนภูมิแท่ง	20

3.2 ใบความรู้

- ใบความรู้ที่ 6.1 เรื่องชนิดข้อมูลแบบอาร์เรย์
- ใบความรู้ที่ 6.2 เรื่องการหาขนาดของอาร์เรย์
- ใบความรู้ที่ 6.3 เรื่องคำสั่ง foreach



- ใบความรู้ที่ 6.4 เรื่องการส่งอาร์เรย์ไปยังเมทอด
- ใบความรู้ที่ 6.5 เรื่องการการอ้างถึงสตริงในรูปแบบอาร์เรย์

3.3 อื่นๆ

4. วิธีดำเนินการกิจกรรม

4.1 การจัดเตรียม

- 4.1.1 แบ่งผู้เรียนออกเป็นกลุ่ม กลุ่มละ 2 คน
- 4.1.2 เตรียมใบงานที่ 6.1 – 6.6 ตามจำนวนกลุ่มและใบความรู้ที่ 6.1 – 6.5 ตามจำนวนผู้เรียน

4.2 ขั้นตอนการดำเนินการ

- 4.2.1 ผู้สอนกล่าวถึงข้อมูลเบื้องต้นเกี่ยวกับอาร์เรย์
- 4.2.2 ผู้เรียนศึกษาใบความรู้ที่ 6.1 เรื่องชนิดข้อมูลแบบอาร์เรย์ แล้วทำใบงานที่ 6.1 เรื่องสร้างอาร์เรย์ของคะแนน
- 4.2.3 ผู้เรียนศึกษาใบความรู้ที่ 6.2 เรื่องการหาขนาดของอาร์เรย์ แล้วทำใบงานที่ 6.2 เรื่องหาค่าเฉลี่ยของคะแนน
- 4.2.4 ผู้เรียนศึกษาใบความรู้ที่ 6.3 เรื่องคำสั่ง foreach และใบความรู้ที่ 6.4 เรื่องการส่งอาร์เรย์ไปยังเมทอด แล้วทำใบงานที่ 6.3 เรื่องรับส่งอาร์เรย์ไปยังเมทอด
- 4.2.5 ผู้เรียนศึกษาใบความรู้ที่ 6.5 เรื่องการการอ้างถึงสตริงในรูปแบบอาร์เรย์ แล้วทำใบงานที่ 6.4 เรื่องวิเคราะห์อักขระ และใบงานที่ 6.5 เรื่องถอดรหัสลับจารชน
- 4.2.6 ผู้เรียนแต่ละกลุ่มทำใบงานที่ 6.6 เรื่องแผนภูมิแท่ง จากนั้นผู้สอนสุ่มกลุ่มออกมานำเสนอ
- 4.2.7 ผู้เรียนและผู้สอนร่วมกันสรุปการใช้งานอาร์เรย์

5. การวัดและประเมินผล

5.1 ตรวจคำตอบจากใบงาน

6. แหล่งความรู้เพิ่มเติม

- 6.1 [http://msdn.microsoft.com/th-th/vcsharp/default\(en-us\).aspx](http://msdn.microsoft.com/th-th/vcsharp/default(en-us).aspx)

7. ข้อเสนอแนะ

-



ใบงานที่ 6.1

สร้างอาร์เรย์ของคะแนน

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 5.1 จากนั้นสร้างโปรเจกต์และเขียนโปรแกรมตามโจทย์ที่กำหนดให้ต่อไปนี้

1. ให้ผู้เรียนเติมค่าในช่องว่างที่เว้นไว้ในโปรแกรมด้านล่างเพื่อให้โปรแกรมสร้างอาร์เรย์ชื่อ *stdScores* สำหรับเก็บข้อมูลแบบ **double** จำนวน 6 จำนวน โดยข้อมูลในอาร์เรย์มีค่าเป็น 78.0, 51.2, 25.1, 62.6, 27.6 และ 18.0 ตามลำดับ จากนั้นให้สร้างโปรเจกต์และเขียนโปรแกรมคำนวณผลรวมของค่าเหล่านี้ และแสดงผลลัพธ์ออกทางจอภาพด้วยทศนิยม 2 ตำแหน่ง

```
using System;
class ScoreSum {
    static void Main() {
        double[] stdScores =
            _____;

        double sum = 0.0;
        for (int i=0; i<6; i++) {
            _____;
        }
        Console.WriteLine("Sum = {0:f2}", sum);
    }
}
```

ทดลองรันโปรแกรมและกรอกผลลัพธ์ของโปรแกรมลงในช่องว่าง



ใบงานที่ 6.2

หาค่าเฉลี่ยของคะแนน

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

1. ให้ผู้เรียนศึกษาใบความรู้ที่ 6.2 เรื่องการหาขนาดของอาเรย์
2. สร้างโปรเจกต์และเขียนโปรแกรมโดยดัดแปลงโปรแกรมจากใบความรู้ที่ 6.2 เพื่อให้โปรแกรมคำนวณค่าเฉลี่ยของคะแนน โดยให้ใช้คุณสมบัติ *Length* ภายในตัวโปรแกรมแทนที่จะใช้ค่า 6 โดยตรง คัดลอกโปรแกรมและผลลัพธ์ของโปรแกรมลงในช่องว่าง

โปรแกรมที่ดัดแปลงแล้ว

ผลลัพธ์ของโปรแกรม

ใบงานที่ 6.3

รับส่งอาร์เรย์ไปยังเมทอด

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 6.3 ใบความรู้ที่ 6.4 และใบความรู้ที่ 6.4 จากนั้นให้สร้างโปรเจกต์และเขียนโปรแกรมจากโจทย์ที่กำหนดให้ต่อไปนี้

ดัดแปลงโปรแกรมจากตัวอย่างที่ 6.7 จากใบความรู้ที่ 6.4 เพื่อรับข้อมูลจากผู้ใช้งานไว้ในอาร์เรย์แทนการกำหนดค่าเริ่มต้นไว้ก่อน และเพิ่มเมทอดชื่อ *ArrayAverage* เพื่อคำนวณค่าเฉลี่ยของค่าในอาร์เรย์ของข้อมูลชนิด **double** ให้เติมคำสั่งลงในบรรทัดที่เว้นไว้เพื่อให้โปรแกรมทำงานได้อย่างถูกต้องโดยใช้พื้นที่ไม่เกินสองบรรทัด (คำแนะนำ: พยายามใช้ประโยชน์จากเมทอด *ArraySum* ที่มีให้อยู่แล้ว)

```
using System;
class ArrayTest {
    static double ArraySum(double[] data) {
        double sum = 0;
        foreach (double x in data)
            sum += x;
        return sum;
    }

    static double ArrayAverage(double[] data) {
        _____
        _____
    }

    static void Main() {
        double[] myData;
        int n;
        Console.WriteLine("Enter number of items: ");
        n = int.Parse(Console.ReadLine());
        myData = new double[n];
        for (int i = 0; i < n; i++) {
            Console.WriteLine("Enter item#{0}: ", i+1);
            myData[i] = double.Parse(Console.ReadLine());
        }

        Console.WriteLine("Average = {0:f2}", ArrayAverage(myData));
    }
}
```



ตัวอย่างผลการทำงาน

```
Enter number of items: 4  
Enter item#1: 10.5  
Enter item#2: 67  
Enter item#3: 3  
Enter item#4: 0.6  
Average = 20.28
```



ใบงานที่ 6.4

วิเคราะห์อักขระ

รายชื่อสมาชิกในกลุ่มที่.....

- | | |
|---------|---------|
| 1. | 2. |
| 3. | 4. |

ให้ผู้เรียนศึกษาใบความรู้ที่ 6.5 จากนั้นสร้างโปรเจกต์และเขียนโปรแกรมต่อไปนี้ให้สมบูรณ์

จงเขียนโปรแกรมเพื่อให้โปรแกรมสามารถรับข้อความจากผู้ใช้และรายงานจำนวนอักขระที่เป็นตัวเลข ('0'..'9') จำนวนอักขระภาษาอังกฤษตัวใหญ่ ('A'..'Z') และจำนวนอักขระภาษาอังกฤษตัวเล็ก ('a'..'z')

```
using System;
class CountAll {
    static void Main() {
        Console.Write("Enter a string: ");
        string s = Console.ReadLine();
        int cntDigit = 0, cntUpper = _____, cntLower = _____;
        foreach (char c in s) {
            if (c >= '0' && _____)
                cntDigit++;
            if (_____ && c <= 'Z')
                cntUpper++;
            if (_____)
                _____;
        }
        Console.WriteLine("There are {0} digits.", cntDigit);
        Console.WriteLine("There are {0} uppercase letters.",
            cntUpper);
        Console.WriteLine("There are {0} lowercase letters.",
            cntLower);
    }
}
```

ตัวอย่างผลการทำงาน

```
Enter a string: Welcome KU-66
There are 2 digits.
There are 3 uppercase letters.
There are 6 lowercase letters.
```



ใบงานที่ 6.5

ถอดรหัสลับจารชน

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

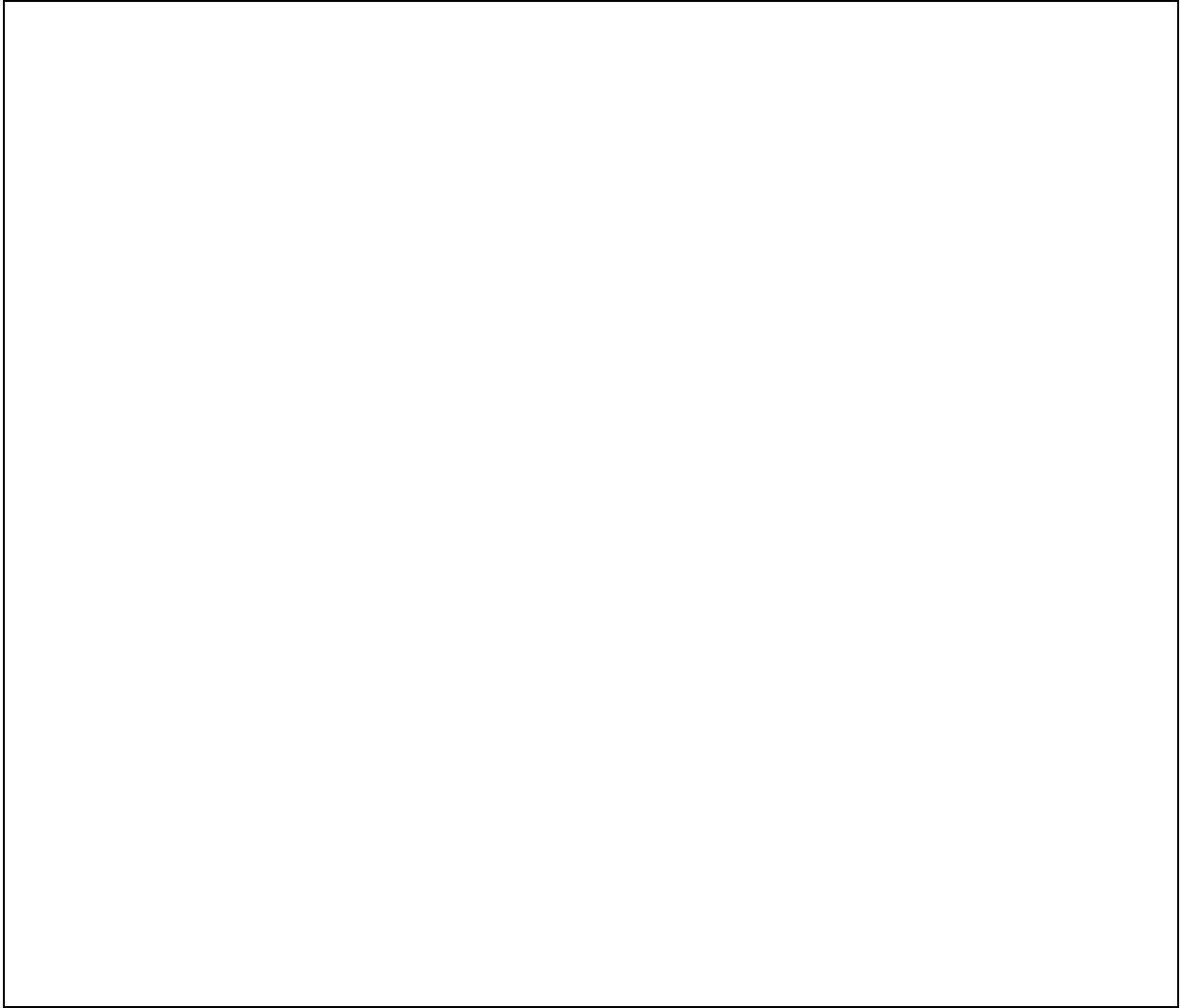
ให้เขียนโปรแกรมเพื่อแก้ปัญหาตามสถานการณ์ต่อไปนี้

ท่านได้รับการติดต่อจากสำนักงานตำรวจสากลเพื่อล้วงข้อมูลจากจดหมายอิเล็กทรอนิกส์ที่ใช้ติดต่อกันระหว่างผู้ก่อการร้ายข้ามชาติ (เรื่องราวในโจทย์เป็นเรื่องที่แต่งขึ้นโดยไม่เกี่ยวข้องกับบุคคล สถานที่ หรือเหตุการณ์ที่มีอยู่จริง) อย่างไรก็ตามข้อความที่ดักจับมาได้นั้นถูกเข้ารหัสเอาไว้ทำให้ไม่สามารถอ่านเข้าใจได้โดยง่าย หลังจากการลองผิดลองถูกนานนับปีและการเสี่ยงชีวิตนับครั้งไม่ถ้วนเพื่อแฝงตัวเข้าไปในกลุ่มผู้ก่อการร้ายในที่สุดท่านก็ได้ล้วงความลับเรื่องกรรมวิธีการเข้ารหัสซึ่งเป็นไปตามกฎเกณฑ์อันแสนซับซ้อนดังนี้

- ตัวหนังสือถูกพิมพ์จากขวาไปซ้าย แทนที่จะเป็นซ้ายไปขวาเหมือนปกติ
- อักษร 'R' ถูกแทนด้วยอักษร 'E' ในการเข้ารหัส
- อักษร 'W' ถูกแทนด้วยอักษร 'R' ในการเข้ารหัส
- อักษร 'E' ถูกแทนด้วยอักษร 'W' ในการเข้ารหัส
- อักษรอื่น ๆ รวมถึงตัวเลขและสัญลักษณ์ต่าง ๆ ไม่มีการเปลี่ยนแปลงใด ๆ

เมื่อได้ข้อมูลสำคัญเหล่านี้มาแล้ว ท่านจึงตัดสินใจเขียนโปรแกรมภาษา C# ขึ้นมาเพื่อเพิ่มความเร็วในการถอดรหัส โปรแกรมนี้จะรับข้อความที่ถูกเข้ารหัสเอาไว้และรายงานผลลัพธ์ออกมาเป็นข้อความที่ถูกถอดรหัสแล้ว ดังตัวอย่าง

Enter text: **SNOPAWR WEOM DWWN**
NEED MORE WEAPONS



จากนั้นลองใช้โปรแกรมที่เขียนขึ้นมาถอดรหัสข้อความต่อไปนี้

LUFITUAWB SI WFIL

ถอดรหัสได้เป็น.....

GNIMMAEGOEP WVOL I

ถอดรหัสได้เป็น.....

UOY HTIR WB WCEO WHT YAM

ถอดรหัสได้เป็น.....



ใบงานที่ 6.6

แผนภูมิแท่ง

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้เขียนโปรแกรมเพื่อแก้ปัญหาตามสถานการณ์ต่อไปนี้

คุณครูท่านหนึ่งต้องการเปรียบเทียบคะแนนสอบของนักเรียนในห้องเรียน แต่การพิจารณาจากคะแนนที่เป็นตัวเลขในตารางนั้นเป็นการยากที่จะแยกแยะคนที่ได้คะแนนสูงต่ำออกจากกัน เพื่อช่วยเหลือคุณครูท่านนี้ ท่านได้เสนอตัวเขียนโปรแกรมสำหรับประมวลผลคะแนนให้อยู่ในรูปแบบแผนภูมิแท่ง โดยโปรแกรมจะรับค่าจำนวนนักเรียนและคะแนนของนักเรียนแต่ละคนจากผู้ใช้ และนำคะแนนแต่ละคนมาวาดกราฟแท่งตามแนวขวางที่มีความยาวเท่ากับคะแนนที่ได้ ดังตัวอย่าง

How many students? 6

Student#1 score: 25

Student#2 score: 39

Student#3 score: 48

Student#4 score: 12

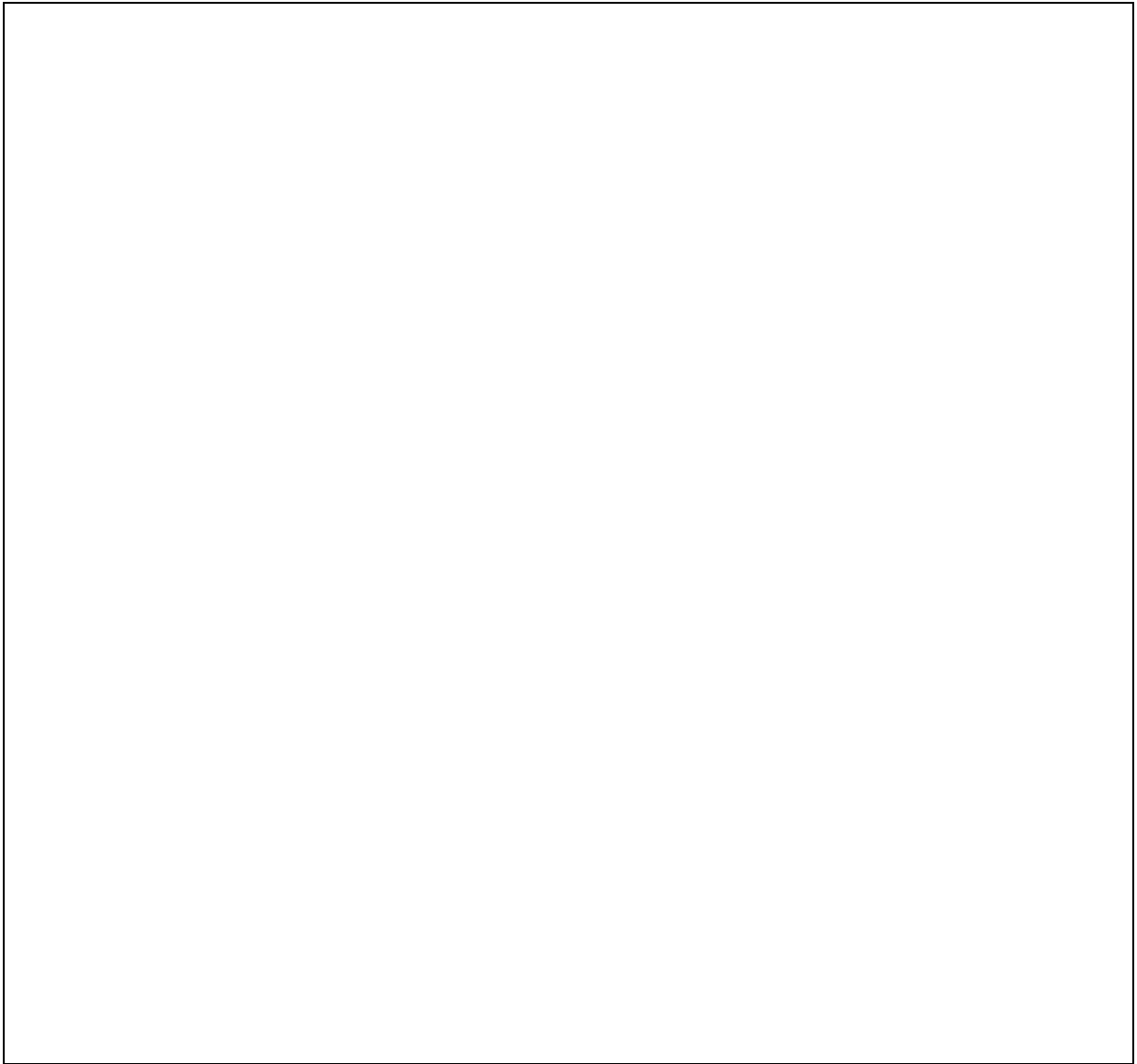
Student#5 score: 20

Student#6 score: 30

ID | Score

```

-----
1 | ***** ( 25 )
2 | ***** ( 39 )
3 | ***** ( 48 )
4 | ***** ( 12 )
5 | ***** ( 20 )
6 | ***** ( 30 )
  
```



ใบความรู้ที่ 6.1

ชนิดข้อมูลแบบอาร์เรย์

คอมพิวเตอร์เป็นอุปกรณ์ที่เพิ่มความรวดเร็วในการประมวลผลข้อมูลโดยเฉพาะอย่างยิ่งสำหรับงานที่มีปริมาณข้อมูลมาก ๆ ที่อาศัยการประมวลผลในรูปแบบเดียวกัน ดังนั้นการเขียนโปรแกรมเพื่อจัดการงานเหล่านี้จะต้องมีการเก็บข้อมูลเป็นจำนวนมากลงในตัวแปร ในกรณีนี้ หากเราประกาศตัวแปรหนึ่งตัวสำหรับข้อมูลหนึ่งจำนวน โปรแกรมของเราจะประกอบด้วยตัวแปรเป็นจำนวนมาก นอกจากนั้นการเข้าถึงค่าภายในตัวแปรจะต้องกระทำผ่านตัวแปรแต่ละตัวโดยตรงและไม่สามารถรวบรวมการประมวลผลในรูปแบบเดียวกันไว้ในโครงสร้างแบบวนซ้ำได้ ภาษาโปรแกรมส่วนใหญ่รวมทั้ง C# จึงรองรับการใช้งานโครงสร้างการเก็บข้อมูลชนิดพิเศษที่อนุญาตให้เราจัดการกับข้อมูลชนิดเดียวกันหลาย ๆ จำนวนผ่านตัวแปรเพียงตัวเดียว โครงสร้างการเก็บข้อมูลดังกล่าวมีชื่อเรียกว่า *อาร์เรย์* (array) และตัวแปรที่ประกาศขึ้นมาเพื่อใช้อ้างถึงอาร์เรย์จะถูกเรียกว่า *ตัวแปรแบบอาร์เรย์*

1. การประกาศตัวแปรแบบอาร์เรย์

ดังเช่นตัวแปรทั่ว ๆ ไป ตัวแปรแบบอาร์เรย์จะต้องมีการประกาศล่วงหน้าก่อนการใช้งาน โดยมีรูปแบบดังต่อไปนี้

```
DataType[] ArrayName;
```

ในที่นี้ *ArrayName* คือชื่อตัวแปรแบบอาร์เรย์ และ *DataType* คือชนิดข้อมูลแต่ละจำนวนที่ตัวแปรนี้เก็บค่าได้ สังเกตว่าการประกาศตัวแปรแบบอาร์เรย์มีความคล้ายคลึงกับการประกาศตัวแปรแบบปกติแต่ต่างกันเพียงเครื่องหมาย `[]` ที่ต้องใส่เพิ่มเข้าไปเท่านั้น นอกจากนั้นหากเรามองการประกาศตัวแปรในลักษณะนี้เทียบกับการประกาศตัวแปรปกติที่เราคุ้นเคยจะเห็นว่าการประกาศดังกล่าวจะเทียบเท่ากับการประกาศตัวแปรชื่อ *ArrayName* ที่มีชนิดของการเก็บข้อมูลเป็น *DataType[]* ซึ่งหมายถึงอาร์เรย์ของข้อมูลประเภท *DataType* นั้นเอง

ตัวอย่างที่ 6.1 ส่วนของโปรแกรมต่อไปนี้เป็นการประกาศตัวแปรแบบอาร์เรย์สามตัวคือ *score*, *grade* และ *Name* เพื่อเก็บข้อมูลประเภทจำนวนเต็ม อักขระ และสตริง ตามลำดับ ให้พิจารณาการประกาศตัวแปรแบบอาร์เรย์เหล่านี้เทียบกับการประกาศตัวแปรตามปกติในตอนท้ายโดยดูความหมายจากคอมเมนต์ที่ท้ายแต่ละบรรทัด

```
int[] score;    // score refers to an array of integers
char[] grade;  // grade refers to an array of characters
string[] name; // Name refers to an array of strings
```

```
int s;           // s refers to an integer
char c;          // c refers to a character
string n;        // n refers to a string
```

2. การสร้างอาร์เรย์

ตัวแปรแบบอาร์เรย์ที่ถูกประกาศขึ้นมาตามรูปแบบข้างต้นนั้นสามารถนำไปใช้เพียงแค่อ้างถึงอาร์เรย์ชนิดนั้นๆเท่านั้น อย่างไรก็ตาม ณ ขณะนี้เรายังไม่มีอาร์เรย์ที่แท้จริงที่จะอ้างถึง ดังนั้นขั้นตอนต่อไปจะเป็นการสั่งให้คอมพิวเตอร์สร้างอาร์เรย์ขึ้นมาไว้ในหน่วยความจำของเครื่อง โดยคำสั่งที่เราจะใช้ในกระบวนการนี้คือคำสั่ง **new** ซึ่งมีรูปแบบการใช้งานดังนี้

```
new DataType[num_elements]
```

ในที่นี้ *DataType* คือชนิดข้อมูล และ *num_elements* คือนิพจน์แบบจำนวนเต็มแสดงขนาดของอาร์เรย์ที่เราต้องการสร้าง การใช้งานคำสั่ง **new** ในรูปแบบข้างต้นนั้นอยู่ในรูปของนิพจน์ที่ให้ค่าอ้างอิงไปยังอาร์เรย์ที่เพิ่งถูกสร้างขึ้นมา ดังนั้นโดยส่วนใหญ่แล้วเราจะนิยมนำค่าอ้างอิงนี้ไปให้กับตัวแปรแบบอาร์เรย์ที่เราได้ประกาศไว้ล่วงหน้าแล้ว เพื่อความสะดวกในการอ้างอิงถึงอาร์เรย์ตัวนี้ในภายหลัง ดังนั้นคำสั่ง **new** จึงมักพบในรูป

```
ArrayName = new DataType[num_elements];
```

โดยในที่นี้ *ArrayName* คือชื่อตัวแปรแบบอาร์เรย์ที่เราได้ประกาศเอาไว้ก่อนหน้านี้ และเช่นเดียวกับการประกาศและการให้ค่าเริ่มต้นกับตัวแปร เราสามารถรวมเอาการประกาศและการสร้างอาร์เรย์เข้าไว้ในคำสั่งเดียวกันได้เลย ดังเช่นตัวอย่างด้านล่าง

ตัวอย่างที่ 6.2 การประกาศตัวแปรแบบอาร์เรย์และการสร้างอาร์เรย์

- สร้างอาร์เรย์สำหรับเก็บจำนวนเต็ม 5 ตัว โดยอ้างอิงผ่านตัวแปรแบบอาร์เรย์ชื่อ *scores*

```
int[] scores;
scores = new int[5];
```

ตามที่ได้กล่าวไว้แล้วข้างต้น เราสามารถรวมเอาการประกาศตัวแปรแบบอาร์เรย์และการสร้างอาร์เรย์ไว้ในคำสั่งเดียวกันดังนี้

```
int[] scores = new int[5];
```

- สร้างอาร์เรย์สำหรับเก็บชื่อนักเรียน 10 คน อ้างอิงผ่านตัวแปรแบบอาร์เรย์ชื่อ *names*

```
string[] names;
names = new string[10];
```

หรือ

```
string[] names = new string[10];
```



นอกจากนี้อาร์เรย์ที่สร้างขึ้นใหม่ยังสามารถกำหนดค่าเริ่มต้นให้กับข้อมูลในอาร์เรย์ได้อีกด้วยโดยอาศัยคำสั่งในรูปแบบต่อไปนี้

```
ArrayName = new DataType[num_elements] {  
    value0, value1, ..., valueN-1};
```

สิ่งที่เพิ่มขึ้นมาคือวงเล็บปีกกาหลังจากคำสั่ง **new** ที่ภายในมีค่าต่างๆ จำนวน *num_elements* ค่า โดยค่า *value0* จะถูกกำหนดเป็นค่าเริ่มต้นให้กับข้อมูลแรกสุดในอาร์เรย์ แล้วตามด้วย *value1* สำหรับข้อมูลช่องถัดมาและไล่ต่อไปเรื่อยๆ จนถึงข้อมูลสุดท้าย

หากมีการกำหนดค่าเริ่มต้นให้กับข้อมูลในอาร์เรย์แล้ว ภาษา C# ยอมให้เราละการระบุขนาดของอาร์เรย์ที่เราสร้างขึ้นมาได้ เนื่องจากขนาดของอาร์เรย์ถูกกำหนดไว้โดยจำนวนค่าเริ่มต้นที่ใส่ไว้ในวงเล็บปีกกาอยู่แล้ว ดังนั้นเราจึงสามารถใช้รูปแบบคำสั่งดังนี้ได้

```
ArrayName = new DataType[] {value0, value1, ..., valueN-1};
```

สังเกตว่าเราได้ละส่วนที่เป็น *num_elements* ไว้ ยิ่งไปกว่านั้นคำสั่งข้างต้นยังสามารถเขียนให้สั้นลงได้อีกโดยละคำสั่ง **new** ไว้ จึงเหลือแค่เพียง

```
DataType[] ArrayName = {value0, value1, ..., valueN-1};
```

อย่างไรก็ตามการสร้างอาร์เรย์ในรูปแบบสุดท้ายนี้มีข้อจำกัดว่าจะต้องรวมอยู่ในคำสั่งเดียวกันกับการประกาศตัวแปรแบบอาร์เรย์

ตัวอย่างที่ 6.3 สร้างอาร์เรย์ชื่อ *scores* เพื่อเก็บจำนวนเต็ม 5 จำนวนโดยมีค่าเริ่มต้นเท่ากับ 10, 50, 10, 55 และ 60 ตามลำดับ

```
int[] scores = new int[5] {10, 50, 10, 55, 60};
```

หรือ

```
int[] scores = new int[] {10, 50, 10, 55, 60};
```

หรือ

```
int[] scores = {10, 50, 10, 55, 60};
```

3. การอ้างถึงข้อมูลในอาร์เรย์

เราสามารถเข้าถึงข้อมูลแต่ละจำนวนภายในอาร์เรย์ผ่านทางดัชนี (index) ที่สอดคล้องกับตำแหน่งของข้อมูลนั้นๆ โดยค่าดัชนีสำหรับข้อมูลตัวแรกสุดในอาร์เรย์ถูกกำหนดให้เป็นดัชนีหมายเลข 0 ถัดมาเป็นหมายเลข 1 เรื่อยไป ดังนั้นสำหรับอาร์เรย์ที่มีความยาว *N* แล้ว ค่าของดัชนีที่ใช้งานได้จึงเริ่มตั้งแต่ 0 ถึง *N-1* ในภาษา C# ข้อมูล ณ ตำแหน่ง *idx* ภายในอาร์เรย์ชื่อ *ArrayName* จะถูกอ้างถึงในรูปแบบ

```
ArrayName[idx]
```


การอ้างอิงถึงข้อมูลภายในอาร์เรย์ในรูปแบบนี้จะมีการใช้งานเสมือนเป็นตัวแปรโคดตัวหนึ่ง นั่นคือหากเราใช้การอ้างอิงนี้เป็นส่วนหนึ่งของนิพจน์ใดๆ ค่าภายในอาร์เรย์จะถูกดึงออกมาใช้เพื่อประเมินค่าของนิพจน์นั้นๆ ในทางตรงกันข้าม หากเราวางการอ้างอิงนี้ไว้ทางด้านซ้ายของเครื่องหมาย = ในคำสั่งให้ค่ากับตัวแปร ค่าของอาร์เรย์ตำแหน่งนี้ก็จะถูกเปลี่ยนค่าไปตามนั้น

ตัวอย่างที่ 6.4 พิจารณาอาร์เรย์ *scores* ซึ่งประกอบด้วยจำนวนเต็ม 5 จำนวน

- กำหนดค่า 52 ให้กับข้อมูลตัวแรกของอาร์เรย์

```
scores[0] = 52;
```

- นำข้อมูลในตำแหน่งท้ายสุดของอาร์เรย์มาแสดงผลบนหน้าจอ

```
Console.WriteLine(scores[4]);
```

- กำหนดให้ค่าทุกค่าในอาร์เรย์มีค่าเท่ากับ 3

```
for (int i = 0; i < 5; i++)  
    scores[i] = 3;
```

- คำนวณผลรวมของค่าทั้งหมดภายในอาร์เรย์

```
int sum = 0;  
for (int i = 0; i < 5; i++)  
    sum = sum + scores[i];
```

- นำค่าทั้งหมดในอาร์เรย์มาแสดงผลบนหน้าจอ

```
for (int i = 0; i < 5; i++)  
    Console.WriteLine("Student {0}: {1}", i+1, scores[i]);
```

ใบความรู้ที่ 6.2

การหาขนาดของอาร์เรย์

บ่อยครั้งที่เราอาจต้องเปลี่ยนแปลงขนาดของอาร์เรย์ภายในโปรแกรมที่เราได้เขียนขึ้นมาเรียบร้อยแล้ว หากเราต้องการเพิ่มเติมความสามารถของโปรแกรมให้ประมวลผลข้อมูลมากขึ้น หรือด้วยสาเหตุอื่นใดก็ตาม หากเราลองพิจารณาโปรแกรมในใบงานที่ 6.1 จะเห็นว่าการดัดแปลงโปรแกรมให้ทำงานกับอาร์เรย์ที่มีขนาดต่างไปจากเดิมนั้น เราไม่สามารถทำได้เพียงแค่เปลี่ยนคำสั่งในการสร้างอาร์เรย์เท่านั้น เรายังต้องเปลี่ยนเงื่อนไขในโครงสร้าง **for** เพื่อให้การวนซ้ำมีจำนวนรอบที่สอดคล้องกับจำนวนข้อมูลในอาร์เรย์อีกด้วย แม้จะเป็นการง่ายที่จะดัดแปลงโปรแกรมในลักษณะดังกล่าว ความผิดพลาดสามารถเกิดขึ้นได้ง่ายในโปรแกรมที่ยาวและซับซ้อนมากยิ่งขึ้น โดยเฉพาะอย่างยิ่งโปรแกรมที่มีการใช้งานอาร์เรย์มากกว่าหนึ่ง และการใช้งานอาร์เรย์ร่วมกับเมทริกซ์ที่เขียนขึ้นมาเพื่อประมวลผลอาร์เรย์ที่มีขนาดใดๆ ปัญหาเหล่านี้สามารถแก้ไขได้โดยใช้คำสั่งอ่านค่าคุณสมบัติ (property) ของอาร์เรย์ที่มีให้อยู่แล้วในภาษา C# โดยคุณสมบัติดังกล่าวมีชื่อว่า *Length* ซึ่งมีรูปแบบการใช้งานในรูปของนิพจน์ดังนี้

```
ArrayName.Length
```

ในที่นี้ *ArrayName* คืออาร์เรย์ที่ต้องการหาขนาด ส่วน *Length* เป็นชื่อคุณสมบัติที่ให้นิพจน์หาขนาดของอาร์เรย์ การใช้งานข้างต้นจะอยู่ในรูปของนิพจน์แบบจำนวนเต็ม ซึ่งมีค่าเท่ากับขนาดของอาร์เรย์ *ArrayName*

ตัวอย่างที่ 6.5 โปรแกรมด้านล่างคำนวณน้ำหนักเฉลี่ยของประชากรตัวอย่างจำนวน 5 คน โดยค่าน้ำหนักถูกเก็บไว้ในอาร์เรย์ *weights*

```
using System;
class AverageWeight {
    static void Main() {
        double[] weights = {65.5, 44.8, 70.0, 54.2, 77.6};
        double sum = 0.0;
        for (int i = 0; i < weights.Length; i++)
            sum += weights[i];
        Console.WriteLine("Average weight is {0:f2}",
            sum/weights.Length);
    }
}
```

ใบความรู้ที่ 6.3

คำสั่ง foreach

ภาษา C# ได้เตรียมโครงสร้าง **foreach** สำหรับการเขียนโปรแกรมแบบวนซ้ำเพื่อความสะดวกในการเข้าถึงข้อมูลในอาร์เรย์ การใช้งานมีรูปแบบดังนี้

```
foreach (DataType var in ArrayName)
    statement;
```

โดย *DataType* คือชนิดข้อมูลภายในอาร์เรย์ *var* คือชื่อตัวแปรสำหรับเก็บค่าทีละค่าจากอาร์เรย์ในการวนซ้ำแต่ละรอบ และ *ArrayName* คืออาร์เรย์ที่จะนำค่าภายในมาใช้งาน ดังนั้นคำสั่ง *statement* จึงถูกกระทำเป็นจำนวนครั้งเท่ากับจำนวนข้อมูลในอาร์เรย์

เช่นเดียวกับโครงสร้างอื่นๆ ที่ได้ศึกษาไปก่อนหน้านี้แล้ว คำสั่งที่ถูกรวนซ้ำจะมีได้เพียงคำสั่งเดียว การใช้งานมากกว่าหนึ่งคำสั่งต้องรวมคำสั่งทั้งหมดไว้ภายในวงเล็บปีกกา

```
foreach (DataType var in ArrayName) {
    statement1;
    statement2;
    :
    statementN;
}
```

ตัวอย่างที่ 6.6 โปรแกรมด้านล่างถูกดัดแปลงจากโปรแกรมในตัวอย่างที่ 6.5 โดยมีการใช้งานโครงสร้าง **foreach** แทนที่จะเป็นโครงสร้าง **for** ตามปกติ ให้สังเกตความแตกต่างที่บรรทัดที่ 6 และ 7

```
1:  using System;
2:  class AverageWeight {
3:      static void Main() {
4:          double[] weights = {65.5, 44.8, 70.0, 54.2, 77.6};
5:          double sum = 0.0;
6:          foreach (double x in weights)
7:              sum += x;
8:          Console.WriteLine("Average weight is {0:f2}",
9:              sum/weights.Length);
10:     }
11: }
```



ใบความรู้ที่ 6.4

การส่งอาร์เรย์ไปยังเมทอด

เราได้ทราบจากกิจกรรมที่ผ่านมาแล้วว่าเมทอดสามารถรับค่าจากผู้เรียกผ่านทางพารามิเตอร์ของเมทอดนั้น ๆ นอกจากการส่งค่าในรูปแบบข้อมูลพื้นฐาน (เช่น **int**, **double**, **string**) แล้ว ภาษา C# อนุญาตให้เมทอดรับพารามิเตอร์ในรูปแบบของอาร์เรย์ได้อีกด้วย ซึ่งทำได้ง่าย ๆ เพียงแค่ระบุให้พารามิเตอร์ที่รับเข้ามามีชนิดข้อมูลเป็นแบบอาร์เรย์เท่านั้นลองพิจารณาตัวอย่าง

ตัวอย่างที่ 6.7 โปรแกรมด้านล่างประกาศเมทอดชื่อ *ArraySum* ซึ่งรับพารามิเตอร์หนึ่งตัวเป็นอาร์เรย์ของ **double** จากนั้นจึงคำนวณผลรวมของค่าทั้งหมดในอาร์เรย์และส่งค่าผลรวมกลับไปยังผู้เรียก สังเกตการระบุชนิดของพารามิเตอร์เป็น **double[]** ในบรรทัดที่ 3 และการใช้งานโครงสร้าง **foreach** ซึ่งทำให้เมทอดนี้ประมวลผลอาร์เรย์ได้ทุกขนาด

```
1: using System;
2: class ArrayTest {
3:     static double ArraySum(double[] data) {
4:         double sum = 0.0;
5:         foreach (double x in data)
6:             sum += x;
7:         return sum;
8:     }
9:
10:    static void Main() {
11:        double[] myData = {1.0, 2.4, 3.6, 4.8};
12:        Console.WriteLine("Sum = {0}", ArraySum(myData));
13:    }
14: }
```

ใบความรู้ที่ 6.5

การเข้าถึงสตริงในรูปแบบอาร์เรย์

ภาษา C# อนุญาตให้เราประมวลผลข้อมูลชนิดข้อความหรือสตริง (string) เสมือนว่าข้อมูลนั้นเป็นอาร์เรย์ของอักขระโดยใช้ตัวดำเนินการต่างๆ ที่ใช้กับอาร์เรย์ได้ทันที เช่นตัวดำเนินการ [] โครงสร้าง **foreach** และคุณสมบัติ *Length* โดยมีข้อจำกัดตรงที่เราทำได้เพียงอ่านค่าอักขระ ณ ตำแหน่งต่างๆ ของข้อความได้เท่านั้น แต่ไม่สามารถเปลี่ยนแปลงส่วนหนึ่งส่วนใดของข้อความนั้นๆ ได้

ตัวอย่างที่ 6.8 ตัวอย่างต่อไปนี้รับข้อความจากผู้ใช้และรายงานจำนวนตัวอักขระ 'E' ภายในข้อความนั้นๆ

```
using System;
class CountE {
    static void Main() {
        Console.Write("Enter a string: ");
        string s = Console.ReadLine();
        int count = 0;
        for (int i = 0; i < s.Length; i++) {
            if (s[i] == 'E') count++;
        }
        Console.WriteLine("There are {0} E's in the string", count);
    }
}
```

ตัวอย่างผลการทำงาน

```
Enter a string: HELLO EVERYONE
There are 4 E's in the string
```

ตัวอย่างที่ 6.9 โปรแกรมต่อไปนี้ให้ผลการทำงานเหมือนโปรแกรมในตัวอย่างที่แล้วทุกประการ แต่มีการใช้งานโครงสร้าง **foreach** แทนการใช้โครงสร้าง **for**

```
using System;
class CountE {
    static void Main() {
        Console.Write("Enter a string: ");
        string s = Console.ReadLine();
        int count = 0;
        foreach (char c in s) {
            if (c == 'E') count++;
        }
        Console.WriteLine("There are {0} E's in the string", count);
    }
}
```



กิจกรรมที่ 7

อาร์เรย์หลายมิติ

1. จุดประสงค์ ให้ผู้เรียนสามารถ

- 1.1 ประกาศตัวแปรอาร์เรย์สองมิติ
- 1.2 เรียกใช้ตัวแปรอาร์เรย์สองมิติ
- 1.3 ใช้เมทอดเพื่อหาขนาดของอาร์เรย์

2. แนวคิด

อาร์เรย์ที่มีโครงสร้างแบบหลายมิติ (multi-dimensional array) โดยจำนวนมิติมีได้ตั้งแต่สองมิติ สามมิติ หรือมากกว่า เพื่อการจัดการข้อมูลที่ซับซ้อนมากขึ้น อย่างไรก็ตามอาร์เรย์แบบสองมิตินั้นจัดว่าเพียงพอแล้วสำหรับงานส่วนใหญ่ ในกรณีนี้ อาร์เรย์จะถูกมองอยู่ในรูปของตารางที่แต่ละช่องสามารถเก็บข้อมูลใด ๆ ได้แทนที่จะอยู่ในรูปแถวเรียงหนึ่ง การประกาศตัวแปรอาร์เรย์ เราสามารถกำหนดค่าเริ่มต้นให้กับข้อมูลแต่ละช่องพร้อมกับการสร้างอาร์เรย์ได้ทันทีโดยระบุค่าไว้ภายในเครื่องหมายปีกกา

การอ้างถึงข้อมูลในอาร์เรย์สองมิติ ต้องมีการระบุทั้งดัชนีของแถวและดัชนีของคอลัมน์เพื่อบอกตำแหน่งที่แน่นอนของช่องเก็บข้อมูลภายในอาร์เรย์ เช่นเดียวกับอาร์เรย์มิติเดียว ดัชนีของแถวและคอลัมน์นี้จะเริ่มต้นที่ค่า 0 และสิ้นสุดที่จำนวนแถวลบหนึ่งและจำนวนคอลัมน์ลบหนึ่ง

การหาขนาดของอาร์เรย์ในแต่ละมิติ (ดังเช่นการประมวลผลเมตริกซ์ (ภาษา C# มีเมทอดชื่อ *GetLength* เพื่อตรวจสอบขนาดอาร์เรย์ที่มีมิติต่าง ๆ ซึ่งมีการใช้งานในรูปของนิพจน์แบบจำนวนเต็ม

3. สื่ออุปกรณ์

3.1 ใบงาน

ใบงานที่	เรื่อง	เวลา (นาที)
7.1	แสดงค่าในเมตริกซ์	20
7.2	กำหนดค่าให้เมตริกซ์	20
7.3	เมตริกซ์ทรานสโพส	20
7.4	ดีเทอร์มิแนนต์ของเมตริกซ์	30
7.5	การคูณเมตริกซ์	30

3.2 ใบความรู้



- ใบความรู้ที่ 7.1 เรื่องอาเรย์สองมิติ
- ใบความรู้ที่ 7.2 เรื่องการหาขนาดของอาเรย์

3.3 อื่นๆ

- เครื่องคอมพิวเตอร์เท่ากับจำนวนกลุ่ม
- โปรแกรมวิชวลซีชาร์ป เอ็กเพรส

4. วิธีดำเนินการ

4.1 การจัดเตรียม

- 4.1.1 แบ่งผู้เรียนเป็นกลุ่ม กลุ่มละ 2 คน
- 4.1.2 เตรียมใบงานที่ 7.1 – 7.5 ตามจำนวนกลุ่มและใบความรู้ที่ 7.1 – 7.2 ตามจำนวนผู้เรียน

4.2 ขั้นตอนการดำเนินการ

- 4.2.1 ผู้สอนกล่าวถึงลักษณะของอาเรย์สองมิติ
- 4.2.2 ผู้เรียนแต่ละกลุ่มศึกษาใบความรู้ที่ 7.1 เรื่องอาเรย์สองมิติ และใบความรู้ที่ 7.2 เรื่องการหาขนาดของอาเรย์ จากนั้นทำใบงานที่ 7.1 เรื่องแสดงค่าในแมตริกซ์ ใบงานที่ 7.2 เรื่องกำหนดค่าให้แมตริกซ์ และใบงานที่ 7.3 เรื่องแมตริกซ์ทรานสโพส
- 4.2.3 ผู้สอนสุ่มผู้เรียนออกมานำเสนอคำตอบในใบงานที่ 7.1-7.3
- 4.2.4 ถ้ามีเวลาเหลือให้ผู้เรียนทำใบงานที่ 7.4 เรื่องดีเทอร์มิแนนต์ของแมตริกซ์ และใบงานที่ 7.5 การคูณแมตริกซ์ หรือทำนอกเวลาเสริม
- 4.2.5 ผู้เรียนและผู้สอนร่วมกันอภิปรายและสรุปเกี่ยวกับการใช้งานอาเรย์สองมิติ

5. การวัดและประเมินผล

5.1 ตรวจคำตอบจากใบงาน

6. แหล่งความรู้เพิ่มเติม

- 6.1 [http://msdn.microsoft.com/th-th/vcsharp/default\(en-us\).aspx](http://msdn.microsoft.com/th-th/vcsharp/default(en-us).aspx)

7. ข้อเสนอแนะ

-



ใบงานที่ 7.1

แสดงค่าในแมทริกซ์

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 7.1 และใบความรู้ที่ 7.2 จากนั้นสร้างโปรเจกต์และเขียนโปรแกรมจากโจทย์ที่กำหนดให้ต่อไปนี้

เราจะเขียนเมทอดชื่อ *ShowMatrix* เพื่อแสดงข้อมูลภายในแมทริกซ์ออกทางหน้าจอ โดยตัวเมทอดจะรับแมทริกซ์มาในรูปของพารามิเตอร์แบบอาร์เรย์สองมิติ จงเติมส่วนที่ขาดไปของเมทอดเพื่อให้โปรแกรมทำงานได้อย่างสมบูรณ์ตามตัวอย่างผลลัพธ์

```
using System;
class Matrix {
    static void ShowMatrix(int[,] m) {
        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}
        };
        ShowMatrix(A);
    }
}
```

ตัวอย่างผลการทำงาน

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

ใบงานที่ 7.2

กำหนดค่าให้แมทริกซ์

รายชื่อสมาชิกในกลุ่มที่.....

- | | |
|---------|---------|
| 1. | 2. |
| 3. | 4. |

ให้ผู้เรียนศึกษาใบความรู้ที่ 7.2 จากนั้นสร้างโปรเจกต์และเขียนโปรแกรมจากโจทย์ที่กำหนดให้ต่อไปนี้

โปรแกรมด้านล่างจะถามขนาดของแมทริกซ์จากผู้ใช้และเรียกเมทอดชื่อ *ReadMatrix* เพื่อสร้างแมทริกซ์ตามขนาดที่กำหนดและอ่านข้อมูลของแมทริกซ์มาทีละค่าจากผู้ใช้ จากนั้นจึงเรียกเมทอด *ShowMatrix* จากใบงานที่ 7.1 เพื่อแสดงแมทริกซ์ออกทางหน้าจอ

```
using System;
class Matrix {
    // คัดลอกเมทอด ShowMatrix จากแบบฝึกหัดที่แล้ว
    // มาปะในตำแหน่งนี้

    static int[,] ReadMatrix(int nrows, int ncols) {

        int[,] m = new int[_____, _____];
        for (int i = 0; i < nrows; i++) {
            for (int j = 0; j < ncols; j++) {
                Console.Write("Enter element[{0},{1}]: ", i+1, j+1);

                _____ = int.Parse(Console.ReadLine());
            }
        }
        return m;
    }

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

        Console.Write("How many rows? ");
        num_rows = int.Parse(Console.ReadLine());
        Console.Write("How many columns? ");
        num_cols = int.Parse(Console.ReadLine());
        A = ReadMatrix(num_rows, num_cols);
        Console.WriteLine("Matrix A is");
        ShowMatrix(A);
    }
}
```



ตัวอย่างผลการทำงาน

```
How many rows? 2
How many columns? 3
Enter element[1,1]: 9
Enter element[1,2]: 8
Enter element[1,3]: 7
Enter element[2,1]: 6
Enter element[2,2]: 5
Enter element[2,3]: 4
Matrix A is
    9    8    7
    6    5    4
```

ใบงานที่ 7.3

เมตริกซ์ทรานสโพส

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 7.2 จากนั้นสร้างโปรเจกต์และเขียนโปรแกรมจากโจทย์ที่กำหนดให้ต่อไปนี้
เพิ่มเมทอดชื่อ *TransposeMatrix* ลงในโปรแกรมจากแบบฝึกหัดที่แล้ว เพื่อคำนวณทรานสโพสเมตริกซ์ (การสลับเปลี่ยนแถวและคอลัมน์) จากเมตริกซ์ที่รับเข้ามา ตัวเมทอดจะรับเมตริกซ์ในรูปแบบอาร์เรย์สองมิติและส่งค่ากลับเป็นอาร์เรย์ตัวใหม่ที่เก็บค่าเมตริกซ์ที่ถูกทรานสโพสแล้ว

```
using System;
class Matrix {

    // คัดลอกเมทอด ShowMatrix และ ReadMatrix
    // จากแบบฝึกหัดที่แล้วมาปะในตำแหน่งนี้

    static int[,] TransposeMatrix(int[,] m) {
        int[,] mt = new int[_____, _____];
        for (int i = 0; i < m.GetLength(0); i++)
            for (int j = 0; j < m.GetLength(1); j++)
                _____;
        return mt;
    }

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

        Console.Write("How many rows? ");
        num_rows = int.Parse(Console.ReadLine());
        Console.Write("How many columns? ");
        num_cols = int.Parse(Console.ReadLine());
        A = ReadMatrix(num_rows, num_cols);
        Console.WriteLine("Matrix A is");
        ShowMatrix(A);
        Console.WriteLine("Transpose of Matrix A is");
        At = TransposeMatrix(A);
        ShowMatrix(At);
    }
}
```



ตัวอย่างผลการทำงาน

```
How many rows? 2
How many columns? 3
Enter element[1,1]: 1
Enter element[1,2]: 2
Enter element[1,3]: 3
Enter element[2,1]: 4
Enter element[2,2]: 5
Enter element[2,3]: 6
Matrix A is
  1  2  3
  4  5  6
Transpose of Matrix A is
  1  4
  2  5
  3  6
```



ใบงานที่ 7.4

ดีเทอร์มิแนนต์ของแมตริกซ์

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ดีเทอร์มิแนนต์ (determinant) ของแมตริกซ์ขนาด 2×2 ซึ่งมีสูตรการคำนวณดังนี้

$$\det(A) = (a_{11}a_{22} - a_{12}a_{21})$$

จงเขียนโปรแกรมเพื่อนำเข้าข้อมูลแมตริกซ์ขนาด 2×2 จากผู้ใช้ คำนวณดีเทอร์มิแนนต์และแสดงผลลัพธ์

ตัวอย่างผลการทำงาน

```
Enter matrix A
Enter element[1,1]: 1
Enter element[1,2]: 2
Enter element[2,1]: 3
Enter element[2,2]: 4
The determenant of A is -2
```

จากนั้นคัดลอกโปรแกรมลงในช่องว่าง โดยไม่ต้องลอกเมทอด *ShowMatrix* และ/หรือ *ReadMatrix* หากนำมาใช้โดยไม่มีการเปลี่ยนแปลง



ใบงานที่ 7.5

การคูณเมตริกซ์

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ผลลัพธ์ของการคูณเมตริกซ์ $A_{m \times n}$ และ $B_{n \times p}$ เข้าด้วยกันจะได้ผลลัพธ์เป็นเมตริกซ์ $C_{m \times p}$ ซึ่งสมาชิกแต่ละตัวจะมีค่าตามสูตร

$$C_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

จงเขียนโปรแกรมเพื่อคูณเมตริกซ์ขนาด 2×2 และแสดงผลลัพธ์ออกทางหน้าจอ

ตัวอย่างผลการทำงาน

```
Enter matrix A
Enter element[1,1]: 1
Enter element[1,2]: 2
Enter element[2,1]: 3
Enter element[2,2]: 4

Enter matrix B
Enter element[1,1]: 5
Enter element[1,2]: 6
Enter element[2,1]: 7
Enter element[2,2]: 8

Matrix A*B is
  19  22
  43  50
```



จากนั้นคัดลอกโปรแกรมลงในช่องว่าง โดยไม่ต้องลอกเมท็อด *ShowMatrix* และ/หรือ *ReadMatrix* หากนำมาใช้โดยไม่มีการเปลี่ยนแปลง



ใบความรู้ที่ 7.1

อาร์เรย์สองมิติ

การใช้งานอาร์เรย์ที่ผ่านมานั้นจำกัดอยู่เพียงโครงสร้างแบบมิติเดียวเท่านั้นซึ่งอาร์เรย์ในรูปแบบนี้จะมี การเก็บข้อมูลแบบแถวเรียงหนึ่งและมีการเข้าถึงข้อมูลผ่านดัชนีซึ่งประกอบด้วยตัวเลขเพียงตัวเดียว อาร์เรย์ แบบนี้ใช้งานได้ดีอยู่แล้วในสถานการณ์ทั่ว ๆ ไป แต่ก็ยังมีงานอีกหลายประเภทที่ต้องการการจัดการข้อมูล ที่ซับซ้อนมากกว่านั้นซึ่งต้องอาศัยโครงสร้างข้อมูลที่ซับซ้อนมากยิ่งขึ้นไปด้วย เพื่อการนี้ภาษา C# จึงอนุญาต ให้เราสร้างและใช้งานอาร์เรย์ที่มีโครงสร้างแบบหลายมิติ (multi-dimensional array) ได้โดยจำนวนมิติมีได้ ตั้งแต่สองมิติ สามมิติ หรือมากกว่า อย่างไรก็ตามอาร์เรย์แบบสองมิตินั้นจัดว่าเพียงพอแล้วสำหรับงานส่วนใหญ่ ในกรณีนี้ อาร์เรย์จะถูกมองอยู่ในรูปของตารางที่แต่ละช่องสามารถเก็บข้อมูลโคด ๆ ได้แทนที่จะอยู่ใน รูปแถวเรียงหนึ่งเหมือนที่ผ่านมาลองเปรียบเทียบอาร์เรย์สองชนิดดังแสดงในภาพ

						2	57	12	90	3	21
						1	32	88	9	10	90
						0	3	10	57	90	12
3	10	57	90	12			0	1	2	3	4

รูปทางซ้ายมือเป็นตัวอย่างของอาร์เรย์หนึ่งมิตินั้นข้อมูลภายในอาร์เรย์จึงถูกอ้างอิงโดยใช้ดัชนีที่ ประกอบด้วยตัวเลขเพียงตัวเดียวเพื่อบ่งบอกตำแหน่งของข้อมูล ตัวอย่างเช่นช่องข้อมูลที่มีดัชนีเท่ากับ 2 มีการเก็บค่า 57 เอาไว้ ส่วนรูปทางขวามือแสดงตัวอย่างของอาร์เรย์แบบสองมิติซึ่งต้องใช้จำนวนเต็มสองค่าในการระบุดัชนี เช่นช่องเก็บข้อมูลที่ตำแหน่ง (1,3) มีการเก็บค่า 10 ส่วนช่องข้อมูลที่มีดัชนี (2,4) มีการเก็บค่า 21

1. การประกาศและสร้างอาร์เรย์สองมิติ

ตัวแปรแบบอาร์เรย์สองมิติมีการประกาศตามรูปแบบด้านล่าง

```
DataType [,] ArrayName;
```

โดยที่ *ArrayName* คือชื่อของตัวแปรแบบอาร์เรย์ และ *DataType* คือชนิดข้อมูลในแต่ละช่องของอาร์เรย์

เช่นเดียวกับการใช้งานอาร์เรย์แบบหนึ่งมิติ ตัวแปรแบบอาร์เรย์สองมิตินั้นยังไม่พร้อมที่จะถูกใช้งาน จนกว่าจะมีการสร้างอาร์เรย์จริง ๆ ขึ้นมาให้ตัวแปรนั้นอ้างอิงถึง การสร้างอาร์เรย์ทำได้โดยการใช้ตัวดำเนินการ **new** ดังนี้

```
new DataType[nrows,ncols]
```


การใช้งานตัวดำเนินการ **new** ข้างต้นจะสร้างอาร์เรย์สองมิติสำหรับข้อมูลชนิด *DataType* ที่มีจำนวน *nrows* แถวและ *ncols* คอลัมน์ เนื่องจากการใช้ตัวดำเนินการ **new** ในรูปแบบข้างต้นนั้นจะอยู่ในรูปของนิพจน์ ส่วนใหญ่จึงต้องมีการนำตัวแปรแบบอาร์เรย์ที่ประกาศไว้ก่อนแล้วมารับค่าเพื่อใช้อ้างอิงค่าในอาร์เรย์ต่อไปภายหลัง ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 7.1 ประกาศตัวแปรแบบอาร์เรย์ชื่อ *students* เพื่อใช้อ้างอิงถึงอาร์เรย์สองมิติที่เก็บรายชื่อของนักเรียนตามที่นั่ง (แถว,คอลัมน์) ภายในชั้นเรียน โดยให้อาร์เรย์มีขนาด 5 แถวและ 3 คอลัมน์

```
string[,] students;  
students = new string[5,3];
```

หรือ

```
string[,] students = new string[5,3];
```

2. การสร้างอาร์เรย์สองมิติโดยกำหนดค่าเริ่มต้น

เราสามารถกำหนดค่าเริ่มต้นให้กับข้อมูลแต่ละช่องพร้อมกับการสร้างอาร์เรย์ได้ทันทีโดยระบุค่าไว้ในเครื่องหมายปีกกาเช่นเดียวกับอาร์เรย์หนึ่งมิติ อย่างไรก็ตาม เนื่องจากมิติที่เพิ่มขึ้น การกำหนดค่าเริ่มต้นจึงมีโครงสร้างที่ซับซ้อนขึ้นโดยประกอบด้วยเครื่องหมายปีกกาหลายชั้นดังแสดง

```
ArrayName = new DataType[nrows,ncols] {  
    { value(0,0), value(0,1), ..., value(0,ncols-1) },  
    { value(1,0), value(1,1), ..., value(1,ncols-1) },  
    :  
    :  
    :  
    { value(nrows-1,0), value(nrows-1,1), ..., value(nrows-1,ncols-1) }  
};
```

เนื่องจากขนาดของอาร์เรย์สามารถจะได้หากมีการกำหนดค่าเริ่มต้น ดังนั้นรูปแบบด้านล่างจึงให้ผลลัพธ์เช่นเดียวกัน

```
ArrayName = new DataType[,] {  
    { value(0,0), value(0,1), ..., value(0,ncols-1) },  
    { value(1,0), value(1,1), ..., value(1,ncols-1) },  
    :  
    :  
    :  
    { value(nrows-1,0), value(nrows-1,1), ..., value(nrows-1,ncols-1) }  
};
```

และเช่นเดียวกับอาร์เรย์หนึ่งมิติ ตัวดำเนินการ **new** สามารถจะได้โดยมีข้อจำกัดว่าเราต้องสร้างอาร์เรย์และประกาศตัวแปรแบบอาร์เรย์ภายในคำสั่งเดียวกัน

```
DataType[,] ArrayName = {  
    { value(0,0), value (0,1), ..., value(0,ncols-1) },  
    { value(1,0), value (1,1), ..., value(1,ncols-1) },  
    :  
    :  
    :  
    { value(nrows-1,0), value(nrows-1,1), ..., value(nrows-1,ncols-1) }  
};
```

ตัวอย่างที่ 7.2 สร้างอาร์เรย์สองมิติและอ้างอิงผ่านตัวแปรชื่อ A เพื่อเก็บค่าของเมตริกซ์ขนาด 4×3 ที่มีสมาชิกดังนี้

$$A = (a_{i,j})_{4 \times 3} = \begin{bmatrix} 5 & 3 & 8 \\ 2 & 6 & 10 \\ 1 & 8 & 25 \\ 12 & 3 & 30 \end{bmatrix}$$

```
int[,] A = {
    { 5, 3, 8 },
    { 2, 6, 10 },
    { 1, 8, 25 },
    { 12, 3, 30 }
}
```

3. การอ้างอิงข้อมูลในอาร์เรย์

เนื่องจากอาร์เรย์มีโครงสร้างแบบสองมิติ การอ้างอิงข้อมูลในอาร์เรย์ต้องมีการระบุทั้งดัชนีของแถวและดัชนีของคอลัมน์เพื่อบ่งบอกตำแหน่งที่แน่นอนของช่องเก็บข้อมูลภายในอาร์เรย์ เช่นเดียวกับอาร์เรย์มิติเดียว ดัชนีของแถวและคอลัมน์นี้จะเริ่มต้นที่ค่า 0 และสิ้นสุดที่จำนวนแถวลบหนึ่งและจำนวนคอลัมน์ลบหนึ่งตามลำดับ เช่นการอ้างอิงข้อมูลที่มีดัชนีแถวเป็น ri และดัชนีคอลัมน์เป็น ci ในอาร์เรย์ชื่อ $ArrayName$ จะเป็นดังนี้

```
ArrayName[ri,ci]
```

และอีกเช่นเคย การอ้างอิงถึงข้อมูลภายในอาร์เรย์ในรูปแบบนี้จะมีการใช้งานเสมือนเป็นตัวแปรโคดตัวหนึ่ง

ตัวอย่างที่ 7.3 พิจารณาเมตริกซ์ A จากตัวอย่างที่ 7.2

- แสดงค่าสมาชิกของเมตริกซ์ $A[3,2]$ (หรือ $a_{3,2}$ นั่นเอง)

```
Console.WriteLine(A[2,1]);
```

หมายเหตุ: สังเกตว่าในโปรแกรมมีการเรียกใช้ $A[2,1]$ แทนที่จะเป็น $A[3,2]$ เนื่องจากว่าการระบุตำแหน่งในอาร์เรย์สำหรับภาษา C# นั้นเริ่มต้นที่ดัชนี 0 ในขณะที่การระบุตำแหน่งในเมตริกซ์ทางคณิตศาสตร์เริ่มต้นที่ 1

- กำหนดค่า 33 ให้กับสมาชิก $a_{4,3}$

```
A[3,2] = 33;
```

ใบความรู้ที่ 7.2

การหาขนาดของอาร์เรย์

เราได้เรียนรู้จากปฏิบัติการครั้งก่อนไปแล้วว่าเราสามารถหาขนาดของอาร์เรย์หนึ่งมิติใด ๆ ได้โดยใช้คุณสมบัติ *Length* ซึ่งเป็นคุณสมบัติที่นำมาใช้ได้กับอาร์เรย์หลายมิติได้เช่นกัน อย่างไรก็ตาม การเขียนโปรแกรมสำหรับงานบางอย่างจำเป็นต้องทราบสัดส่วนของอาร์เรย์ในแต่ละมิติ (ดังเช่นการประมวลผลเมตริกซ์) ดังนั้นภาษา C# จึงเตรียมเมทอดชื่อ *GetLength* เพื่อตรวจสอบขนาดอาร์เรย์ที่มิติต่าง ๆ ซึ่งมีการใช้งานในรูปแบบของนิพจน์แบบจำนวนเต็มดังนี้

```
ArrayName.GetLength(dim_idx)
```

โดยที่ *dim_idx* แสดงหมายเลขมิติที่เราต้องการทราบขนาด สำหรับอาร์เรย์แบบสองมิตินั้นมิติหมายเลข 0 หมายถึงแถว และมิติหมายเลข 1 หมายถึงคอลัมน์ ดังนั้นนิพจน์สำหรับตรวจสอบจำนวนแถวของอาร์เรย์สองมิติจึงเขียนได้เป็น

```
ArrayName.GetLength(0)
```

และนิพจน์สำหรับตรวจสอบจำนวนคอลัมน์ของอาร์เรย์จึงเป็น

```
ArrayName.GetLength(1)
```

ตัวอย่างที่ 7.4 พิจารณาเมตริกซ์ A จากตัวอย่างที่ 7.2

- ทำให้ทุกค่าในคอลัมน์แรกของเมตริกซ์ A มีค่าเท่ากับ 1

```
for (int i = 0; i < A.GetLength(0); i++)  
    A[i,0] = 1;
```

- ทำให้ทุกค่าในแถวที่สองของเมตริกซ์ A มีค่าเท่ากับ 5

```
for (int i = 0; i < A.GetLength(1); i++)  
    A[1,i] = 5;
```

- ทำให้เมตริกซ์ A กลายเป็นเมตริกซ์ศูนย์

```
for (int i = 0; i < A.GetLength(0); i++)  
    for (int j = 0; j < A.GetLength(1); j++)  
        A[i,j] = 0;
```



แบบฝึกหัดเพิ่มเติมเรื่องอาร์เรย์สองมิติ

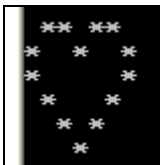
1. เขียนโปรแกรมบวกเมทริกซ์ 2 เมทริกซ์ ดังนี้

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 3 & 4 & 1 \end{bmatrix} + \begin{bmatrix} 5 & 6 & 4 \\ 1 & 3 & 2 \\ 4 & 1 & 3 \end{bmatrix}$$

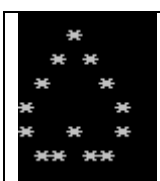
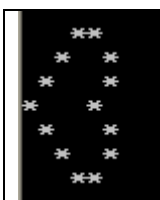
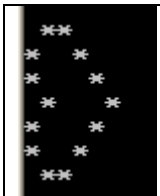
ต้องการผลลัพธ์ ดังนี้

6	8	7
6	9	9
7	5	4

2. เขียนโปรแกรมโดยใช้อาร์เรย์สองมิติเก็บข้อมูล เพื่อแสดงภาพดังนี้



3. เขียนโปรแกรมโดยใช้ข้อมูลจากตัวแปรอาร์เรย์ในข้อ 1 และให้แสดงภาพดังนี้



กิจกรรมที่ 8

การพัฒนาโปรแกรมติดต่อกับผู้ใช้แบบกราฟิก

1. จุดประสงค์ ให้ผู้เรียนสามารถ

- 1.1 อธิบายหลักการเขียนโปรแกรมเชิงวัตถุ
- 1.2 สร้างโปรเจกต์แบบ Windows Application ด้วย MS Visual C#

2. แนวคิด

หลักการของการคิดเชิงวัตถุเป็นการมองการพัฒนาระบบเหมือนกับการมองโลกแห่งความเป็นจริง คือมองสิ่งต่าง ๆ เป็นวัตถุหรือออบเจกต์ (object) ซึ่งแต่ละออบเจกต์จะมีคุณสมบัติและการทำงานเฉพาะตัวที่สามารถตรวจสอบและสั่งงานให้ออบเจกต์นั้น ๆ มีพฤติกรรมตามที่ต้องการได้

แนวคิดเชิงวัตถุเช่นนี้มีความสำคัญมากในการพัฒนาโปรแกรมในลักษณะที่เป็นวินโดวส์แอปพลิเคชันซึ่งมีการติดต่อกับผู้ใช้แบบกราฟิกแทนที่จะเป็นข้อความเพียงอย่างเดียวเหมือนที่ผ่านมา ทั้งนี้เนื่องจากส่วนประกอบที่ปรากฏอยู่บนหน้าต่างของแอปพลิเคชัน รวมถึงตัวหน้าต่างเองล้วนแต่ถูกมองเป็นออบเจกต์ทั้งสิ้น การเขียนโปรแกรมเพื่อปรับเปลี่ยนค่าคุณสมบัติของออบเจกต์เหล่านี้จะมีผลทำให้ลักษณะที่ปรากฏบนหน้าจอของออบเจกต์นั้น ๆ เปลี่ยนแปลงไป นอกจากนั้นเรายังสามารถเขียนเมทอดพิเศษกำกับให้กับออบเจกต์บางชนิด อาทิเช่น ปุ่มกด (button) ซึ่งจะถูกระบุการทำงานโดยอัตโนมัติหากผู้ใช้มีการนำเมาส์ไปคลิกที่ปุ่มนั้น เป็นต้น

3. สื่ออุปกรณ์

3.1 ใบงาน

ใบงานที่	เรื่อง	เวลา (นาที)
8.1	ออกแบบโปรแกรมเครื่องคิดเลข	30
8.2	โปรแกรมเครื่องคิดเลขอย่างง่าย	30

3.2 ใบความรู้

- ใบความรู้ที่ 8.1 เรื่องหลักการโปรแกรมเชิงวัตถุ
- ใบความรู้ที่ 8.2 เรื่องการสร้างโปรเจกต์แบบ Windows

3.3 อื่นๆ

- เครื่องคอมพิวเตอร์เท่ากับจำนวนกลุ่ม
- โปรแกรมวิซวลชีชาร์ป เอ็กเพรส



4. วิธีดำเนินการ

4.1 การจัดเตรียม

4.1.1 แบ่งผู้เรียนเป็นกลุ่ม กลุ่มละ 2 คน

4.1.2 เตรียมใบงานที่ 8.1 – 8.2 ตามจำนวนกลุ่มและใบความรู้ที่ 8.1 – 8.2 ตามจำนวนผู้เรียน

4.2 ขั้นตอนการดำเนินการ

4.2.1 ผู้สอนกล่าวถึงการโปรแกรมเชิงวัตถุ

4.2.2 ผู้เรียนแต่ละกลุ่มศึกษาใบความรู้ที่ 8.1 เรื่องหลักการโปรแกรมเชิงวัตถุ และทำใบงานที่ 8.1 เรื่องออกแบบโปรแกรมเครื่องคิดเลข

4.2.3 ผู้สอนสุ่มผู้เรียนออกมานำเสนอคำตอบในใบงานที่ 8.1

4.2.4 ผู้เรียนแต่ละกลุ่มศึกษาใบความรู้ที่ 8.2 เรื่องการสร้างโปรเจกต์แบบ Windows และทำใบงานที่ 8.2 เรื่องโปรแกรมเครื่องคิดเลขอย่างง่าย

4.2.5 ผู้เรียนและผู้สอนร่วมกันอภิปรายและสรุปเกี่ยวกับหลักการโปรแกรมเชิงวัตถุ และการสร้างโปรเจกต์แบบ Windows

5. การวัดและประเมินผล

5.1 ตรวจคำตอบจากใบงาน

6. แหล่งความรู้เพิ่มเติม

6.1 [http://msdn.microsoft.com/th-th/vcsharp/default\(en-us\).aspx](http://msdn.microsoft.com/th-th/vcsharp/default(en-us).aspx)

7. ข้อเสนอแนะ

-

ใบงานที่ 8.1
ออกแบบโปรแกรมเครื่องคิดเลข

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.

3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 8.1 แล้วออกแบบโปรแกรมสร้างเครื่องคิดเลขตามหลักของการออกแบบโปรแกรมเชิงวัตถุ



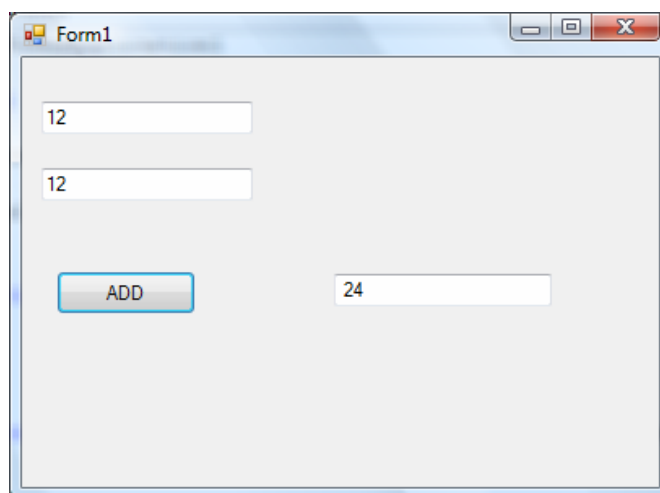
ใบงานที่ 8.2

โปรแกรมเครื่องคิดเลขอย่างง่าย

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 8.2 และปฏิบัติตามขั้นตอนในใบความรู้ จากนั้นให้เขียนโปรแกรมตามที่โจทย์กำหนดให้ต่อไปนี้



สำหรับโปรแกรมต่อไปจะเป็นการสร้างเครื่องคิดเลขอย่างง่าย โดยให้ป้อนตัวเลขเข้าไปในช่องรับข้อความทั้งสองช่อง เมื่อกดปุ่ม ADD โปรแกรมจะนำตัวเลขทั้งสองมาบวกกันแล้วแสดงผลทางช่องแสดงข้อความที่สาม

ในการพิมพ์ข้อความเข้าไปในช่องรับข้อมูลนั้น สิ่งพิมพ์เข้าไปโปรแกรมจะเก็บเป็นข้อมูลแบบสตริง ถึงแม้ว่าจะพิมพ์ตัวเลขเข้าไปโปรแกรมก็จะเก็บในลักษณะของสตริง ซึ่งไม่สามารถนำไปประมวลผลได้ ถ้าหากต้องการนำสตริงที่เป็นตัวเลขมาคำนวณจะต้องแปลงข้อความสตริงนั้นให้เป็นตัวเลขเสียก่อน

ให้ผู้เรียนสร้างโปรแกรมเครื่องคิดเลขอย่างง่ายดังนี้

1. ให้สร้างฟอร์มของโปรแกรม โดยประกอบด้วย
 - 1.1 ช่องรับข้อมูลสามช่อง
 - 1.2 ปุ่มกดหนึ่งช่อง และเปลี่ยนชื่อปุ่มเป็น ADD
2. ในการเขียนคำสั่งโปรแกรมให้ดับเบิลคลิกที่ปุ่มแล้วเขียนโปรแกรกดต่อไปนี้


```
private void button1_Click(object sender, EventArgs e)
{
    int x,y,z;
    string output;
    x = int.Parse(textBox1.Text);
    y = int.Parse(textBox2.Text);
    z = x + y;
    output = " " + z;
    textBox3.Text = output;
}
```

จากโปรแกรมจะมีการประกาศตัวแปรอะไรบ้าง

- 1.....เป็นตัวแปรชนิด
- 2.....เป็นตัวแปรชนิด
- 3.....เป็นตัวแปรชนิด
- 4.....เป็นตัวแปรชนิด

ใช้เมทอด Parse เพื่อ.....

โปรแกรมมีขั้นตอนการทำงานโดย.....



ใบความรู้ที่ 8.1

การโปรแกรมเชิงวัตถุ

แนวคิดของการโปรแกรมเชิงวัตถุหรือ OOP (Object-Oriented Programming) เป็นพื้นฐานแนวคิดที่สำคัญของนักพัฒนาโปรแกรมยุคใหม่ เนื่องจากแนวคิดการเขียนโปรแกรมแบบเดิม ๆ นั้นจะเริ่มใช้ไม่ได้หรือไม่ค่อยมีประสิทธิภาพมากนักกับงานที่มีขนาดใหญ่ และมีความซับซ้อนมาก ๆ อย่างปัจจุบัน

หลักการของการคิดเชิงวัตถุเป็นการมองการพัฒนากระบวนการเหมือนกับการมองโลกแห่งความเป็นจริง คือมองสิ่งต่าง ๆ เป็นวัตถุหรือออบเจกต์ ซึ่งแต่ละออบเจกต์จะมีคุณสมบัติและการทำงานเฉพาะตัว บางออบเจกต์สามารถมีความสัมพันธ์กับออบเจกต์อื่น ๆ ได้อีก และถ้าหากหลาย ๆ ออบเจกต์มีคุณลักษณะบางประการคล้าย ๆ กัน เราก็จะนำกลุ่มออบเจกต์เหล่านั้นมารวมให้อยู่ในกลุ่มเดียวกัน ตัวอย่างเช่นถ้าหากมีวงกลมหลาย ๆ วง แต่ละวงอาจมีรัศมีและจุดศูนย์กลางที่ต่างกัน เราสามารถมองรวมกันว่าเป็นคลาสวงกลมเพียงคลาสเดียว ส่วนวงกลมแต่ละวงจะเป็นออบเจกต์ของคลาสวงกลมนั้น โดยที่ออบเจกต์แต่ละออบเจกต์จะมีคุณสมบัติต่างกัมนั่นเอง

ในใบความรู้นี้จะแนะนำความรู้พื้นฐานและแนวคิดเชิงวัตถุที่เป็นมาตรฐานของการพัฒนาระบบงานที่ใช้กันอยู่ในปัจจุบัน

ลักษณะของซอฟต์แวร์ที่ยอมรับว่ามีประสิทธิภาพจึงมักมีความซับซ้อนค่อนข้างมากและสามารถปรับปรุงหรือนำกลับมาใช้ใหม่ได้ ความซับซ้อนดังกล่าวจะอยู่ในระดับที่น้อยคนนักที่จะสามารถเข้าใจแอปพลิเคชันทั้งหมดได้โดยอาศัยคนเพียงคนเดียว ดังนั้นแอปพลิเคชันจึงมักถูกพัฒนาขึ้นเป็นทีมเพื่อที่จะได้มีการแบ่งงานกัน

ในการพัฒนาระบบงานขึ้นมาสักระบบงานหนึ่ง ถ้ามองแบบกว้าง ๆ เราจะสามารถแบ่งการพัฒนาออกได้ใน 2 ลักษณะ คือ การพัฒนาด้วยทีมงานเป็นกลุ่มเล็ก ๆ ในลักษณะที่เรียกว่า “in-house development” ซึ่งการพัฒนาจะไม่มีรูปแบบที่แน่นอนตายตัว ขึ้นอยู่กับโปรแกรมเมอร์แต่ละคน การพัฒนาอีกลักษณะหนึ่งคือการพัฒนาในลักษณะที่เป็น “industry development” หรือการพัฒนาในระดับอุตสาหกรรม คือ การพัฒนาระบบที่มีการสร้างกรอบและมาตรฐานการทำงานเพื่อให้ผู้พัฒนายึดถือและพัฒนาระบบไปในแนวทางเดียวกัน ซึ่งการพัฒนาในลักษณะ in-house นั้นมักจะเป็นแอปพลิเคชันที่ไม่ค่อยมีความยืดหยุ่นในการที่จะนำไปเปลี่ยนแปลงแก้ไขในอนาคต เพราะการพัฒนาในลักษณะนี้ค่อนข้างไม่มีทิศทางการทำงานที่แน่นอน การพัฒนาในปัจจุบันจึงมุ่งไปที่การสร้างมาตรฐานของการพัฒนาระบบ ซึ่งแม้จะยังไม่มีมาตรฐานใดเป็นหนึ่งเดียวกันทั้งหมด แต่แนวทางก็เริ่มชัดเจนขึ้นเรื่อย ๆ ว่านักพัฒนาระบบจะยึดอะไรเป็นมาตรฐานได้

ในปัจจุบันการพัฒนาแอปพลิเคชันหรือซอฟต์แวร์ต่าง ๆ ที่ใช้กันในองค์กรต่าง ๆ มักจะไม่ได้มีผู้พัฒนาเพียงคนเดียว เนื่องจากระบบงานจะมีขนาดใหญ่ขึ้น ต้องการความเร่งด่วน ระบบมีความ

สลับซับซ้อน และต้องใช้เครื่องมือต่าง ๆ ช่วยในการพัฒนางาน ดังนั้นจะต้องมีการทำงานเป็นทีม และทีมงานอาจต้องเป็นทีมงานขนาดใหญ่ที่ทำงานในระบบเดียวกัน แต่มีการแบ่งงานออกเป็นส่วน ๆ มีการทำงานบนเครือข่ายคอมพิวเตอร์ และต้องตามให้ทันกับเทคโนโลยีที่มีการเปลี่ยนแปลงอย่างรวดเร็ว

ดังที่กล่าวมาแล้วว่าเราไม่อาจหลีกเลี่ยงความซับซ้อนได้ แต่เราสามารถจัดการกับความซับซ้อนได้ ซึ่งวิธีการจัดการกับความซับซ้อนของระบบก็มีวิวัฒนาการและมีการคิดค้นวิธีการใหม่ ๆ ขึ้นมาเรื่อย ๆ

8.1 กลวิธีในการพัฒนาแอปพลิเคชัน

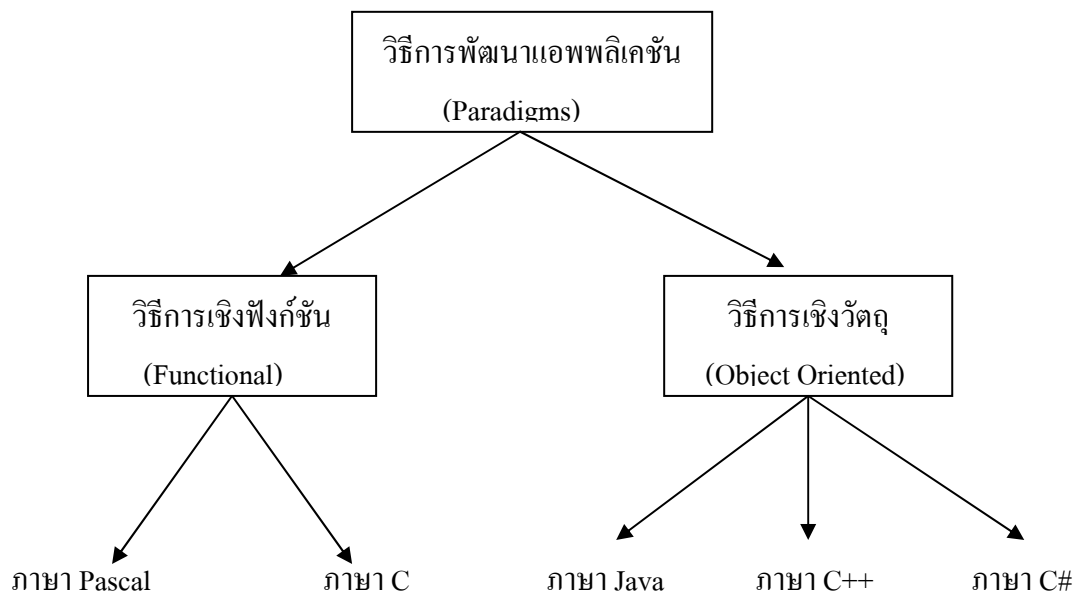
กลวิธีพัฒนาแอปพลิเคชันมีอยู่มากมาย ในที่นี้จะพิจารณาเพียง 2 วิธีที่เป็นที่นิยมเท่านั้น ได้แก่

1. วิธีการเชิงฟังก์ชัน (Function)
2. วิธีการเชิงวัตถุ (Object oriented)

วิธีการเชิงฟังก์ชัน วิธีการนี้บางครั้งเรียกว่า Algorithmic decomposition ซึ่งถือกำเนิดจากงานอุตสาหกรรม โดยมีหลักการว่าให้มองปัญหาในรูปของกระบวนการทำงาน จากนั้นให้แตกกระบวนการทำงานดังกล่าว ออกเป็นส่วนย่อย ๆ เรียกว่า “ฟังก์ชัน” แล้วจึงนำฟังก์ชันทั้งหลายมาเชื่อมโยงการทำงานเข้าด้วยกันในภายหลัง

วิธีการเชิงฟังก์ชันนับว่าเป็นวิธีการที่ได้รับความนิยมมาเป็นระยะเวลานานมาก จนกระทั่งปัจจุบันงานบางอย่างก็ยังใช้วิธีการนี้อยู่ ตัวอย่างภาษาคอมพิวเตอร์ที่ใช้วิธีการเชิงฟังก์ชัน ได้แก่ ภาษาปาสคาล ภาษาซี เป็นต้น

วิธีการเชิงวัตถุ เป็นการศึกษาและสร้างระบบงานในลักษณะโลกของความเป็นจริง โดยมองสิ่งต่าง ๆ เป็นวัตถุหรือออบเจกต์ ซึ่งออบเจกต์ต่าง ๆ จะมีความเป็นอิสระไม่ขึ้นต่อกัน แต่มีการทำงานร่วมกัน ตัวอย่างภาษาคอมพิวเตอร์ที่ใช้วิธีการเชิงวัตถุ ได้แก่ ภาษาจาวา ภาษา C++ , ภาษา C# , ภาษา SmallTalk เป็นต้น



ตารางเปรียบเทียบวิธีการเชิงฟังก์ชันกับวิธีการเชิงวัตถุ

	วิธีการเชิงฟังก์ชัน	วิธีการเชิงวัตถุ
ลักษณะทั่วไป	นำปัญหามาแตกเป็นส่วนย่อย ๆ ในรูปของกระบวนการทำงาน	มองสิ่งต่าง ๆ ในระบบเป็นออบเจกต์ซึ่งมีความเป็นอิสระต่อกัน แต่ทำงานร่วมกัน
ลักษณะการจำแนกงาน	แตกกระบวนการทำงานเป็นหน่วยย่อย ๆ เรียกว่า “ฟังก์ชัน”	จำแนกออบเจกต์แล้วแบ่งกลุ่มของออบเจกต์ตามคุณลักษณะของแต่ละออบเจกต์
ความขึ้นต่อกัน	ฟังก์ชันการทำงานต่าง ๆ จะมีลักษณะการทำงานขึ้นตรงต่อกัน มีการส่งพารามิเตอร์จากฟังก์ชันหนึ่งไปยังอีกฟังก์ชันหนึ่ง	แต่ละออบเจกต์มีความเป็นอิสระ ไม่ขึ้นต่อกัน และติดต่อกันโดยการส่งเมสเสจ (message) ถึงกัน
ขั้นตอนการทำงาน	เริ่มต้นที่การกำหนดโครงสร้างและประเภทของข้อมูล จากนั้นกำหนดฟังก์ชันการทำงานกับโครงสร้างของข้อมูลดังกล่าว	เริ่มต้นด้วยการกำหนดคุณสมบัติและพฤติกรรมให้ออบเจกต์ต่าง ๆ จากนั้นสร้างความสัมพันธ์ระหว่างออบเจกต์ว่า จะทำงานร่วมกันได้อย่างไร

8.2 ลักษณะของออบเจกต์

ออบเจกต์คือสิ่งใด ๆ ที่เราสนใจ อาจจะเป็นสิ่งที่จับต้องได้ เช่น สินค้า ลูกค้า หรืออาจจะเป็นสิ่งที่จับต้องไม่ได้ เช่น ฝ่ายต่าง ๆ ในบริษัท เป็นต้น โดยที่ออบเจกต์ต่าง ๆ จะสามารถติดต่อสื่อสารกันได้ด้วยการส่งเมสเสจ (message) ถึงกัน ดังรูปที่ 8.1



รูปที่ 8.1 การติดต่อสื่อสารกันระหว่างออบเจกต์

ออบเจกต์ 1 ออบเจกต์ 2 ไม่ว่าจะเป็นออบเจกต์แบบใดก็ตามล้วนมีลักษณะ 3 ประการดังต่อไปนี้

1. State
2. Behavior
3. Identify

State คือสถานะของความเป็นออบเจกต์หนึ่ง ๆ ซึ่งจะมีคุณสมบัติเฉพาะบางประการที่ทำให้เราทราบว่าออบเจกต์นั้นคืออะไร โดยคุณสมบัติของออบเจกต์จะขึ้นอยู่กับมุมมองของแต่ละคน ยกตัวอย่างเช่น ค้อน จะมีคุณสมบัติคือมีหัวเป็นโลหะแข็งและด้ามจับเป็นไม้ ในที่นี้หัวและด้ามจับคือคุณสมบัติหรือสถานะ (Stage) ของค้อน ซึ่งสถานะของออบเจกต์สามารถเปลี่ยนแปลงไปจากเดิมได้เมื่อมีเงื่อนไขบางประการ อาทิ เมื่อเวลาผ่านไปหัวและด้ามของค้อนอาจสึกกร่อนหรือผุพัง เป็นต้น

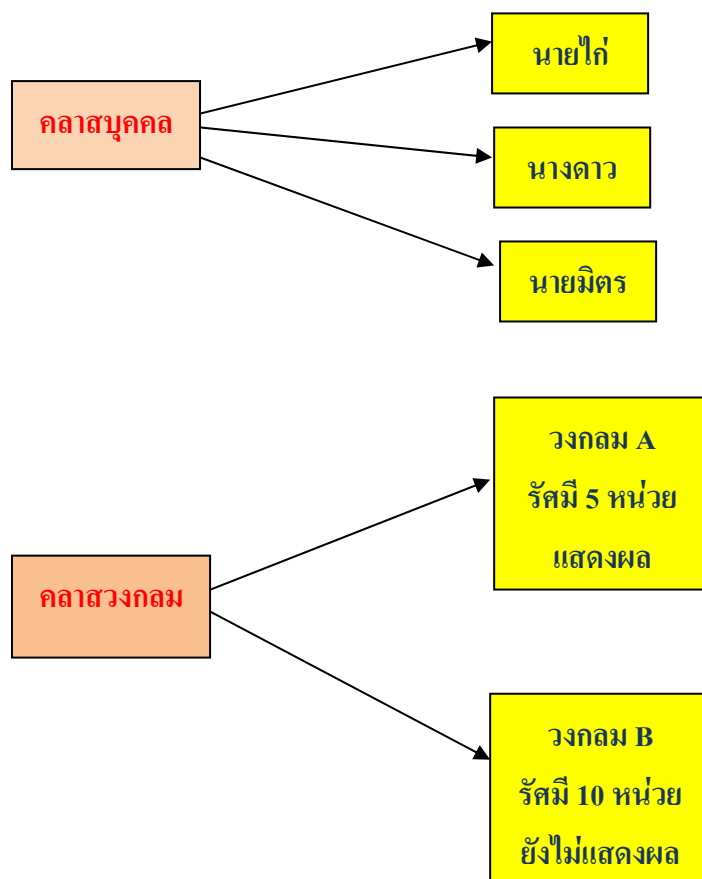
Behavior คือพฤติกรรมของออบเจกต์ ซึ่งอาจจะเป็นการส่งเมสเสจไปยังออบเจกต์หนึ่ง เป็นการส่งเมสเสจตอบกลับ หรือเป็นการกระทำบางอย่างเพื่อให้สถานะ (State) เปลี่ยนไปจากเดิม เช่น เราอาจจะเอาค้อนไปตอกตะปู ซึ่งในที่นี้ตัวเราเป็นผู้ส่งเมสเสจไปยังค้อน เป็นต้น

Identity คือคุณลักษณะบางอย่างที่ทำให้ออบเจกต์แต่ละออบเจกต์แตกต่างกัน ทำให้เรารู้ว่าเป็นคนละออบเจกต์กัน เช่น ถึงจะมีดินสอชนิดเดียวกันหลายอันในที่หนึ่ง ๆ เราก็สามารถแยกออกว่าดินสออันไหนคือดินสอของเรา โดยอาจจะดูจากคำนิหรือตำแหน่งที่เราวางดินสอเอาไว้ประจำ เป็นต้น

8.3 สร้างแบบแผนให้ออบเจกต์ด้วยคลาส

วิธีการเชิงวัตถุจะมีกลไกอย่างหนึ่งคือ “คลาส” (class) โดยคลาสถือเป็นนามธรรม (Abstract) โดยไม่สามารถนำคลาสไปดำเนินการใด ๆ ได้ คลาสนั้นเป็นการจัดกลุ่มให้แก่ออบเจกต์ต่าง ๆ ที่มีคุณสมบัติหรือพฤติกรรมบางอย่างเหมือนกัน เมื่อเวลาที่เราจะใช้งาน เราจะไม่ใช้งานคลาสตรง ๆ แต่เราจะสร้างสิ่งที่

เรียกว่า “อินสแตนท์” (instance) หรือออบเจกต์ของคลาสขึ้นมาใช้งานแทน กลไกดังกล่าวทำให้เราสามารถใช่เพียงคลาส ๆ เดียว แต่สร้างอินสแตนท์ของคลาสไปทำงานได้หลายอินสแตนท์ ซึ่งอินสแตนท์ของคลาสนี้คือออบเจกต์นั่นเอง เราจึงกล่าวว่าออบเจกต์หนึ่ง ๆ เป็นอินสแตนท์ของคลาสหนึ่ง เปรียบเทียบง่าย ๆ เสมือนว่าเรามีแปลนบ้านอยู่แปลนหนึ่ง เราก็สามารถนำแปลนบ้านนี้ไปสร้างบ้านได้อีกหลาย ๆ หลังตามความต้องการของเรา



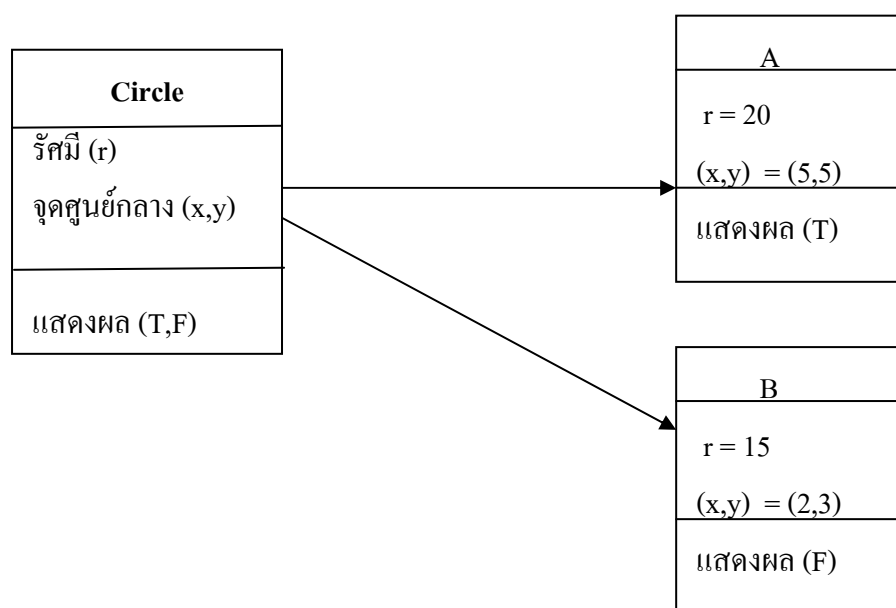
รูปที่ 8.2 แสดงตัวอย่างการสร้างออบเจกต์จากคลาส

ตัวอย่างการกำหนดคลาสให้แก่ออบเจกต์

คลาส	ออบเจกต์
บุคลากร	นางสาว สุนทริน วงศ์ศิริกุล
ปฏิกิริยาทางเคมี	เหล็กขึ้นสนิม
แม่น้ำ	เจ้าพระยา
เครื่องเขียน	ดินสอ 2B
กีฬา	บาสเกตบอล

คลาสหนึ่งคลาสสามารถมีอินสแตนซ์หรือออบเจกต์ได้หลายตัว โดยที่ออบเจกต์แต่ละตัวสามารถมีลักษณะแตกต่างกันได้ตามที่กล่าวมาแล้วในหัวข้อที่ผ่านมา ตัวอย่างเช่นถ้าหากสร้างคลาสวงกลมขึ้นมา คลาสนี้สามารถนำไปสร้างวงกลมได้หลาย ๆ วง หรือมีออบเจกต์หลาย ๆ ตัวได้ ซึ่งก็คือออบเจกต์แต่ละออบเจกต์มาจากคลาสเดียวกันก็คือคลาสวงกลมนั่นเองโดยที่วงกลมแต่ละวงจะมีรัศมีที่แตกต่างกันได้ มีเส้นผ่านศูนย์กลางที่แตกต่างกันได้

ในการพัฒนาโปรแกรมผู้พัฒนาตัวแปรภาษาสามารถรวบรวมคุณสมบัติและฟังก์ชันต่าง ๆ ของสิ่งทีคล้าย ๆ กันมารวมเป็นคลาสได้ ตัวอย่างเช่น ถ้าหากต้องการสร้างวงกลมลักษณะต่าง ๆ เช่น มีขนาดต่างกัน มีจุดศูนย์กลางตำแหน่งที่ต่าง ๆ กัน สามารถนำมารวมกันเป็นคลาสเพื่อนำไปสร้างวงกลมแบบต่าง ๆ ได้



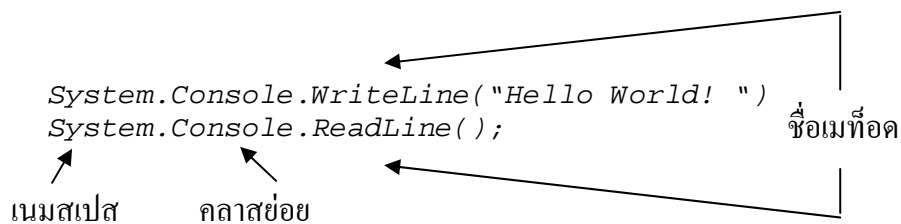
จากตัวอย่างเป็นการสร้างคลาสวงกลมและมีการสร้างออบเจกต์เป็นวงกลมขึ้นมาอีกสองวงคือ วงกลม A และวงกลม B โดยวงกลมทั้งสองวงจะมีคุณสมบัติที่ต่างกัน และให้วงกลม A แสดงผล ส่วนวงกลม B ไม่แสดงผล ถ้าหากสร้างคลาสขึ้นมาแล้วและมีการสร้างออบเจกต์สำหรับคลาสนั้นขึ้นมา จะต้องมีการระบุว่าจะให้คุณสมบัติในออบเจกต์นั้นมีลักษณะอย่างไร ในการเขียนโปรแกรมด้วยภาษา C# การระบุหรืออ้างถึงคุณสมบัติต่าง ๆ หรือเมทอด ต่าง ๆ ของออบเจกต์จะใช้เครื่องหมายจุด (.) โดยเริ่มต้นด้วยชื่อคลาสตามด้วยเครื่องหมายจุดจากนั้นตามด้วยเมทอด หรือคุณสมบัติของออบเจกต์ที่เกิดจากคลาสนั้น

รูปแบบ

ชื่อคลาส . ชื่อเมทอด [ชื่อเมทอด .]



ในการเขียนโปรแกรมด้วยภาษา C# จะมีการสร้างคลาสชื่อ *Console* ขึ้นมา โดยคลาสนี้จะรวมคลาสและเมทอด ต่าง ๆ ที่เกี่ยวข้องกับการติดต่อกับอินพุตเอาต์พุตของระบบคอมพิวเตอร์เอาไว้ ถ้าหากผู้เขียนโปรแกรมต้องการแสดงข้อมูลออกทางจอภาพจะใช้เมทอด *WriteLine()* และถ้าหากต้องการรับข้อมูลจากคีย์บอร์ดจะใช้เมทอด *ReadLine()* ในการเขียนโปรแกรมเพื่อแสดงข้อมูลและรับข้อมูลจะเขียนรหัสโปรแกรมได้ดังนี้



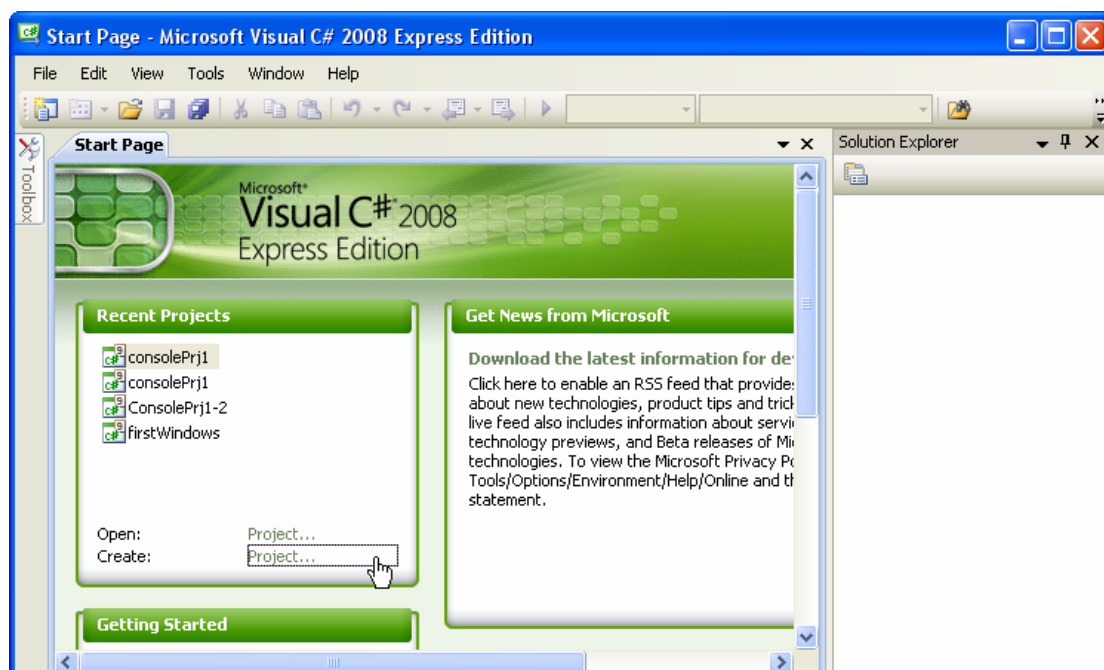
ใบความรู้ที่ 8.2

การสร้างโปรเจกต์แบบ Windows

ภาษา C# เป็นภาษาโปรแกรมเชิงวัตถุ (Object-Oriented Programming Language) ที่ถูกพัฒนาขึ้นมาโดยบริษัทไมโครซอฟต์ ในการเขียนโปรแกรมจะต้องสร้างโปรแกรมต้นฉบับมาก่อนโดยมีไฟล์ที่นามสกุลเป็น .cs เช่น prog.cs จากนั้นจะต้องแปลภาษาโดยใช้ คอมไพเลอร์ (compiler) ให้เป็นไฟล์ที่นามสกุลเป็น .exe หรือ executable

สำหรับเครื่องมือที่ใช้ในการเขียนโปรแกรมในหัวข้อนี้จะใช้โปรแกรม MS Visual Studio 2008 ซึ่งเป็นชุดโปรแกรมที่เก็บเครื่องมือในการพัฒนาไว้มากมาย แต่ในที่นี้จะใช้ MS Visual C# ในการเขียนโปรแกรม โดยการสร้างโปรเจกต์แบบ Windows ทำได้ตามขั้นตอนต่อไปนี้

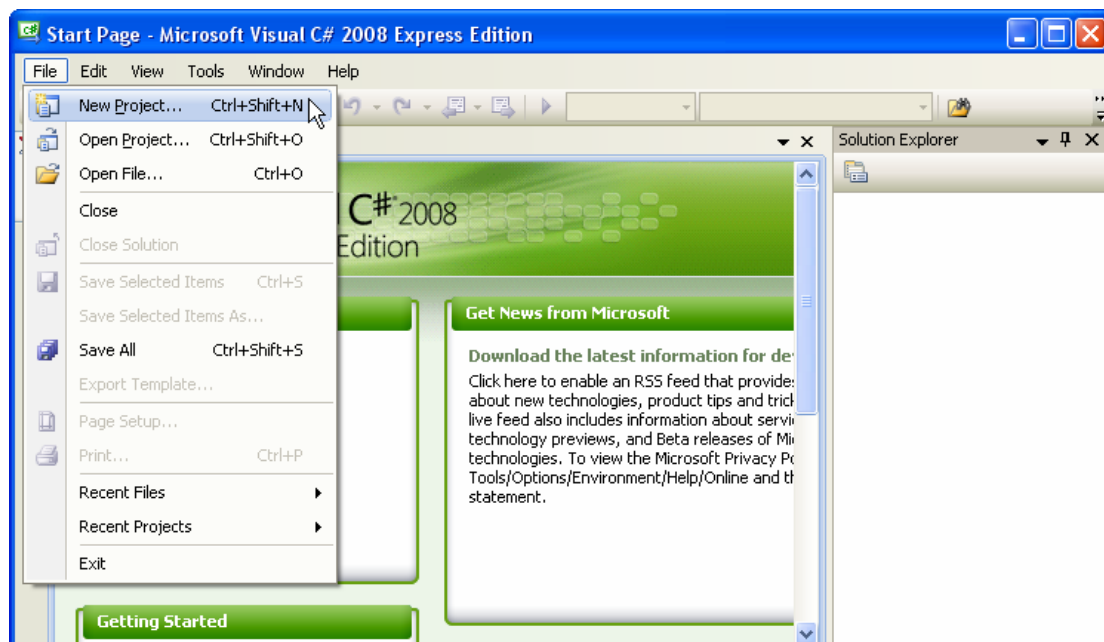
1. เมื่อเปิดโปรแกรม Microsoft Visual C# 2008 ให้เลือก Create Project.. ดังรูปที่ 8.3



รูปที่ 8.3 การเลือก Create Project..

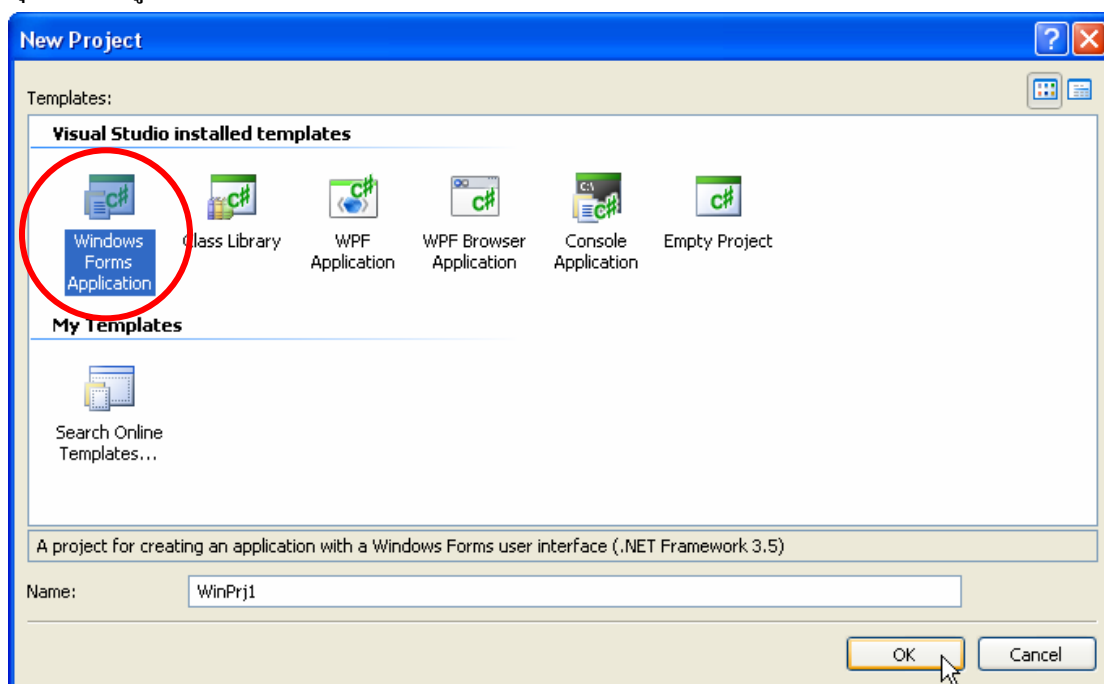
หรือเลือกจากเมนู File > New Project... ดังรูปที่ 8.4





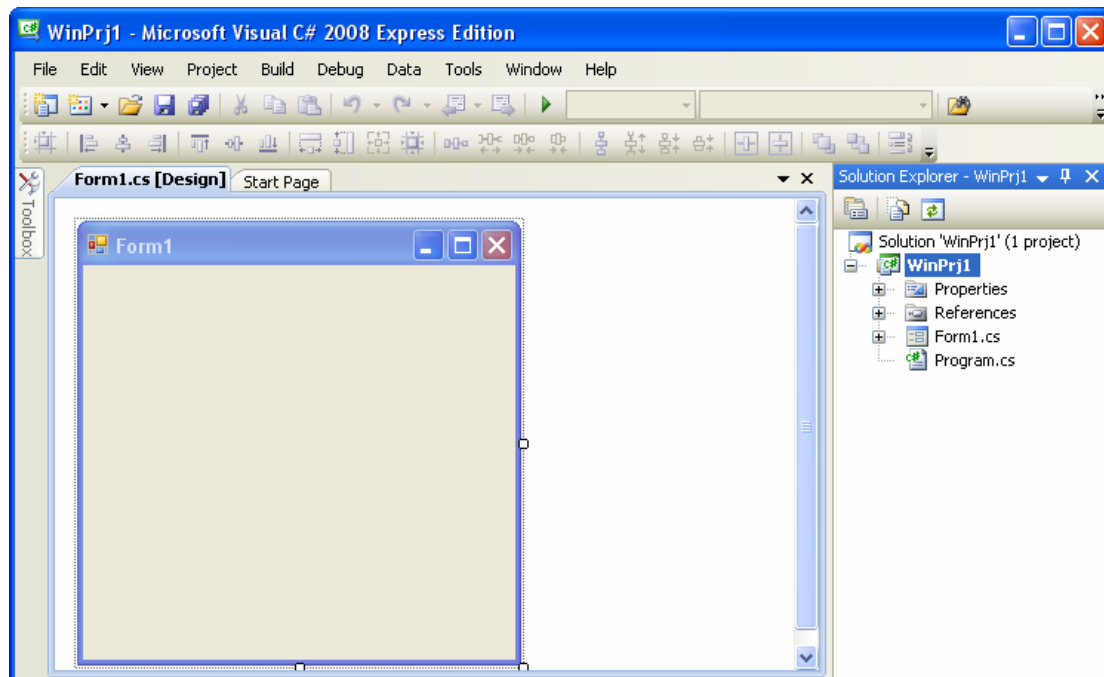
รูปที่ 8.4 เมนู File > New Project...

2. จะปรากฏหน้าต่าง New Project ให้เลือก Windows Forms Application -> ตั้งชื่อ ในช่อง Name -> คลิกปุ่ม OK ดังรูปที่ 8.5



รูปที่ 8.5 หน้าต่าง New Project

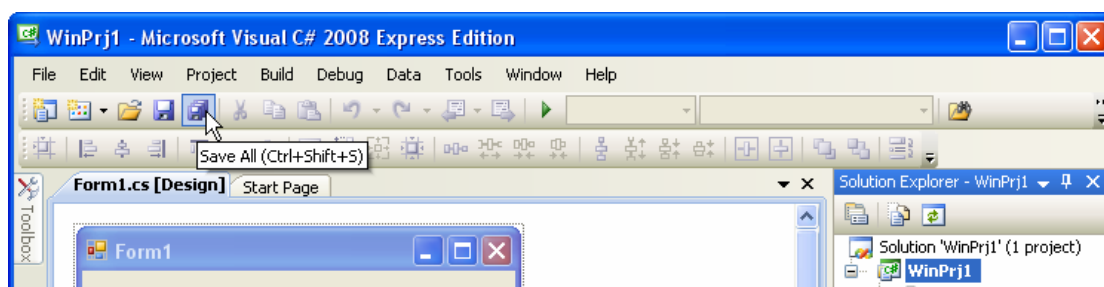
3. จะปรากฏแถบ Form1.cs[Design] สำหรับพัฒนาโปรแกรมแบบ Windows Form ดังรูปที่ 8.6



รูปที่ 8.6 แถบ Form1.cs[Design]

4. การบันทึกโปรเจกต์

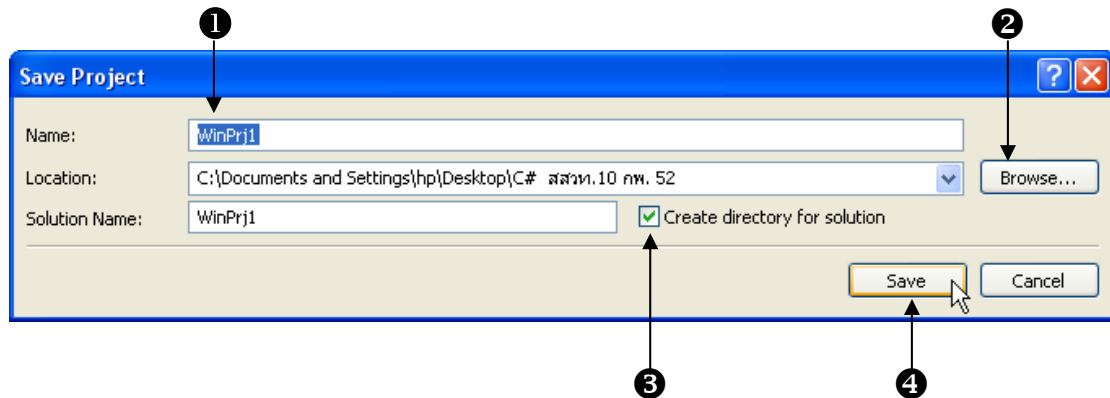
- 4.1 คลิกที่สัญลักษณ์ Save All ดังรูปที่ 8.7



รูปที่ 8.7 สัญลักษณ์ Save All

4.2 จะปรากฏกรอบโต้ตอบ Save Project

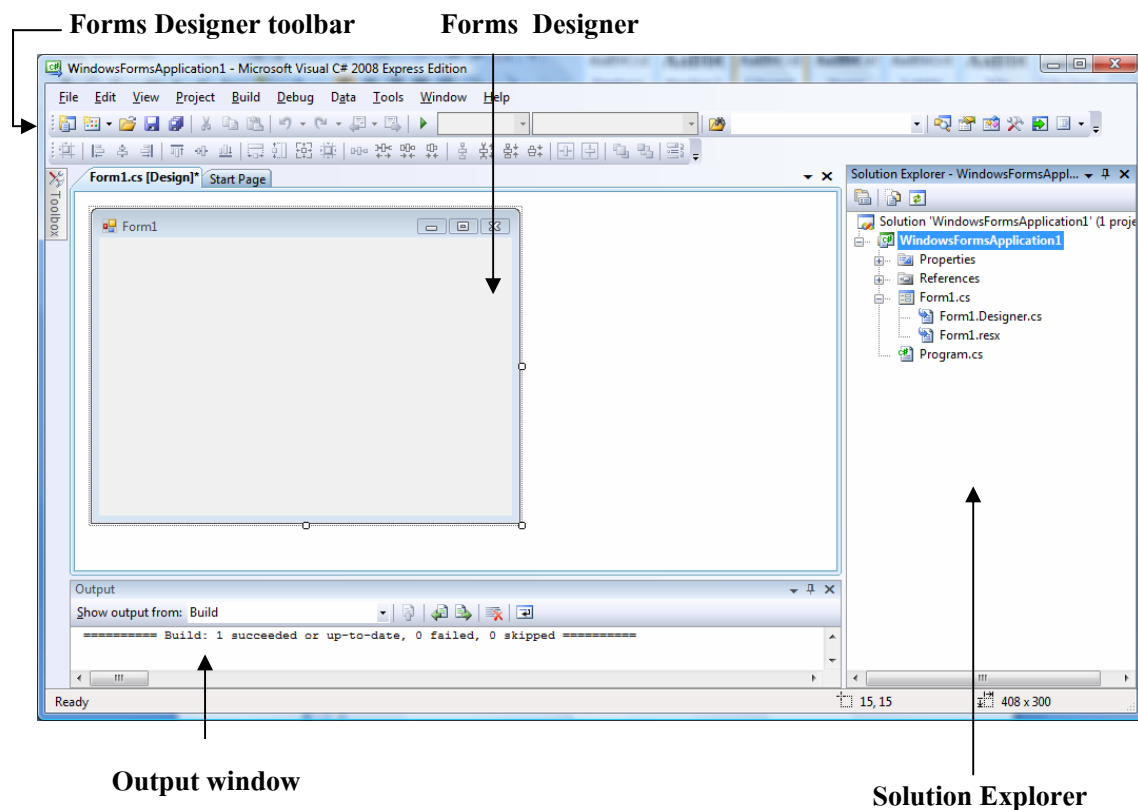
- ① ให้กำหนดชื่อในช่อง Name
- ② ในช่อง Location ให้เลือกโฟลเดอร์ที่จะเก็บโปรเจกต์ โดยคลิกเลือกที่ Browse..
- ③ เลือก Create directory for solution
- ④ เลือก Save



รูปที่ 8.8 กรอบโต้ตอบ Save Project

องค์ประกอบสภาพแวดล้อมของการพัฒนาโปรแกรมแบบ Windows

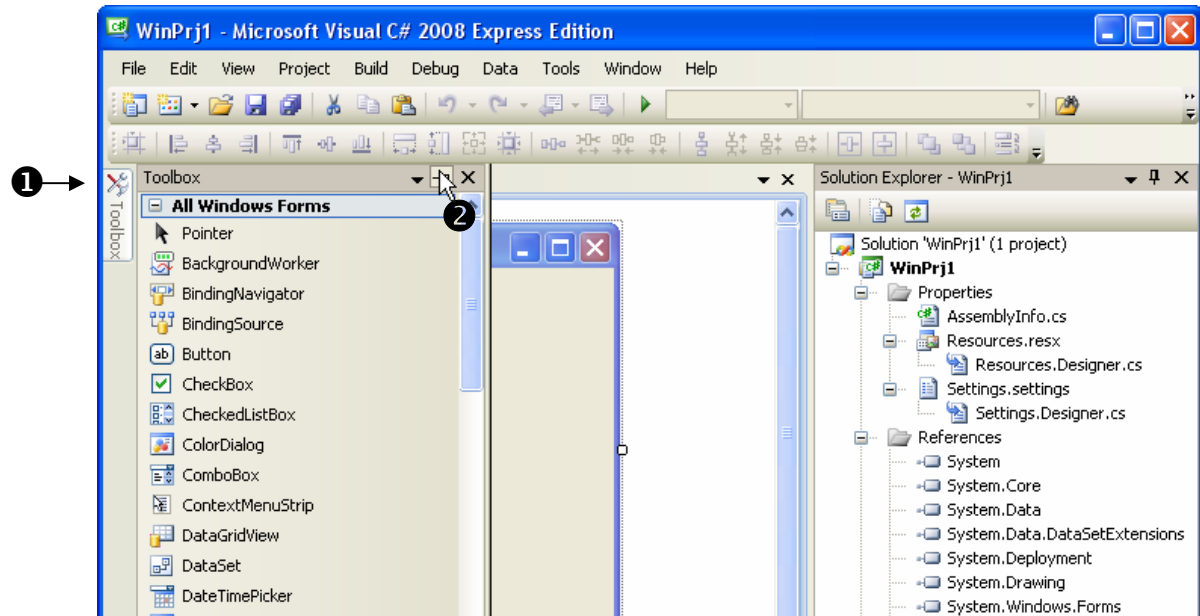
1. หน้าต่าง Design เมื่อเริ่มสร้างโปรเจกต์โปรแกรมจะแสดงหน้าต่างดังต่อไปนี้



รูปที่ 8.9 หน้าต่าง Design

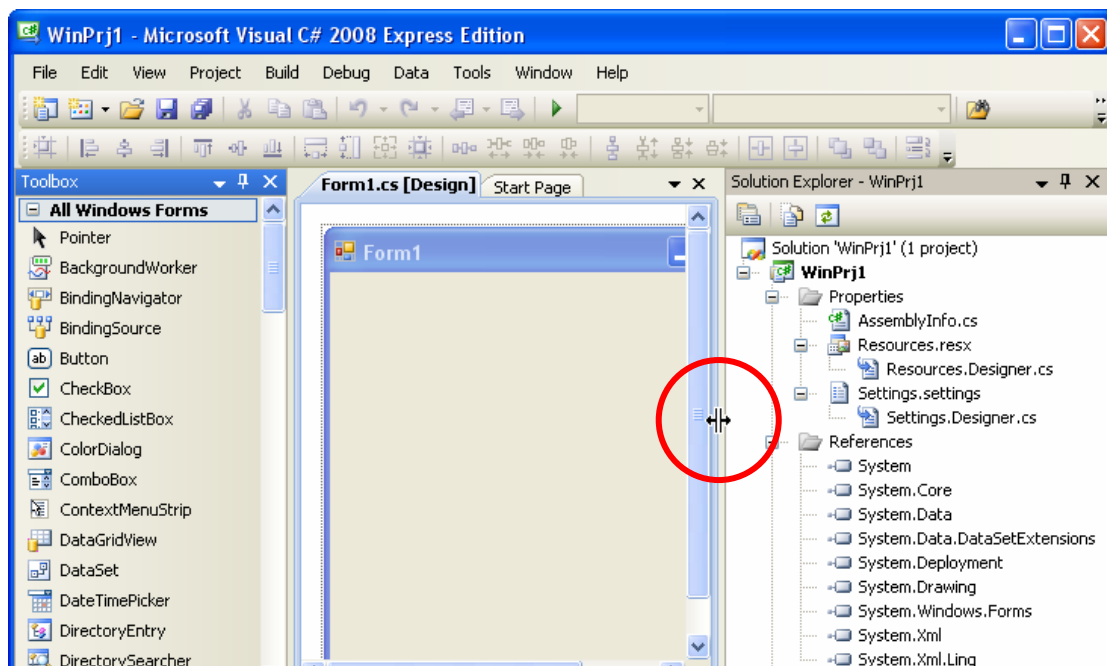
2. กล่องเครื่องมือ (Toolbox) ให้ดำเนินการเพื่อแสดงรายการเครื่องมือดังนี้

- ❶ คลิกที่แถบ Toolbox จะทำให้แถบ Toolbox ขยายออกมา
- ❷ คลิกที่ Auto Hide เพื่อให้แสดงกล่องเครื่องมือตลอดเวลา



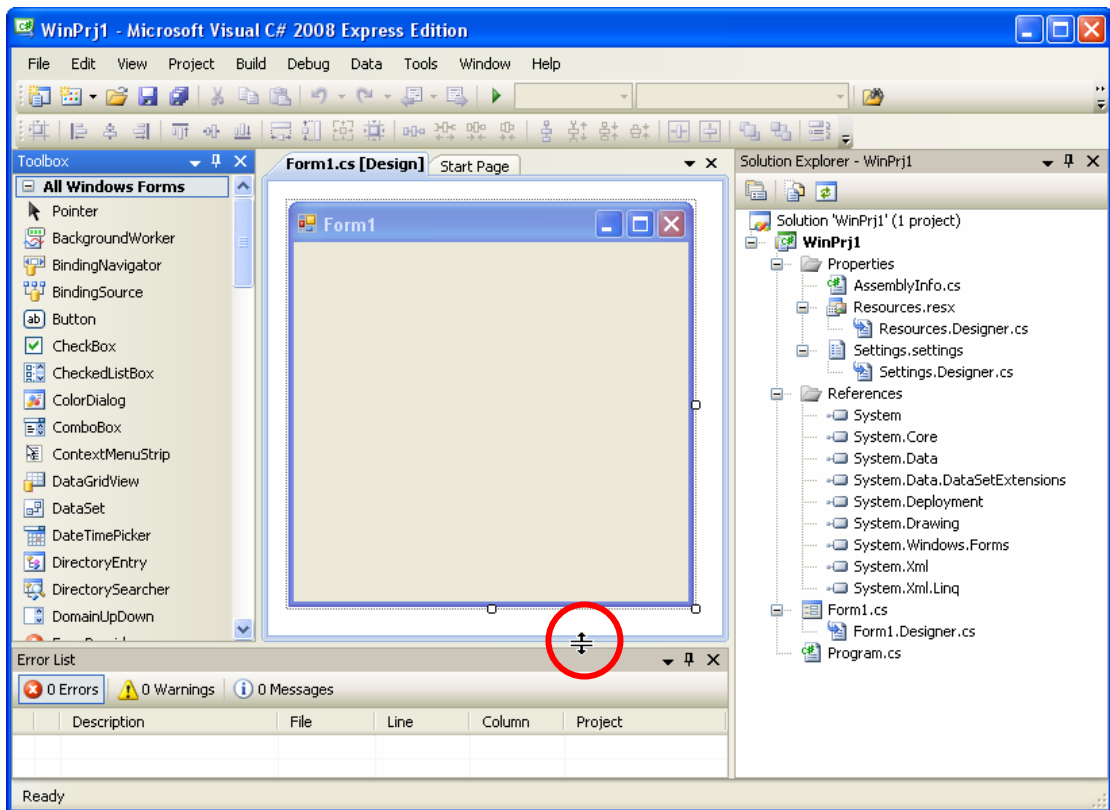
รูปที่ 8.10 การแสดงรายการกล่องเครื่องมือ

- ❸ การปรับขนาดของหน้าต่าง ๆ เพื่อความสะดวกในการใช้งาน ทำได้โดยการใช้เมาส์ลากจากเส้นกรอบของหน้าต่าง แต่ละด้านดังตัวอย่าง



รูปที่ 8.11 การปรับขนาดของหน้าต่าง

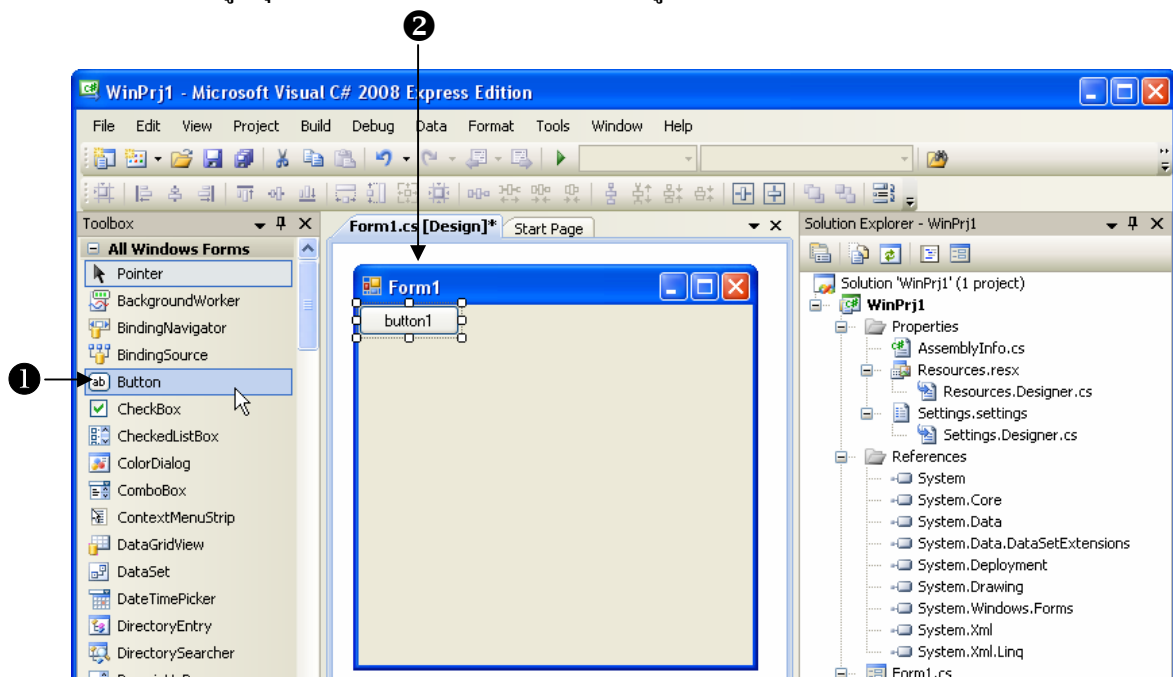




รูปที่ 8.12 ตัวอย่างการปรับขนาดหน้าต่างในแนวตั้ง

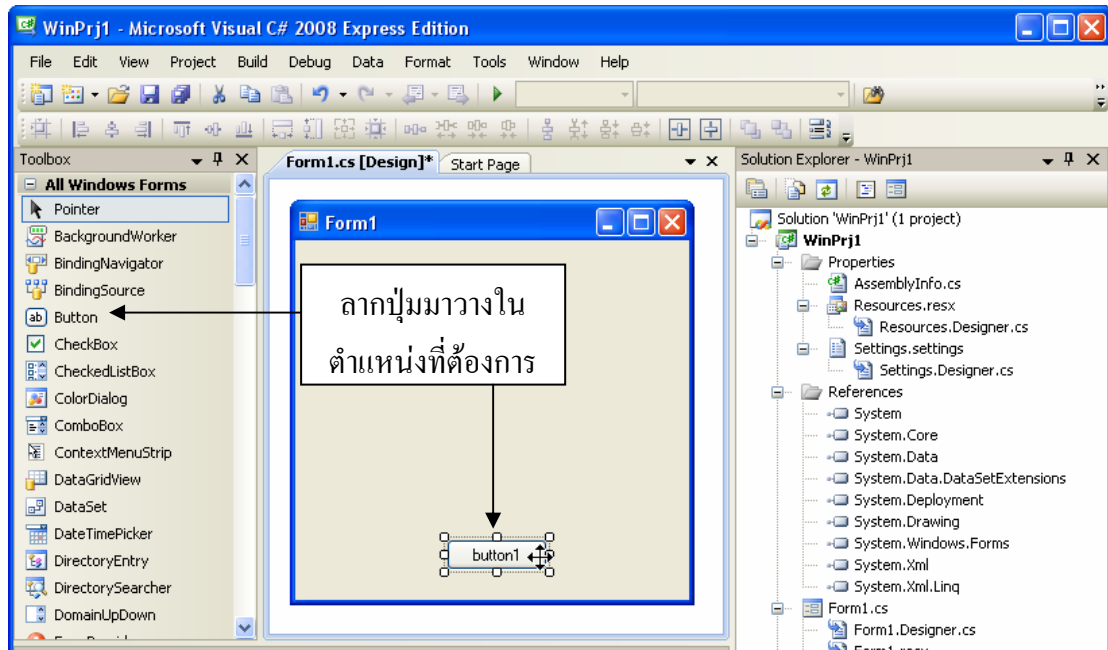
3. การเลือกเครื่องมือจาก Toolbox

3.1 การดับเบิลคลิก เช่น เมื่อดับเบิลคลิกที่เครื่องมือ Button ดังหมายเลข ❶ จะปรากฏเครื่องมือนั้นบน Form1 อยู่ที่มุมบนด้านซ้าย ดังหมายเลข ❷ ดังรูปที่ 8.13



รูปที่ 8.13 การเลือกเครื่องมือโดยการดับเบิลคลิก

3.2 การลาก-วาง การลากทำได้โดยนำเมาส์ไปลากเครื่องมือที่ต้องการจาก Toolbox มาวางในตำแหน่งที่ต้องการ ดังรูปที่ 8.14

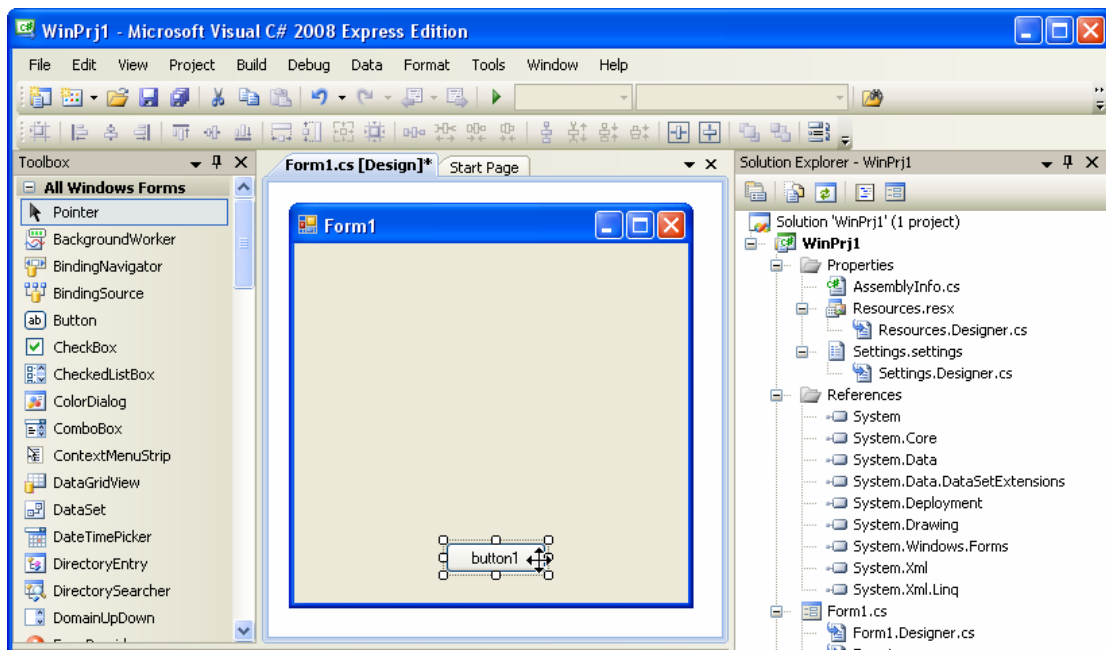


รูปที่ 8.14 การเลือกเครื่องมือโดยการลาก-วาง

4. การเขียนคำสั่ง

การเขียนคำสั่งในโปรแกรมแบบ Visual Programming จะเป็นการเขียนโปรแกรมเพื่อตอบสนองการทำงานตามเหตุการณ์ เช่น การคลิกเมาส์ มีขั้นตอนดังนี้

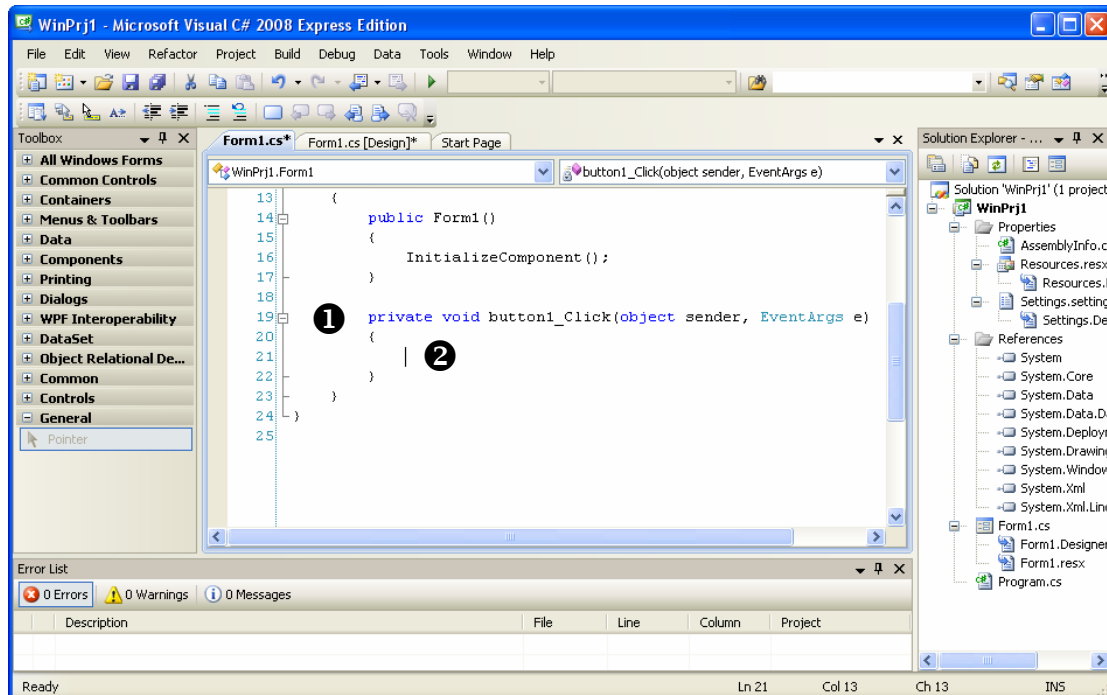
4.1 ดับเบิลคลิกที่เครื่องมือที่ต้องการสั่งงาน ที่ได้ออกแบบไว้บนฟอร์ม ดังตัวอย่างคือปุ่ม button1



รูปที่ 8.15 การดับเบิลคลิกที่เครื่องมือที่ต้องการเขียนโปรแกรมสั่งงาน

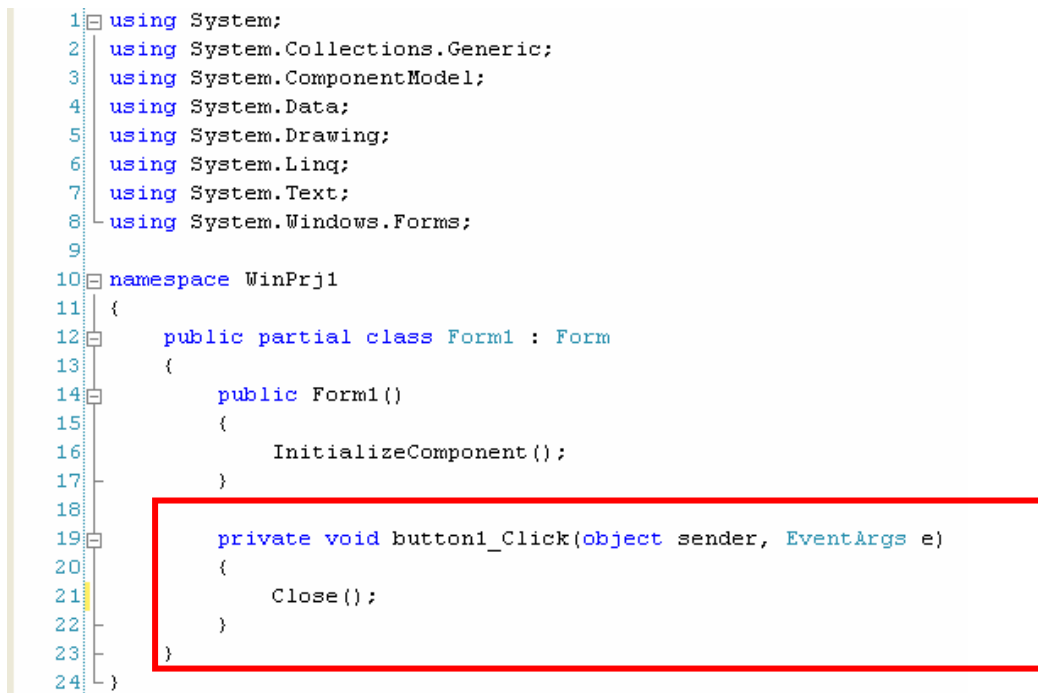


4.2 จะปรากฏหน้าต่าง Form1.cs พร้อมกับคำสั่งดังต้น ดังหมายเลข ❶ และเคอร์เซอร์จะปรากฏอยู่ตรงเมทอดที่สัมพันธ์กับเครื่องมือ ดังหมายเลข ❷ ดังรูปที่ 8.16



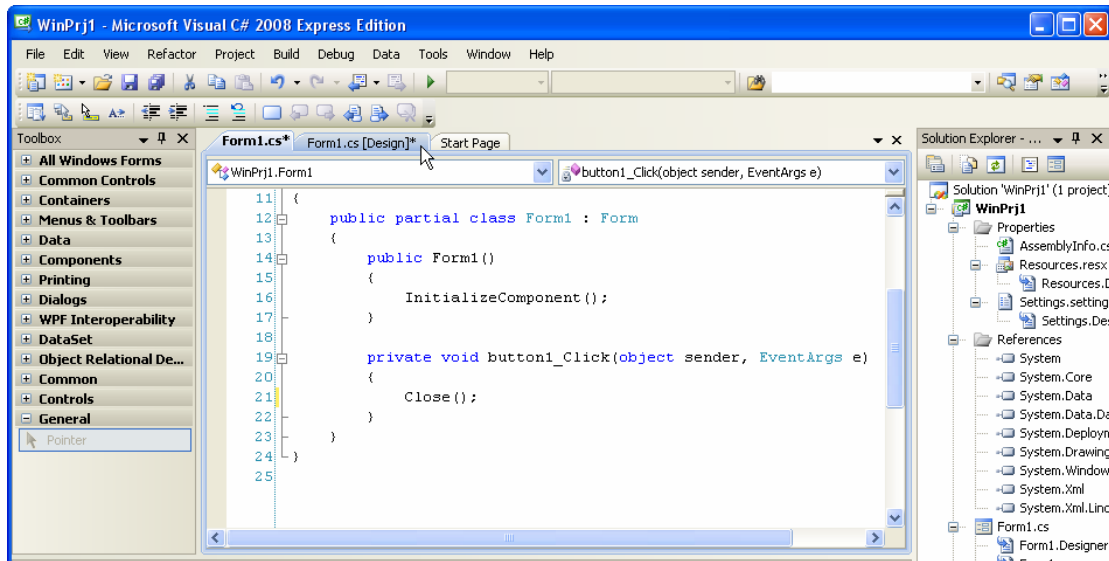
รูปที่ 8.16 หน้าต่าง Form1.cs ที่มีโค้ดโปรแกรม

สมมติว่าต้องการเขียนโปรแกรมที่สั่งให้เมื่อคลิกปุ่ม button1 แล้วจะทำการปิดโปรแกรมให้เขียนคำสั่ง Close(); ในเมทอด button1_Click() ดังรูปที่ 8.17



รูปที่ 8.17 การเขียนคำสั่ง

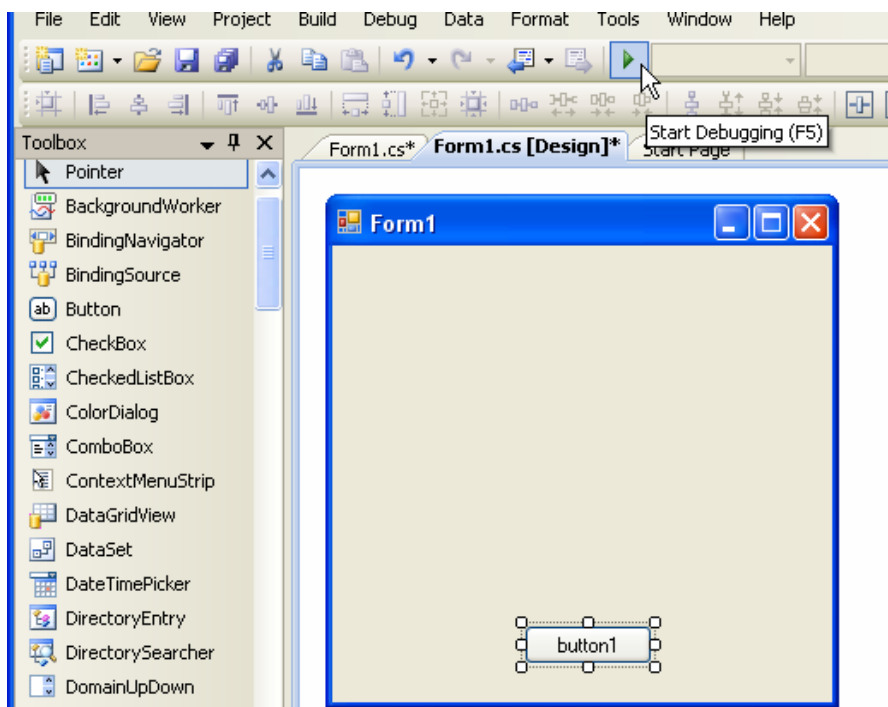
เมื่อเขียนคำสั่งเสร็จแล้ว ถ้าต้องการกลับไปยังหน้าออกแบบ สามารถทำได้โดยคลิกที่แถบ Form1.cs[Design] ดังรูปที่ 8.18



รูปที่ 8.18 แถบ Form1.cs[Design]

5. การรันโปรแกรม

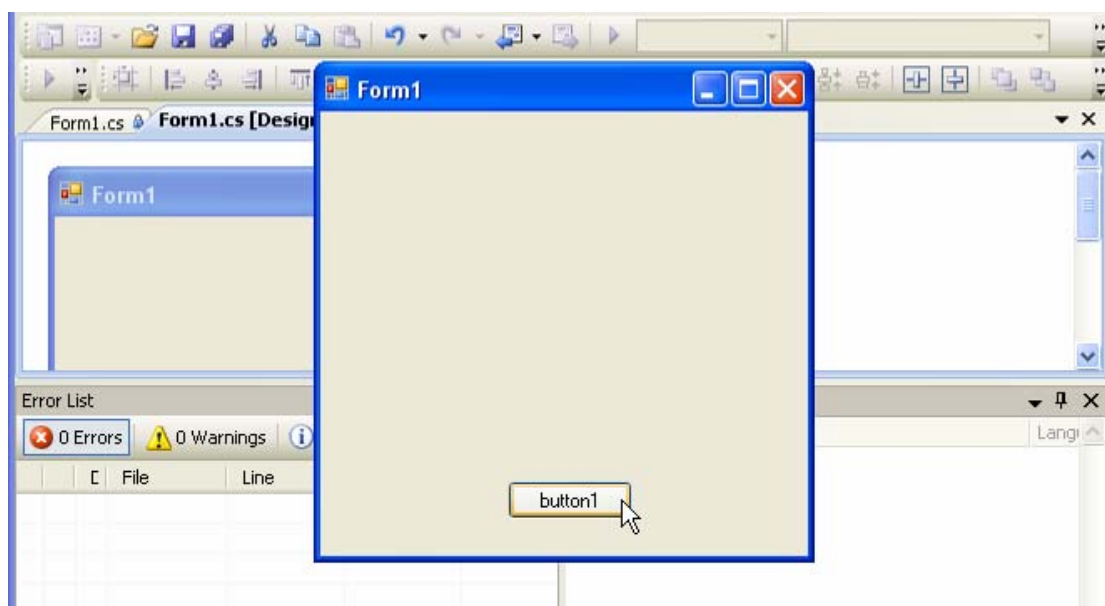
5.1 คลิกปุ่ม Start Debugging หรือกดคีย์ F5 ดังรูปที่ 8.19



รูปที่ 8.19 ปุ่ม Start Debugging



5.2 จะปรากฏโปรแกรมที่สร้างขึ้นดังรูปที่ 8.20

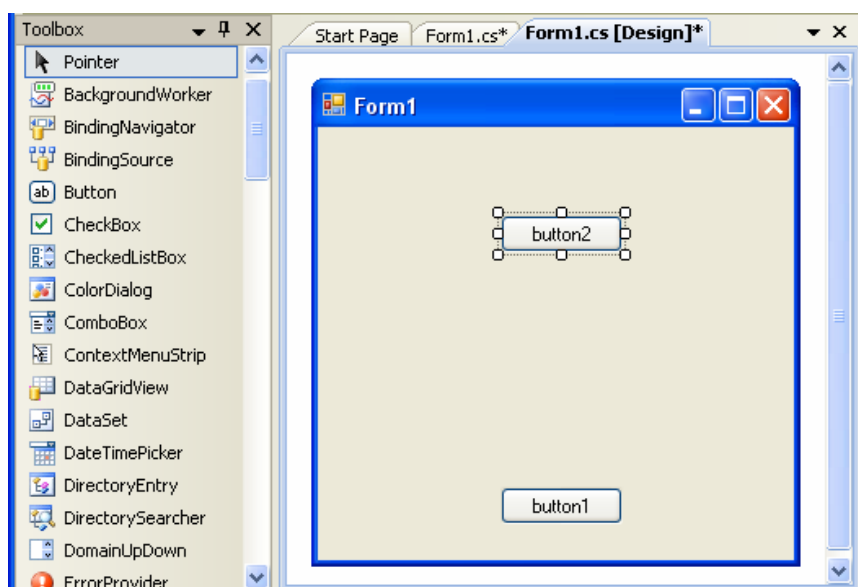


รูปที่ 8.20 โปรแกรมที่สร้างขึ้น

การออกแบบหน้าจอโดยการเปลี่ยนแปลง **Properties** เพื่อการแสดงผลที่เหมาะสม

เราสามารถกำหนดค่า Properties ให้กับเครื่องมือต่างๆ ได้ ดังเช่นตัวอย่างดังนี้

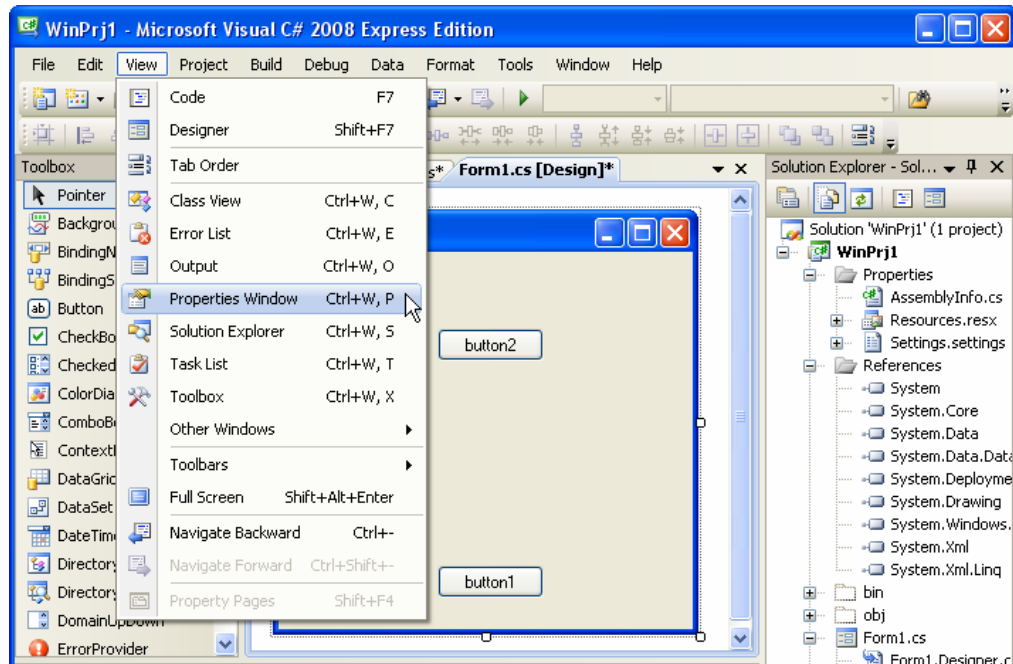
1. เลือกเครื่องมือ เช่น button2 มาวางบนฟอร์มดังรูปที่ 8.21



รูปที่ 8.21 การเพิ่มปุ่ม

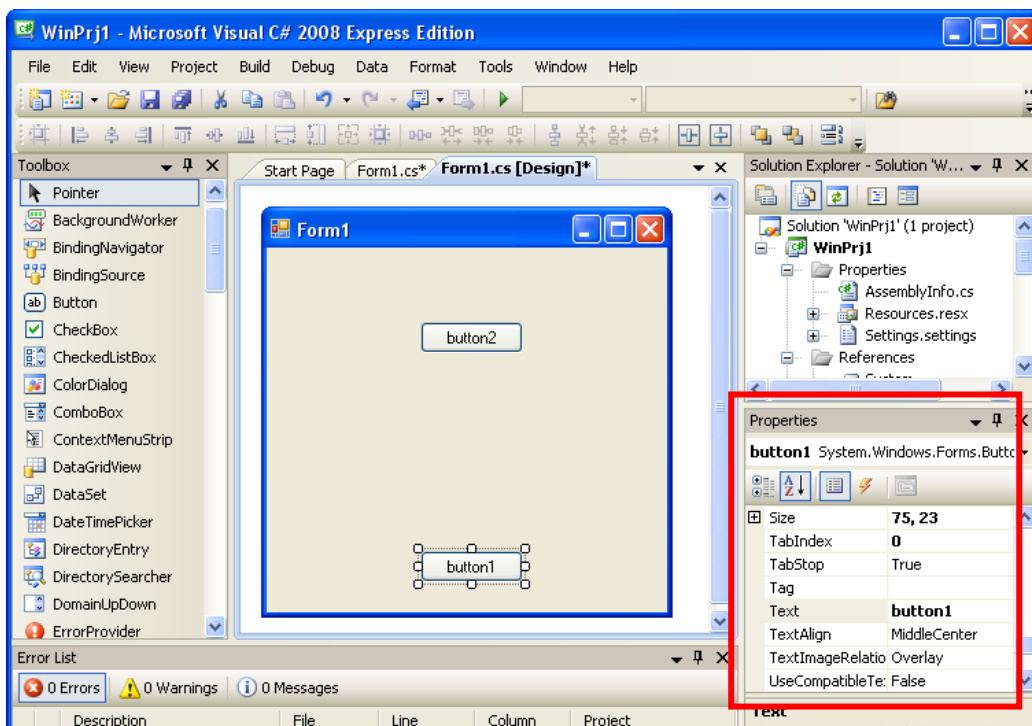
2. กำหนดให้แสดงหน้าต่าง Properties ดังนี้

2.1 คลิกที่เมนู View -> Properties Window ดังรูปที่ 8.22



รูปที่ 8.22 เมนู View -> Properties Window

2.2 จะปรากฏส่วน Properties ดังรูปที่ 8.23

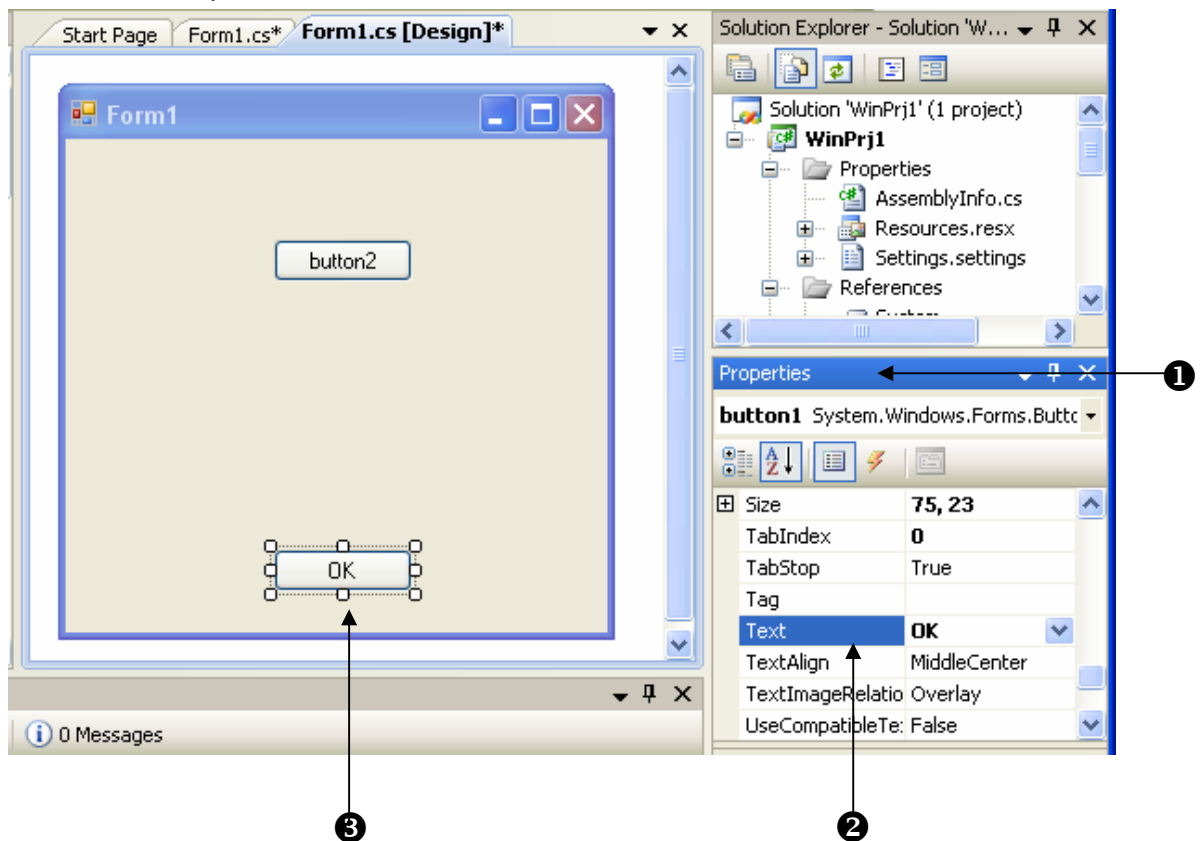


รูปที่ 8.23 ส่วน Properties



2.3 การเปลี่ยน Properties ขององค์ประกอบต่างๆ ในแถบ Design ของโปรแกรม ทำได้ดังนี้

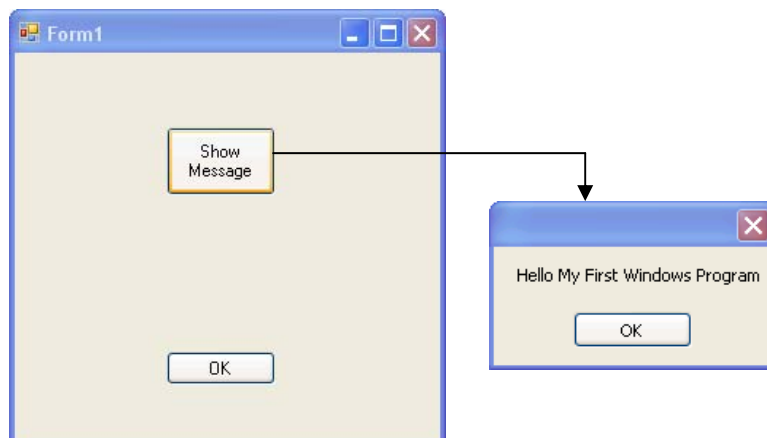
- ❶ เลือกวัตถุนั้น เช่น เลือก ปุ่ม button1 โดยที่ส่วน Properties จะเชื่อมโยงกับวัตถุนั้น
- ❷ เลือก Properties ที่ต้องการ ไขแล้วเปลี่ยนแปลงตามที่ต้องการ เช่น เปลี่ยน Text Properties ให้มีค่าเป็น OK
- ❸ จะปรากฏปุ่มที่มีข้อความ OK



รูปที่ 8.24 การเปลี่ยน Properties ของปุ่ม

ตัวอย่างการสร้างโปรแกรมเพื่อใช้งาน Message Box โดยมีขั้นตอนการทำงานดังนี้

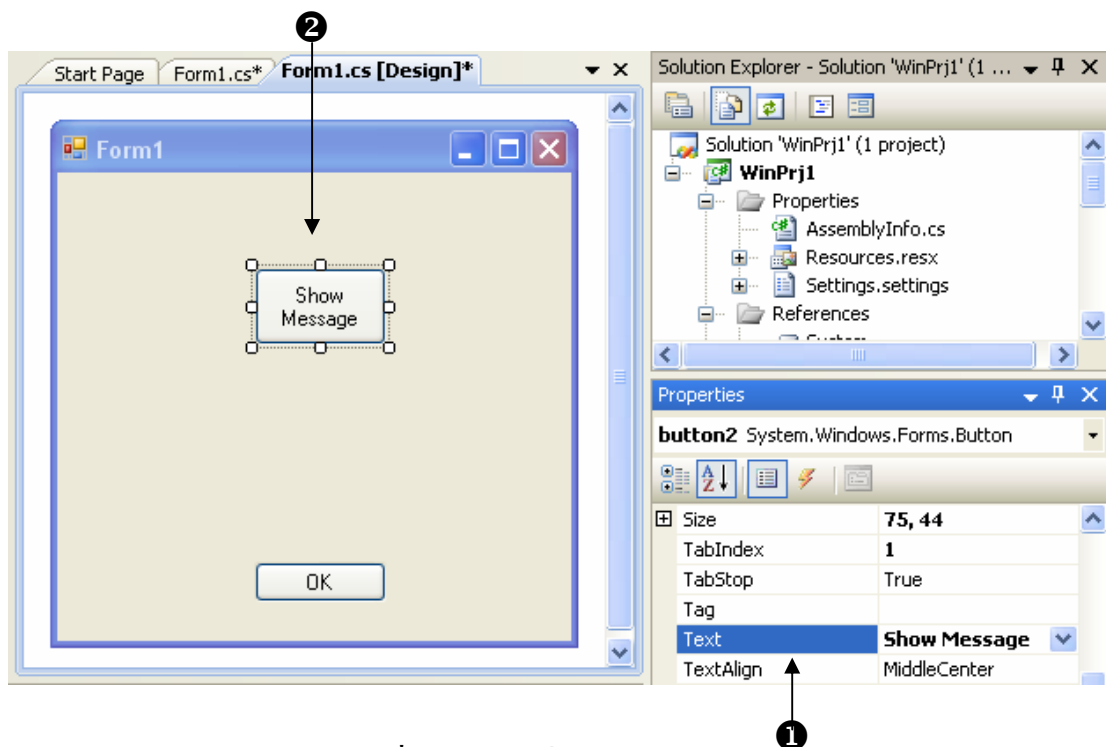
ต้องการสร้างโปรแกรมที่เมื่อทำการคลิก ปุ่ม Show Message จะแสดงข้อความใน Message Box ว่า Hello My First Windows Program ดังรูปที่ 8.25



รูปที่ 8.25 ตัวอย่างโปรแกรม

ขั้นตอนการสร้างมีดังนี้

1. คลิกเลือกที่ button2 แล้วเปลี่ยน Properties ของ button2 ในส่วน Text ให้เป็น Show Message ดังรูปที่ 8.26
2. ข้อความที่ button2 จะเปลี่ยนเป็น Show Message



รูปที่ 8.26 การเปลี่ยน Properties ของ button2

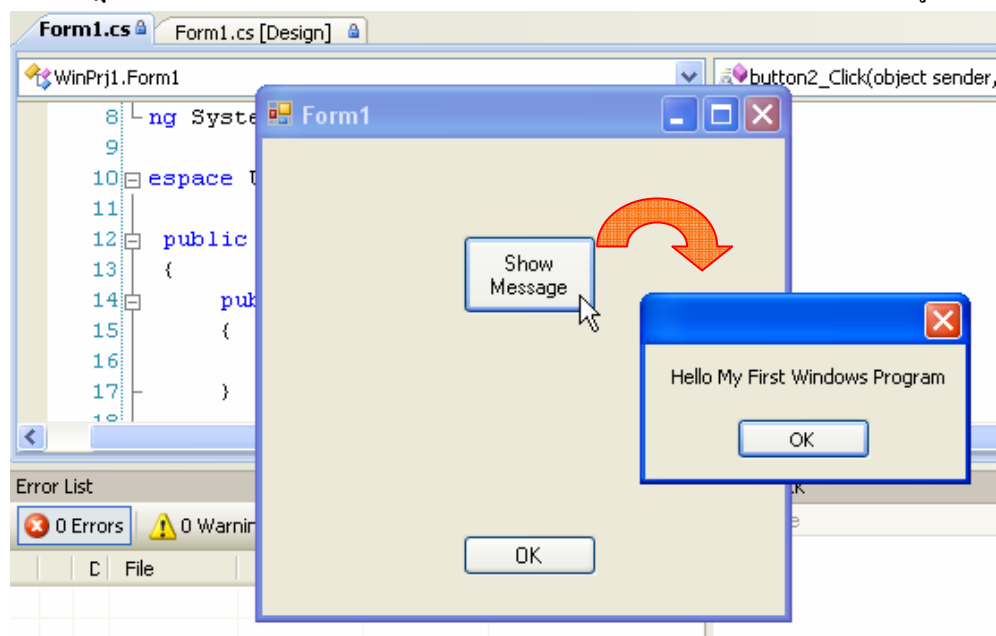


3. ดับเบิลคลิกที่ปุ่ม Show Message จะปรากฏหน้าต่าง Program.cs ขึ้นมา และเคอร์เซอร์จะปรากฏตรงเมทอด์ของปุ่มที่ถูกดับเบิลคลิก ในที่นี้คือ button2_click() เขียนคำสั่งดังรูปที่ 8.27

```
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace WinPrj1
11 {
12     public partial class Form1 : Form
13     {
14         public Form1()
15         {
16             InitializeComponent();
17         }
18
19         private void button1_Click(object sender, EventArgs e)
20         {
21             Close();
22         }
23
24         private void button2_Click(object sender, EventArgs e)
25         {
26             MessageBox.Show("Hello My First Windows Program ");
27         }
28     }
29 }
```

รูปที่ 8.27 การเขียนคำสั่งที่ปุ่ม Show Message

4. กลับไปที่หน้าจอออกแบบแบบ Form1.cs[Design]
- 4.1 บันทึกโปรแกรม
- 4.2 สัมผัสโปรแกรมจะปรากฏโปรแกรมที่ประกอบไปด้วยปุ่ม 2 ปุ่ม และเมื่อคลิกที่ปุ่ม Show Message จะปรากฏ Message Box ที่แสดงข้อความว่า Hello My First Windows Program ดังรูปที่ 8.28



รูปที่ 8.28 โปรแกรมแสดง Message Box

กิจกรรมที่ 9

พัฒนาโครงการ

1. จุดประสงค์ ให้ผู้เรียนสามารถ

พัฒนาโครงการแบบ Windows Application ด้วย MS Visual C#

2. แนวคิด

ภาษาซีชาร์ปสามารถพัฒนาเป็นแอปพลิเคชันเพื่อใช้ในการทำงานได้หลายด้าน โดยโปรแกรม MS Visual C# สนับสนุนการพัฒนาโปรแกรมประเภทที่ใช้คุณสมบัติของวินโดวส์เต็มรูปแบบ (Windows Application) ซึ่งผู้พัฒนาสามารถเขียนโปรแกรมในรูปแบบกราฟิกเพื่อสร้างงานเช่น โปรแกรมเกม โปรแกรมวาดภาพ โปรแกรมการคำนวณ ได้ง่ายและสะดวกต่อการใช้งาน

3. สื่ออุปกรณ์

3.1 ใบงาน

ใบงานที่	เรื่อง	เวลา (นาที)
9.1	พัฒนาโครงการแบบ Windows Application	360

3.2 ใบความรู้

-

3.3 อื่นๆ

- เครื่องคอมพิวเตอร์เท่ากับจำนวนกลุ่ม
- โปรแกรมวิซวลซีชาร์ป เอ็กเพรส
- โปรแกรมตัวอย่างผลงานที่สร้างด้วยภาษาซีชาร์ป
- แบบประเมินโครงการ

4. วิธีดำเนินการ

4.1 การจัดเตรียม

4.1.1 แบ่งผู้เรียนเป็นกลุ่ม กลุ่มละ 4 คน

4.1.2 เตรียมใบงานที่ 9.1 และแบบประเมินโครงการตามจำนวนกลุ่ม

4.2 ขั้นตอนการดำเนินการ



- 4.2.1 ผู้สอนเปิดโปรแกรมผลงานตัวอย่างที่พัฒนาด้วยภาษาซีชาร์ป ให้ผู้เรียนได้ศึกษา และอาจนำไปประยุกต์ใช้ในการพัฒนาผลงานของกลุ่ม
- 4.2.2 ผู้เรียนแต่ละกลุ่มลงมือพัฒนาโครงงานตามความเห็นของกลุ่ม และเขียนรายละเอียดลงใบปงานที่ 9.1
- 4.2.3 ผู้เรียนออกมานำเสนอโครงงานทุกกลุ่ม (อาจใช้การจัดแสดงนิทรรศการโครงงาน)
- 4.2.4 ผู้เรียนประเมินผลงานของกลุ่มอื่นโดยใช้แบบประเมินโครงงาน
- 4.2.5 ผู้สอนรวมคะแนน ให้รางวัลแก่กลุ่มที่มีคะแนนมากที่สุด 3 กลุ่ม
- 4.2.6 ผู้เรียนและผู้สอนร่วมกันอภิปรายและสรุป

5. การวัดและประเมินผล

5.1 แบบประเมินโครงงาน

6. แหล่งความรู้เพิ่มเติม

- 6.1 [http://msdn.microsoft.com/th-th/vcsharp/default\(en-us\).aspx](http://msdn.microsoft.com/th-th/vcsharp/default(en-us).aspx)

7. ข้อเสนอแนะ

-

ใบงานที่ 9.1

พัฒนาโครงงานแบบ Windows Application

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.

3. 4.

ให้ผู้เรียนพัฒนาโครงงานที่สร้างด้วยโปรแกรม MS Visual C# ตามความเห็นของกลุ่มและเขียนอธิบายรายละเอียดของโครงงานดังต่อไปนี้

ชื่อโครงงาน

ประเภทของโครงงาน

วัตถุประสงค์

.....
.....
.....

ลักษณะการทำงานของโปรแกรม

.....
.....
.....
.....
.....
.....
.....
.....
.....



แบบประเมินโครงการ

กลุ่มที่.....

คำชี้แจง ให้แต่ละกลุ่มให้คะแนนผลงานของกลุ่มอื่นตามหัวข้อดังต่อไปนี้ โดยให้คะแนนตามระดับด้านล่าง
หัวข้อสำหรับการประเมิน

ก. ความคิดริเริ่มสร้างสรรค์/ความใหม่ของงาน

ข. ความสวยงามและความง่ายต่อการใช้งานของโปรแกรม

ค. ความซับซ้อนทางเทคนิค

ง. ความถูกต้องในการทำงาน

จ. คุณภาพของงานในภาพรวม

ลำดับ	ชื่อกลุ่ม	● ความคิดริเริ่มสร้างสรรค์	● ความสวยงามและความง่ายต่อการใช้งานของโปรแกรม	● ความซับซ้อนทางเทคนิค	● ความถูกต้องในการทำงาน	● คุณภาพของงานในภาพรวม	คะแนนรวม
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

ระดับการให้คะแนน

	ดีมาก	ดี	ปานกลาง	สมควรปรับปรุง	สมควรปรับปรุงอย่างมาก	ไม่ผ่าน
ระดับ	A	B	C	D	E	F
คะแนน	5	4	3	2	1	0



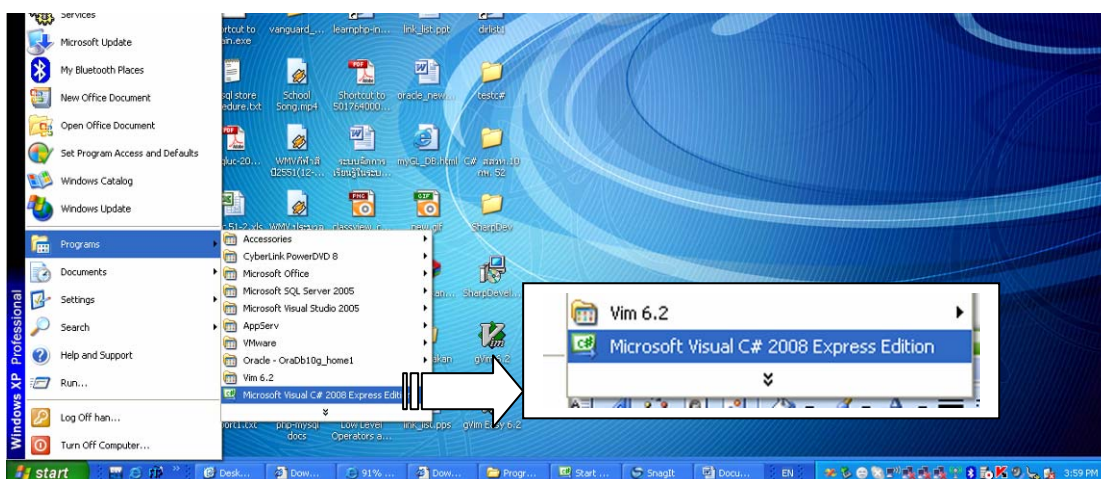
ภาคผนวก



การใช้งานโปรแกรม Microsoft Visual C# 2008 Express

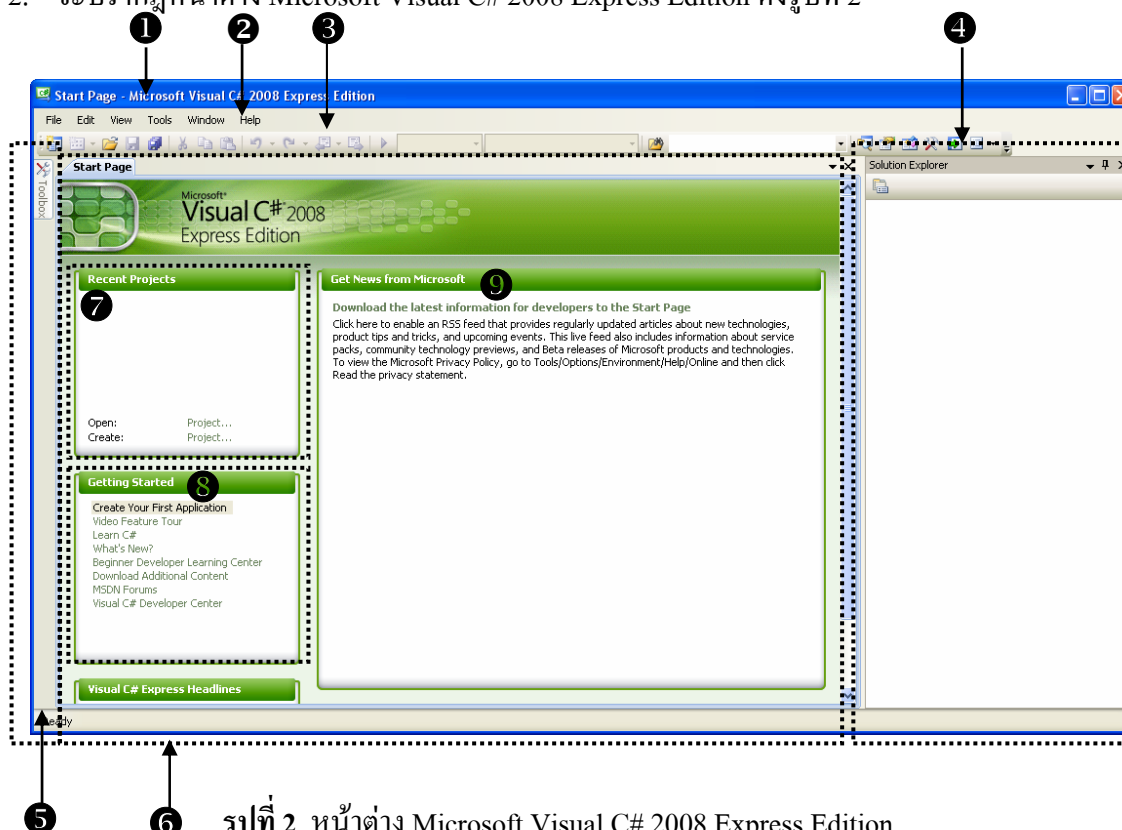
Microsoft Visual C# 2008 Express Edition เป็นโปรแกรมที่อยู่ในชุดโปรแกรมของ MS Visual Studio 2008 ซึ่งเป็นชุดโปรแกรมที่เก็บเครื่องมือในการพัฒนาไว้มากมาย การเรียกใช้งานโปรแกรมทำได้ตามขั้นตอนต่อไปนี้

1. คลิก Start -> Programs -> Microsoft Visual C# 2008 Express Edition



รูปที่ 1 การเรียกใช้งานโปรแกรม

2. จะปรากฏหน้าต่าง Microsoft Visual C# 2008 Express Edition ดังรูปที่ 2



รูปที่ 2 หน้าต่าง Microsoft Visual C# 2008 Express Edition

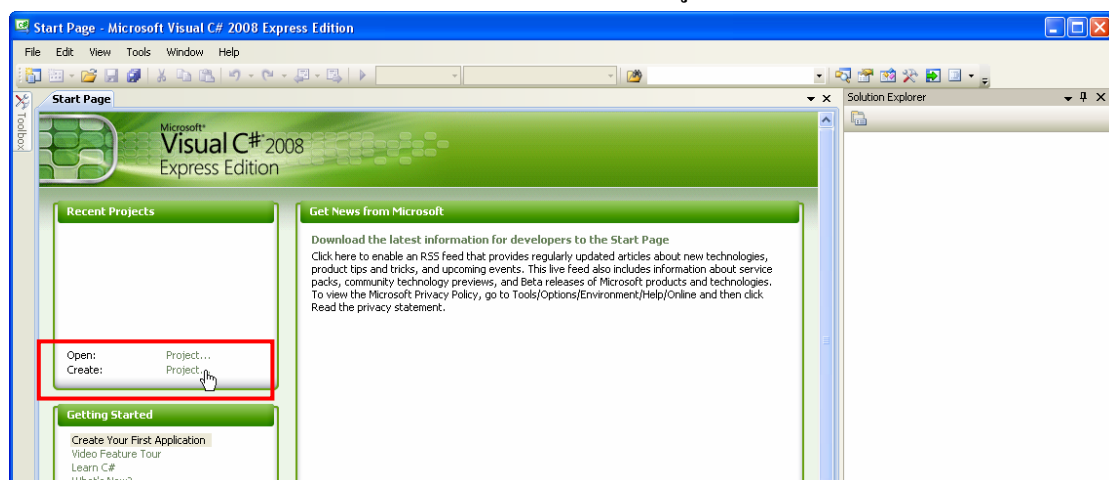
ส่วนประกอบของ Microsoft Visual C# 2008 Express Edition มีดังนี้

- 2.1 แถบชื่อ(Title Bar) เป็นส่วนที่แสดงชื่อของหน้าต่างที่ทำงานอยู่เช่นชื่อโปรเจกต์
- 2.2 แถบเมนู(Menu Bar) เป็นแถบแสดงรายการคำสั่งของโปรแกรม เช่นเมนู File , Edit , View เป็นต้น
- 2.3 แถบเครื่องมือ(Tool Bar) เป็นแถบรวมสัญลักษณ์(Icon) ที่ใช้แทนรายการคำสั่งของโปรแกรม
- 2.4 หน้าต่าง Solution Explorer เป็นหน้าต่างแสดงรายการของไฟล์และองค์ประกอบต่างๆ ที่รวมกันเป็นโปรเจกต์
- 2.5 กล่องเครื่องมือ(Tool Box) เป็นส่วนที่เก็บเครื่องมือที่ใช้สร้างโปรเจกต์ ส่วนกล่องเครื่องมือนี้จะเปลี่ยนแปลงไปตามประเภทของโปรเจกต์ที่ผู้เขียนโปรแกรมเลือก
- 2.6 หน้าต่าง Start Page เป็นหน้าต่างหลัก และเป็นหน้าต่างแรกที่เกิดขึ้นเมื่อเปิดโปรแกรม Microsoft Visual C# 2008 Express Edition ขึ้นมา โดยภายในจะมีหน้าต่างย่อยอื่นๆ
- 2.7 หน้าต่าง Recent Project เป็นหน้าต่างย่อยภายในหน้าต่าง Start Page เป็นส่วนที่ใช้สร้างโปรเจกต์ใหม่ หรือเปิดโปรเจกต์เก่าขึ้นมาทำงาน
- 2.8 หน้าต่าง Getting Started เป็นส่วนที่ใช้เรียนรู้เกี่ยวกับการพัฒนาซอฟต์แวร์โดย Microsoft Visual C#
- 2.9 หน้าต่าง Gets New from Microsoft เป็นส่วนที่ใช้ดาวน์โหลดเครื่องมือใหม่ ๆ จากไมโครซอฟต์

การเริ่มสร้างโปรเจกต์ใหม่ แบบ Console Application

Console Application เป็นการเขียนโปรแกรมที่เหมาะสมสำหรับผู้เริ่มต้นศึกษาการพัฒนาโปรแกรม สามารถสร้างได้ดังนี้

1. ในส่วน Recent Project ที่ Create คลิกเลือก Project ดังรูปที่ 3

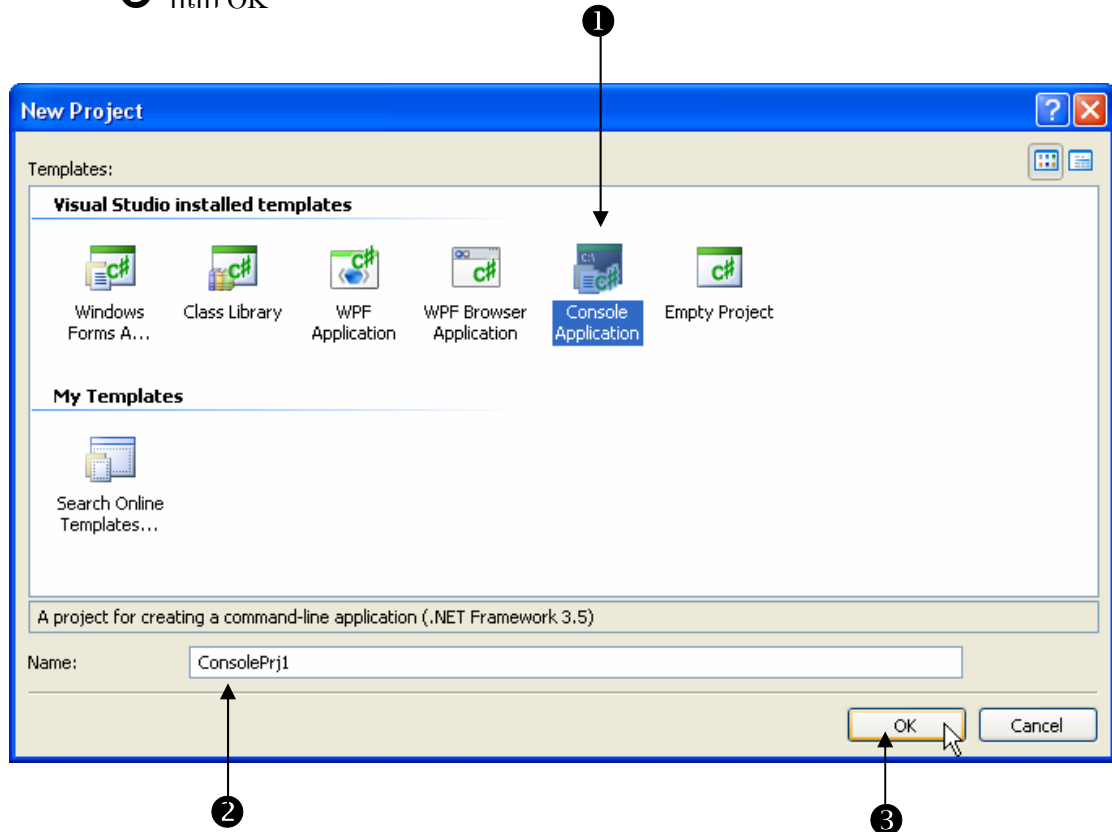


รูปที่ 3 การสร้างชิ้นงานใหม่



2. จะปรากฏกรอบโต้ตอบ New Project ดังรูปที่ 4 ให้ดำเนินการดังนี้

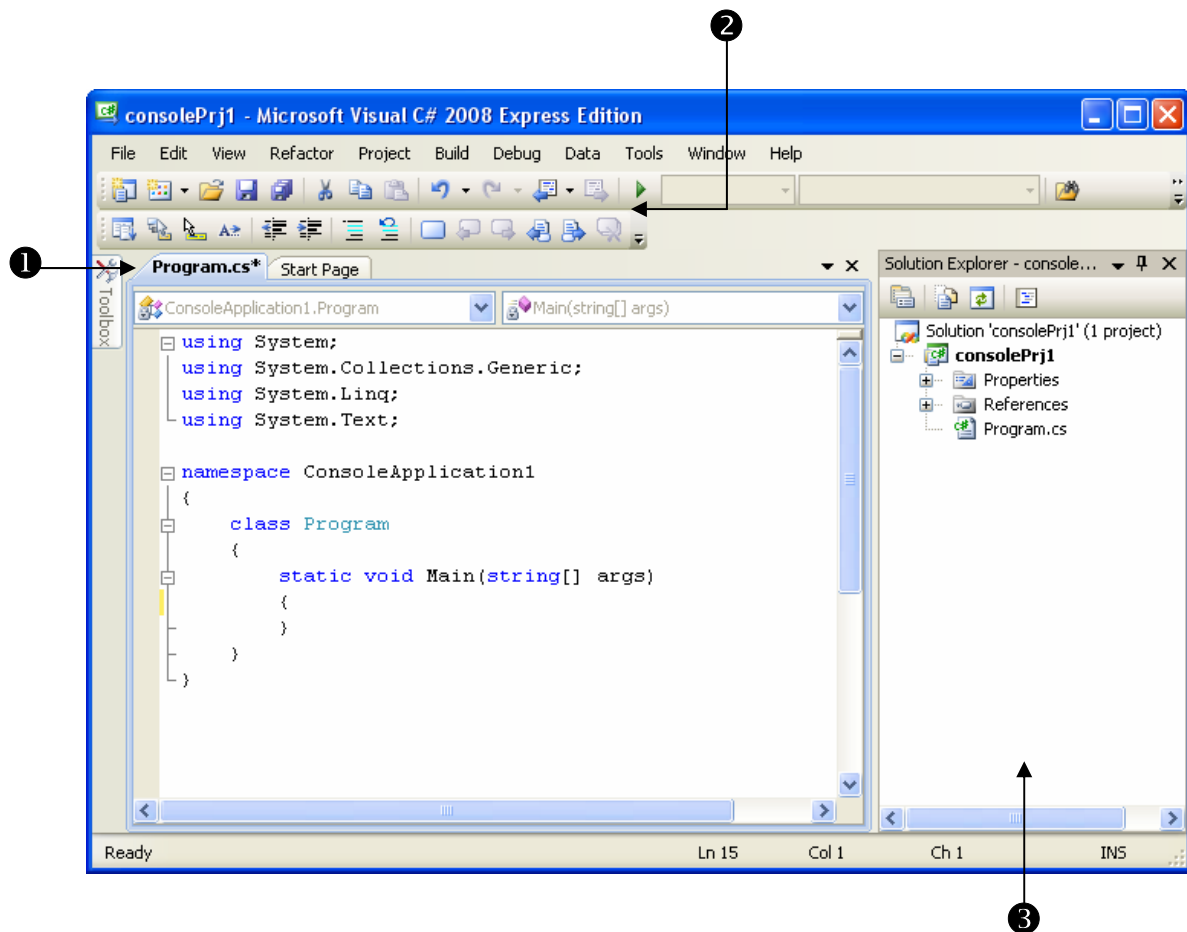
- ❶ เลือก สัญรูป Console Application
- ❷ ตั้งชื่อในช่อง Name
- ❸ คลิก OK



รูปที่ 4 กรอบโต้ตอบ New Project

3. โปรแกรมจะสร้างหน้าต่าง Program.cs พร้อมให้พัฒนาโปรแกรมต่อไป มีรายละเอียดที่ควรรู้ดังนี้

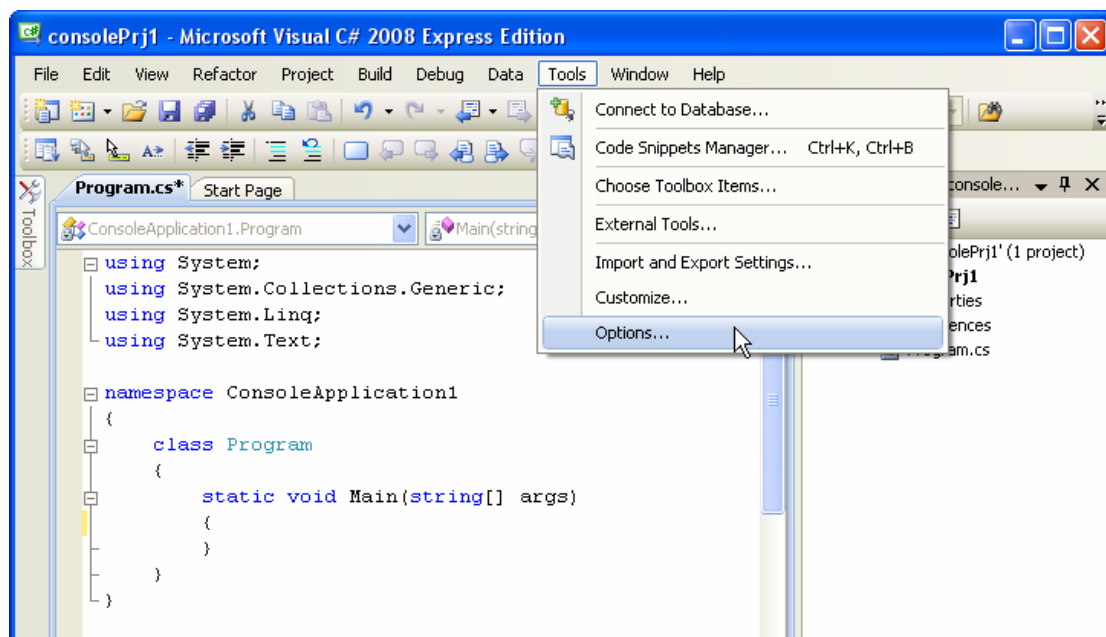
- 3.1 หน้าต่าง ที่เป็นส่วนเขียนโปรแกรมหรือ Editor จะมีชื่อตามชื่อไฟล์ คือ Program.cs ซึ่งเป็นส่วนที่ใช้เขียนคำสั่งต่าง ๆ ในภาษา C#
- 3.2 แถบเครื่องมือเพิ่มเติม
- 3.3 หน้าต่าง Solution Explorer ซึ่งใช้แสดงรายการองค์ประกอบของโปรเจกต์



รูปที่ 5 รายละเอียดหน้าต่าง Program.cs

4. การกำหนดให้แสดงเลขบรรทัดของคำสั่ง ทำได้ดังนี้

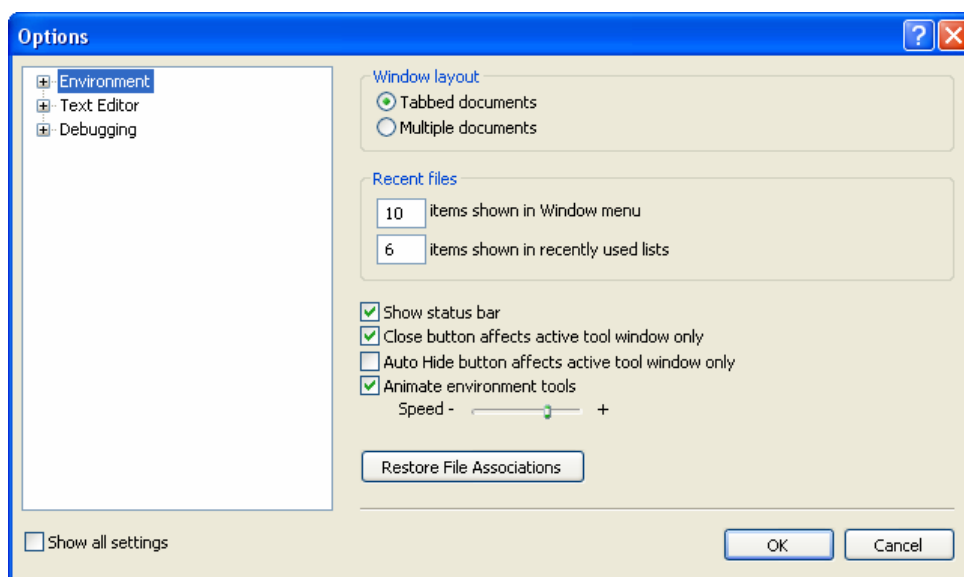
4.1 คลิกที่เมนู Tool > Options



รูปที่ 6 การกำหนดให้แสดงเลขบรรทัด

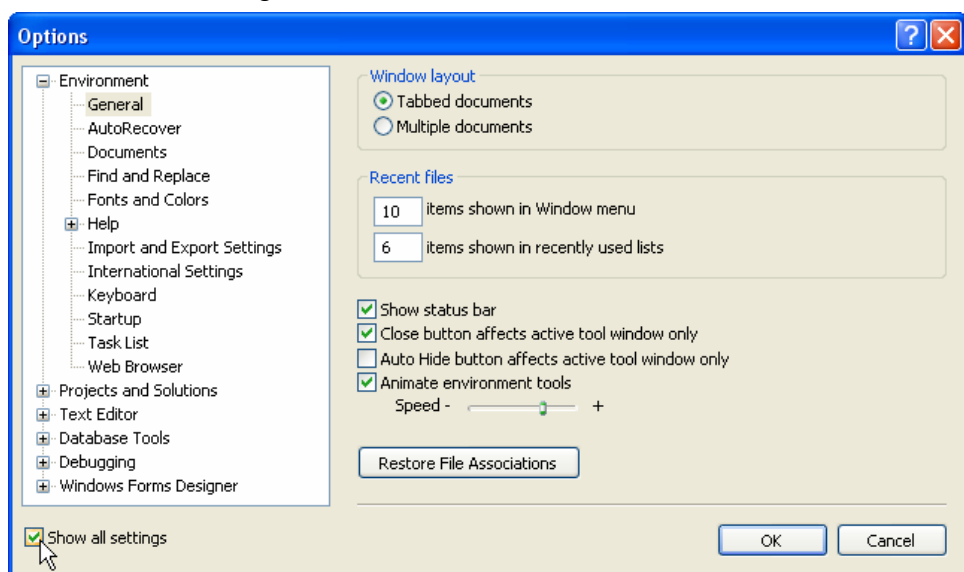


4.2 จะปรากฏกรอบโต้ตอบ Options ดังรูปที่ 7



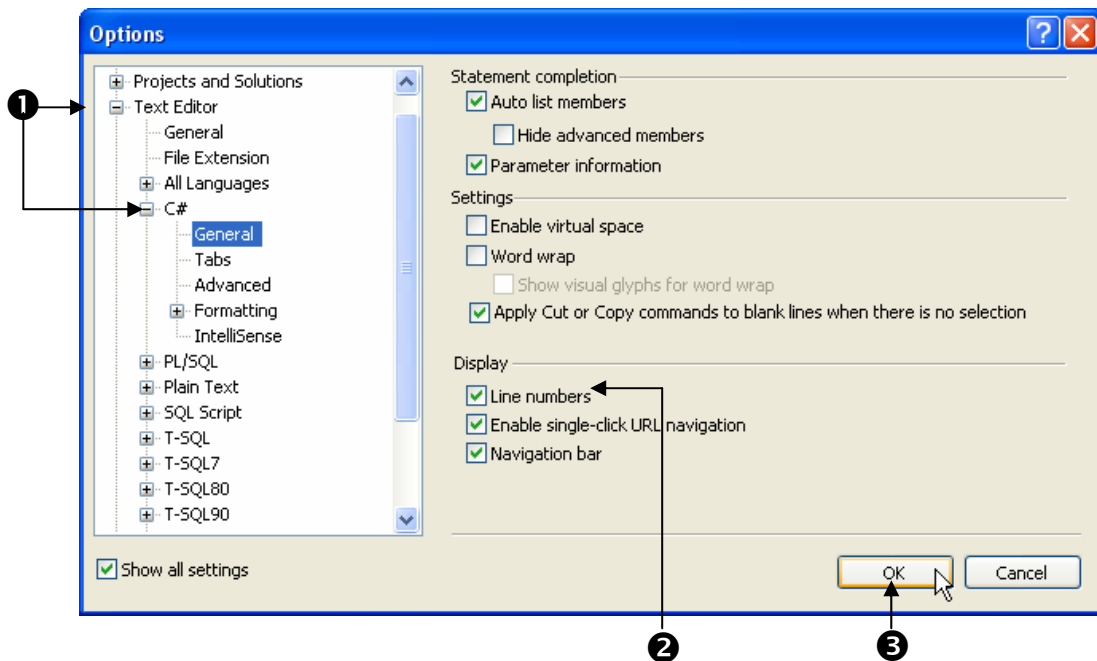
รูปที่ 7 กรอบโต้ตอบ Options

4.3 ให้เลือก Show all settings



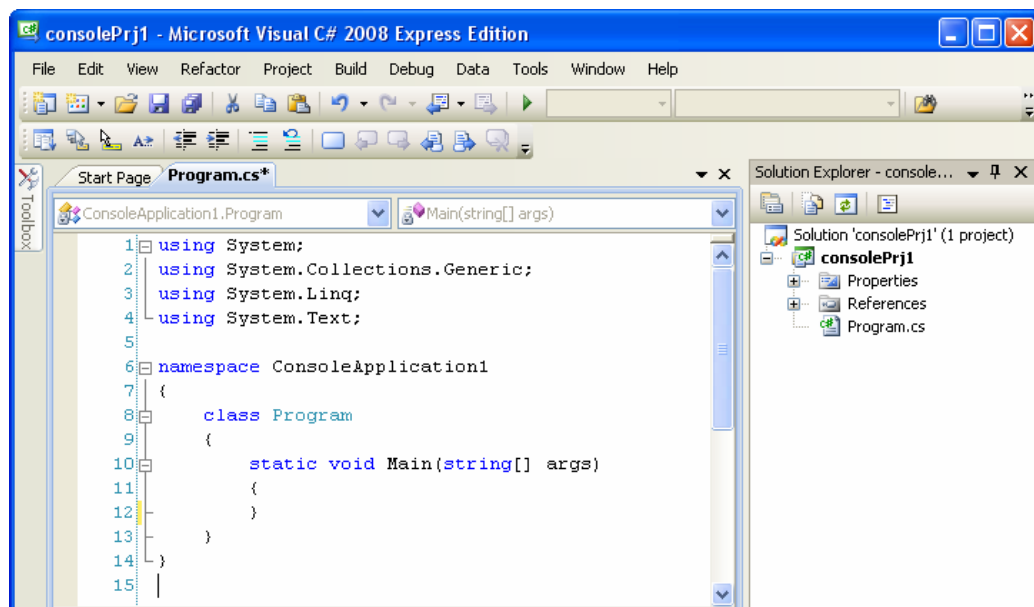
รูปที่ 8 การเลือก Show all settings

4.4 เลือก Text Editor -> C# -> General ดังหมายเลข ❶ จากนั้นที่ Display คลิกเลือก Line number ดังหมายเลข ❷ แล้วคลิก OK ดังหมายเลข ❸



รูปที่ 9 การเลือก Line numbers

4.5 หน้าต่าง Editor จะแสดงเลขบรรทัดดังรูปที่ 10



รูปที่ 10 เลขบรรทัด



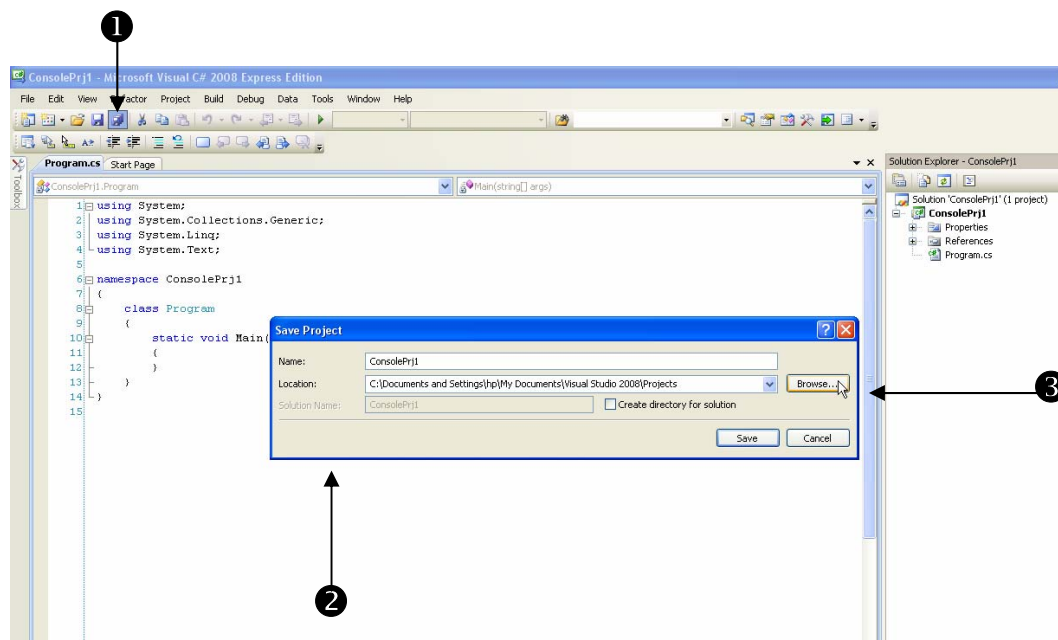
5. การบันทึกโปรเจกต์

การสร้างโปรเจกต์โดย MS Visual C# จะมีไฟล์เป็นองค์ประกอบของโปรแกรมหลายไฟล์ ผู้เขียนโปรแกรมควรสร้างโฟลเดอร์แยกแต่ละโปรเจกต์ออกจากกัน และควรบันทึกตั้งแต่เริ่มสร้างโปรเจกต์ใหม่ในทันที ดังนี้

5.1 คลิกที่ สัญลักษณ์ Save All

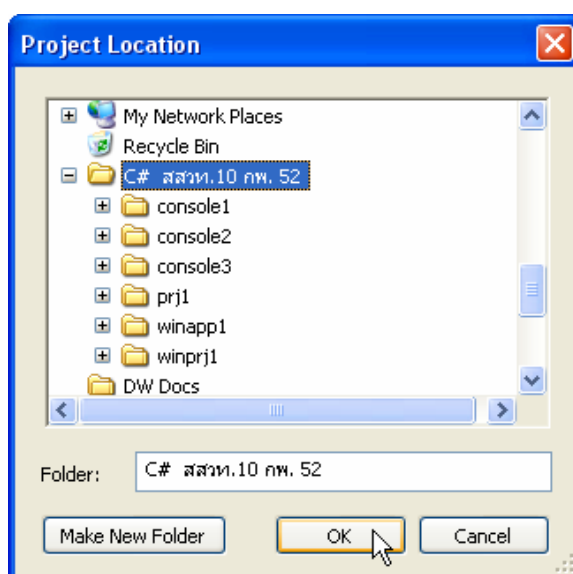
5.2 จะปรากฏกล่องโต้ตอบ Save Project

5.3 เลือกโฟลเดอร์เพื่อเก็บโปรเจกต์ในช่อง Location โดยคลิกที่ปุ่ม Browse...



รูปที่ 11 การบันทึกโปรเจกต์

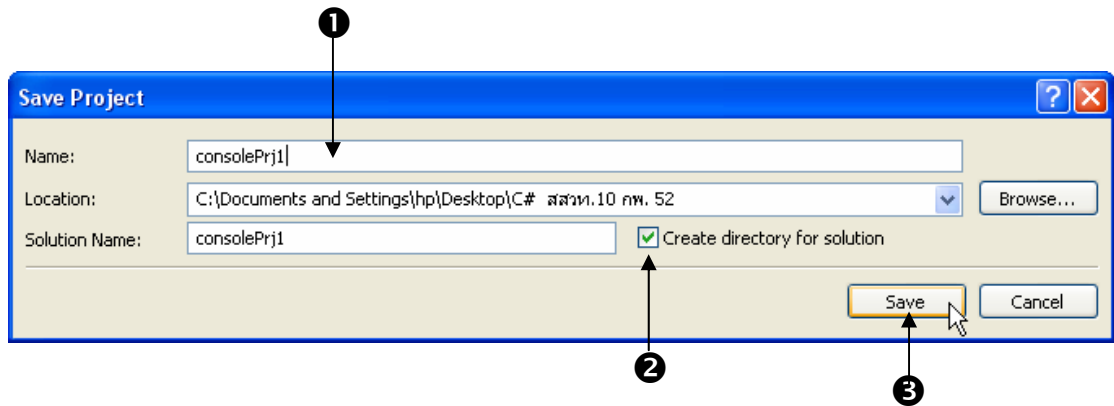
5.4 จะปรากฏกรอบโต้ตอบ Project Location ให้เลือกโฟลเดอร์ที่ต้องการ แล้วคลิกปุ่ม OK ดังรูปที่ 12



รูปที่ 12 กรอบโต้ตอบ Project Location

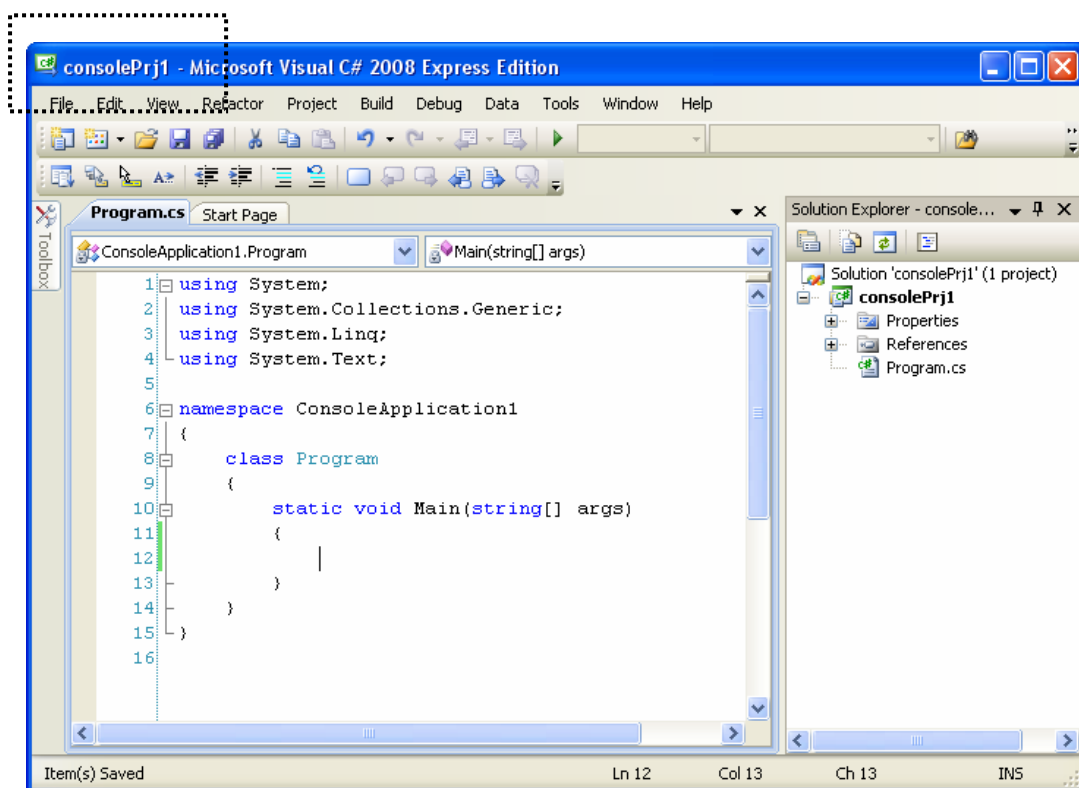
5.5 ในกรอบโต้ตอบ Save Project ในช่อง Location จะแสดงชื่อโฟลเดอร์ ให้ตั้งชื่อโปรเจกต์ แล้วบันทึกดังรูปที่ 13

- 1 ตั้งชื่อในช่อง Name ตามต้องการ
- 2 คลิกเลือก Create directory for solution
- 3 คลิก Save



รูปที่ 13 การบันทึกโปรเจกต์

5.6 จะได้โปรเจกต์ที่มีชื่อตามที่ได้ตั้งชื่อดังตัวอย่าง



รูปที่ 14 โปรเจกต์ consolePrj1



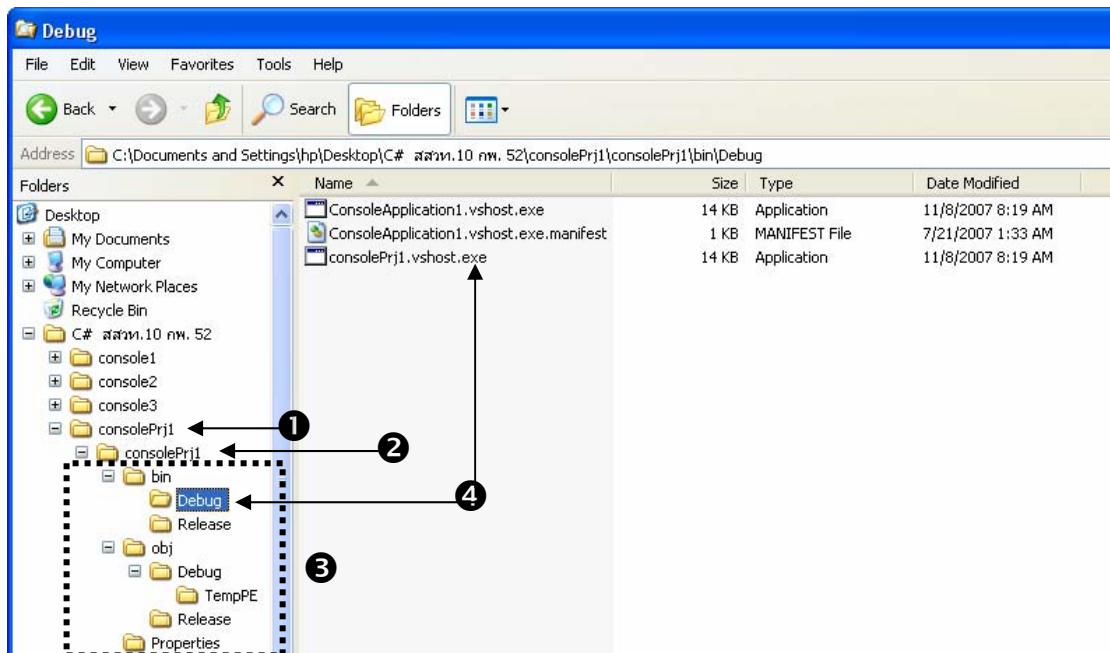
6. รายการไฟล์และโฟลเดอร์ที่เกิดจากการบันทึกโปรเจกต์

6.1 โฟลเดอร์ที่ใช้บันทึกโปรเจกต์

6.2 โฟลเดอร์โปรเจกต์ที่บันทึกไว้

6.3 รายการทรัพยากรที่เกิดขึ้นภายในโปรเจกต์

6.4 รายการไฟล์ที่อยู่ภายในโฟลเดอร์ Debug

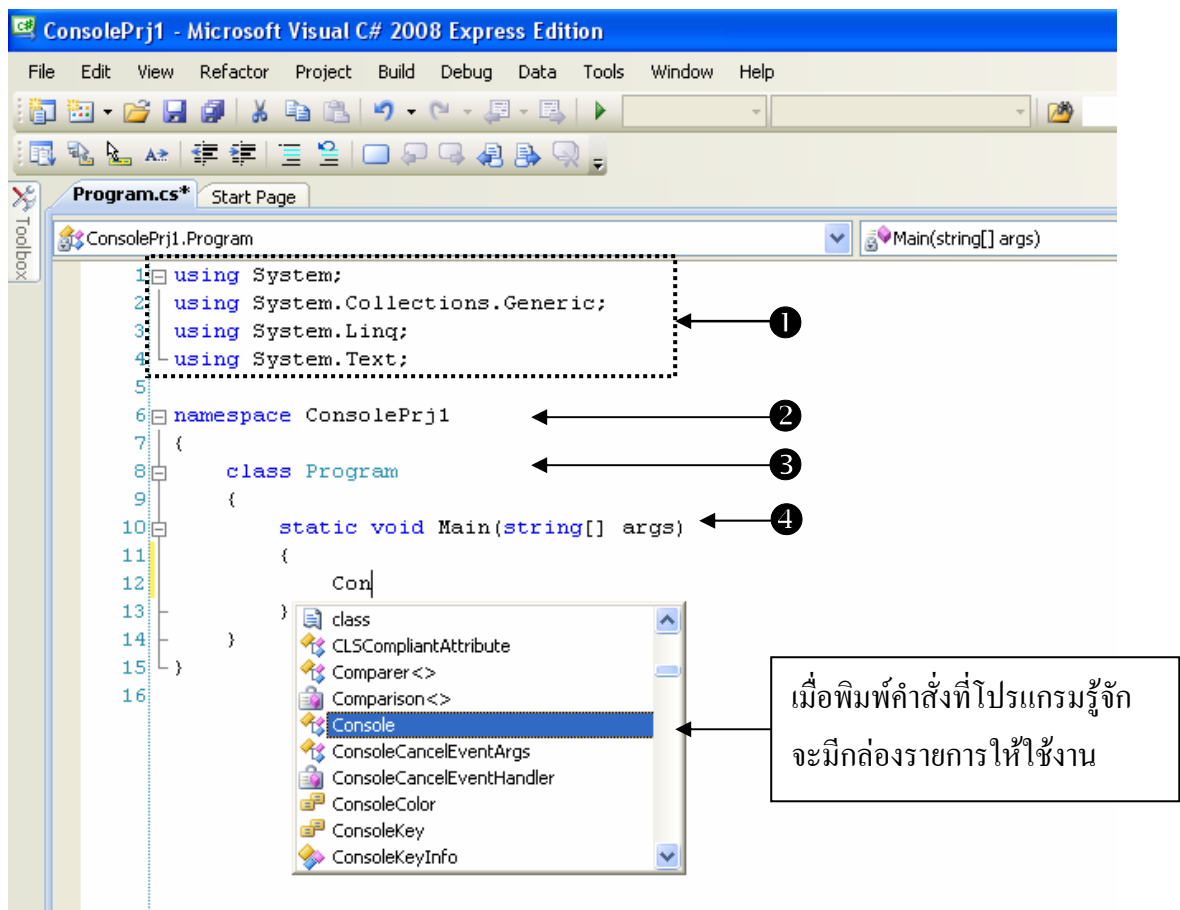


รูปที่ 15 รายการไฟล์และโฟลเดอร์ที่เกิดจากการบันทึกโปรเจกต์

7. การเริ่มเขียนคำสั่งในหน้าต่าง Editor

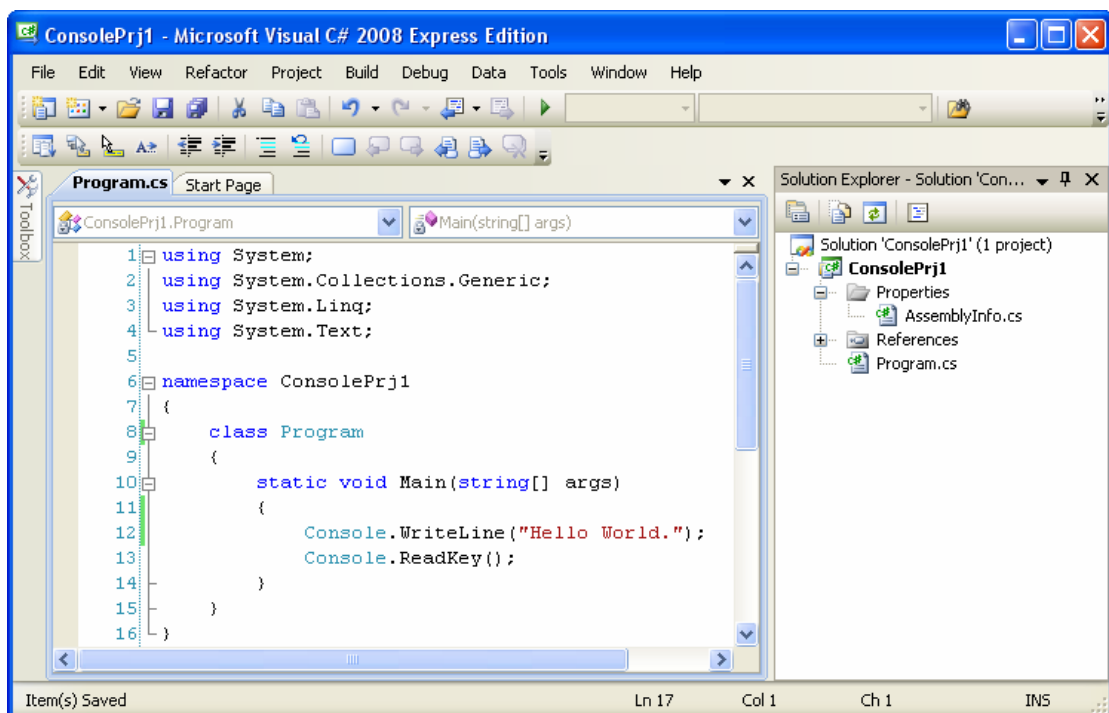
การสร้างโปรเจกต์โดย MS Visual C# จะต้องเขียนคำสั่งในไฟล์ .cs ซึ่งโดย Default จะมีชื่อว่า Program.cs โดยองค์ประกอบของโปรแกรมภาษา C# มีดังนี้

- ① ส่วน using เป็นส่วนที่ใช้เรียก namespace ที่รวม class ต่างๆ จากภายนอกมาใช้
- ② ส่วน namespace เป็นโครงสร้างบล็อกของโปรเจกต์ที่ประกอบด้วย class ต่างๆ ที่ผู้เขียนโปรแกรมสร้างขึ้น รวมทั้ง class Program ที่โปรแกรมสร้างให้
- ③ ส่วน class Program เป็นคลาสที่โปรแกรมสร้างให้โดยอัตโนมัติ ซึ่งสามารถเปลี่ยนชื่อได้ตามต้องการ ใน namespace หนึ่งๆ สามารถสร้าง class ได้หลาย class และแต่ละคลาส สามารถสร้างเมทอดได้หลายๆ เมทอด โดยมีเมทอดหลักคือ Main
- ④ ส่วนเมทอด Main เป็นเมทอดหลักของโปรแกรม ซึ่งเป็นส่วนเริ่มทำงานของโปรแกรมที่พัฒนาขึ้น ในโปรแกรมหนึ่งๆ จะมีเมทอด Main ได้เพียงเมทอดเดียว ไม่ว่าจะมียกี่คลาสนั้นก็ตาม



รูปที่ 16 คำสั่งในหน้าต่าง Editor

ตัวอย่างโปรแกรมที่แสดงข้อความบนหน้าจอ

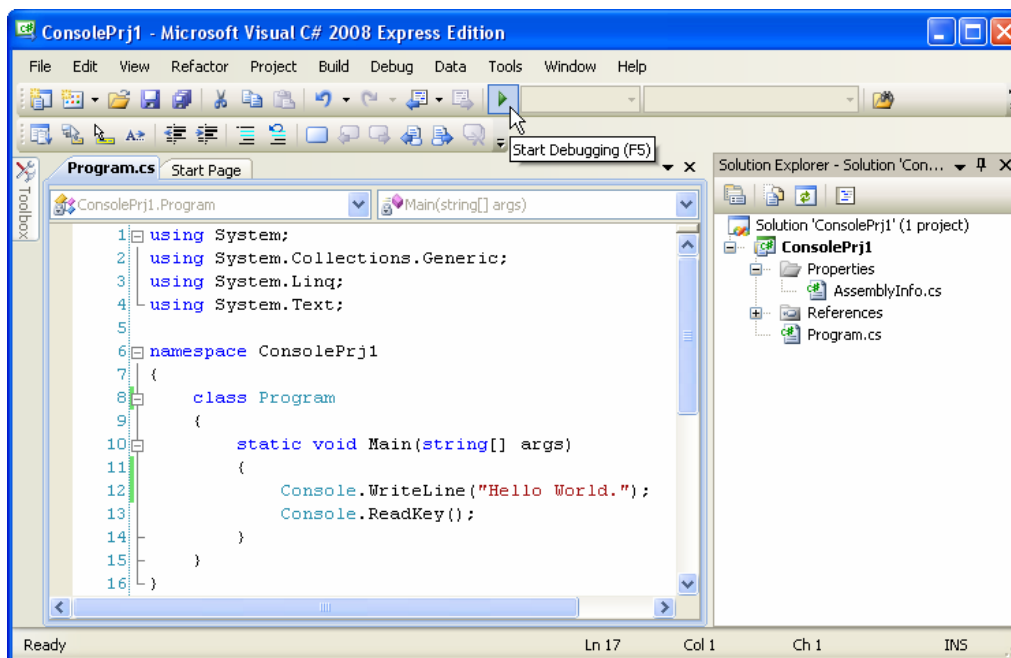


รูปที่ 17 ตัวอย่างโปรแกรมภาษาซีชาร์ป



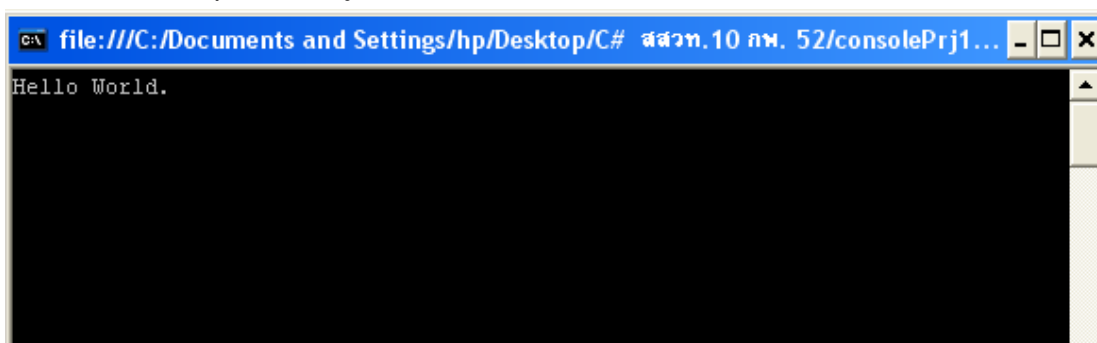
8. การรันโปรแกรม

เมื่อเขียนโปรแกรมเสร็จแล้ว ให้บันทึกโปรแกรม และเริ่มต้นรันโปรแกรม โดยการคลิกที่สัญลักษณ์ Start Debugging หรือกดคีย์ F5 ดังรูปที่ 18



รูปที่ 18 การรันโปรแกรม

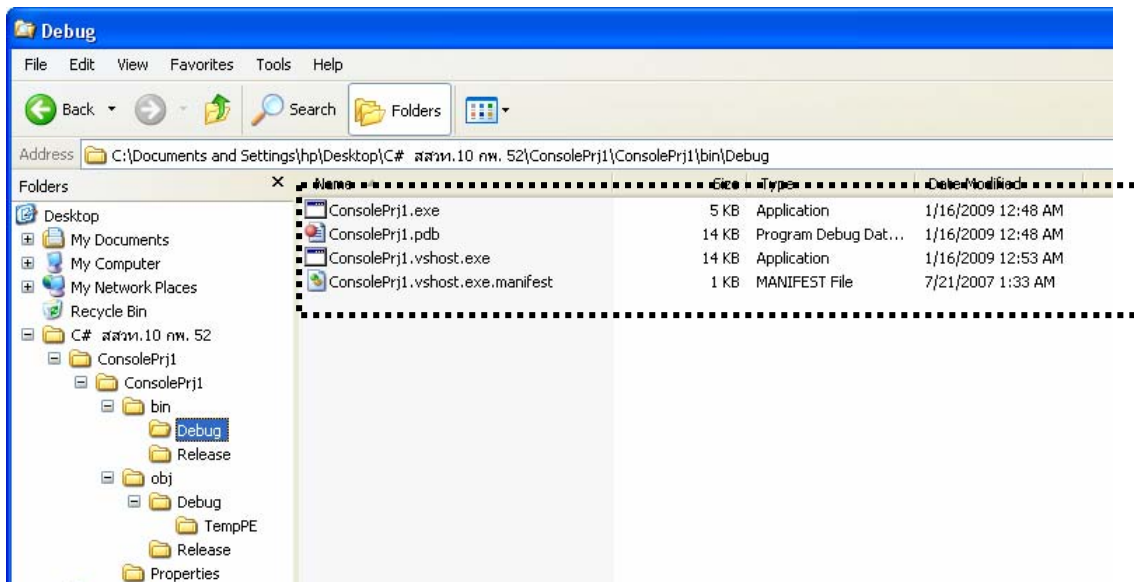
จะปรากฏผลลัพธ์ดังรูปที่ 19



รูปที่ 19 ผลลัพธ์การรันโปรแกรม

9. รายการไฟล์ที่เกิดขึ้นจากการรันโปรแกรม

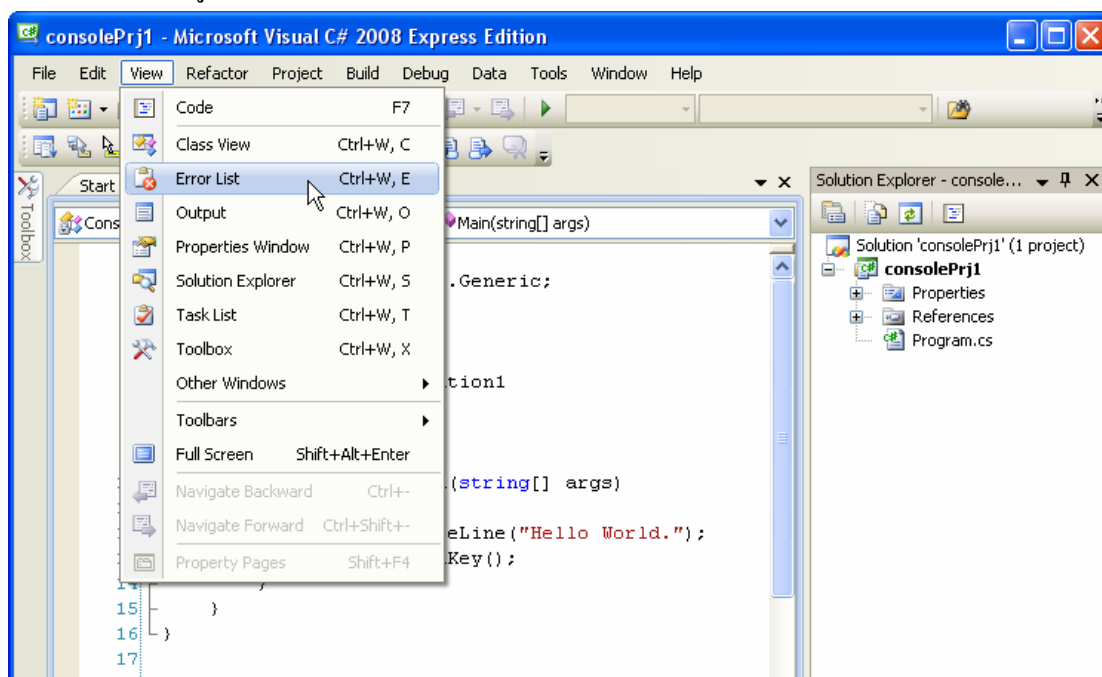
ในการรันโปรแกรมนั้น MS Visual C# จะทำการคอมไพล์ แล้วสร้างไฟล์ .exe ให้ก่อน ดังรูปที่ 20



รูปที่ 20 ไฟล์ .exe ที่เกิดขึ้นจากการรันโปรแกรม

10. การเตรียมเครื่องมือ เพื่อช่วยพัฒนาโปรแกรม เช่นการแสดงรายการการผิดพลาดของการเขียนคำสั่ง ตัวอย่างการแสดงผลหน้าต่าง Error List มีขั้นตอนดังนี้

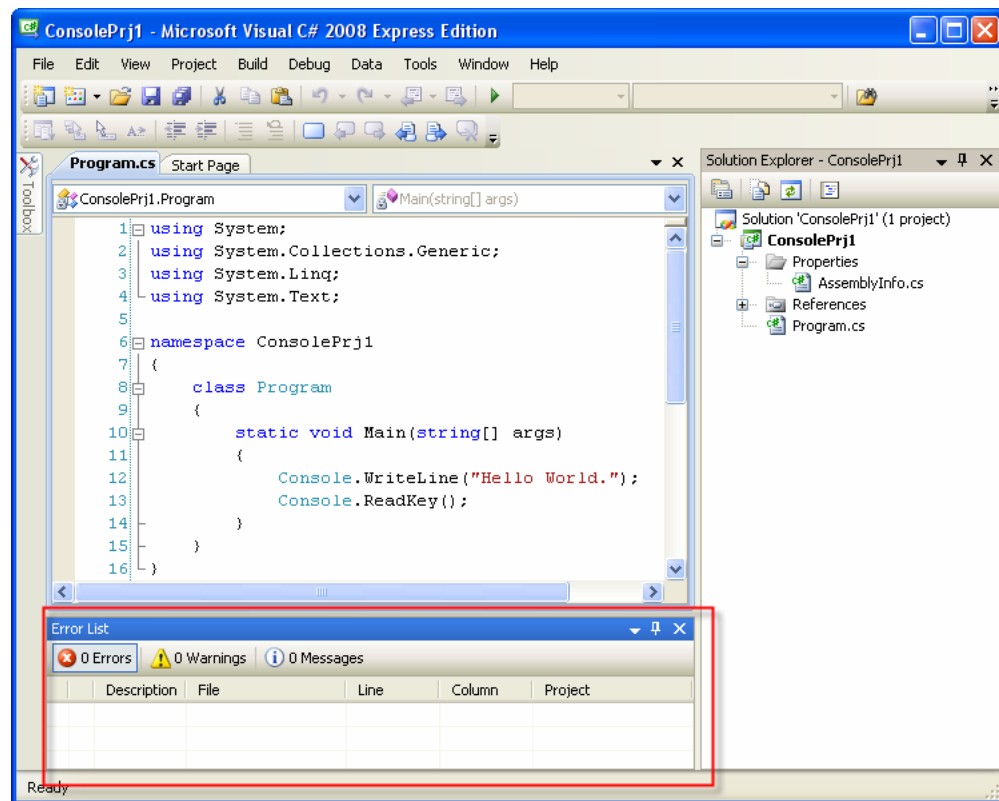
10.1 คลิกที่เมนู view -> Error List



รูปที่ 21 เมนู view -> Error List

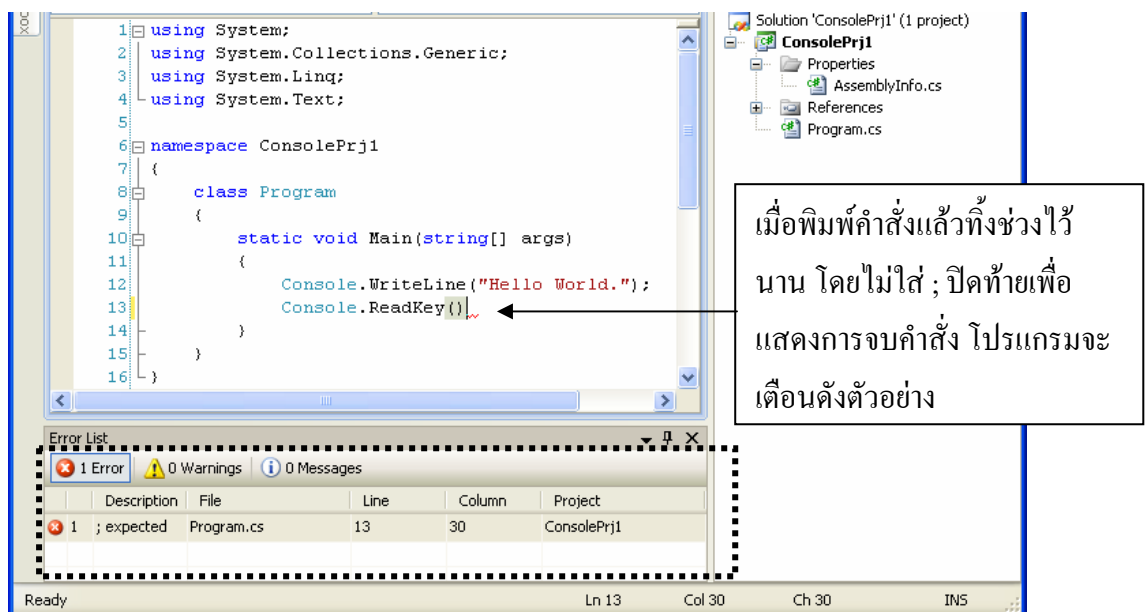


10.2 จะปรากฏหน้าต่าง Error List ดังรูปที่ 22



รูปที่ 22 หน้าต่าง Error List

ตัวอย่างการแสดงความผิดพลาดเมื่อโปรแกรมผิดพลาดในขณะที่กำลังรันโปรแกรม



รูปที่ 23 ตัวอย่างการแสดงความผิดพลาดเมื่อโปรแกรมผิดพลาด

กิจกรรมที่ 10

การจัดกลุ่มข้อมูลด้วยโครงสร้าง

1. จุดประสงค์ ให้ผู้เรียนสามารถ

- 1.1. อธิบายข้อมูลแบบโครงสร้าง
- 1.2. สร้างเมทอดสำหรับคำนวณเวกเตอร์
- 1.3. เขียนโปรแกรมโดยใช้งานโครงสร้างร่วมกับอาเรย์

2. แนวคิด

ภาษา C# รวมถึงภาษาโปรแกรมอีกหลายภาษารองรับการใช้งาน โครงสร้าง (structure) ที่อนุญาตให้เรานำข้อมูลย่อยต่าง ๆ ที่อาจประกอบด้วยข้อมูลต่างชนิดกันมารวมไว้เป็นกลุ่มเดียวกัน โดยที่การอ้างถึงก็จะกระทำผ่านตัวแปรตัวเดียวกันทั้งหมด การทำเช่นนั้นนอกจากจะทำให้โปรแกรมดูเป็นระเบียบขึ้นแล้ว การใช้โครงสร้างยังมีประโยชน์อย่างมากในการรวมข้อมูลเป็นกลุ่มเพื่อส่งไปประมวลผลในเมทอดอื่น ๆ ผ่านทางพารามิเตอร์เพียงตัวเดียว

3. สื่ออุปกรณ์

3.1 ใบงาน

ใบงานที่	เรื่อง	เวลา (นาที)
10.1	รู้จักกับข้อมูลแบบโครงสร้าง	30
10.2	เวกเตอร์	30
10.3	ฐานข้อมูลนักเรียน	60

3.2 ใบความรู้

- ใบความรู้ที่ 10.1 เรื่องชนิดข้อมูลแบบโครงสร้าง
- ใบความรู้ที่ 10.2 เรื่องการใช้งานโครงสร้างร่วมกับอาเรย์

3.3 อื่นๆ

- เครื่องคอมพิวเตอร์เท่ากับจำนวนกลุ่ม
- โปรแกรมวิซวลซีชาร์ป เอ็กเพรส

4. วิธีดำเนินการ

4.1 การจัดเตรียม



4.1.1 แบ่งผู้เรียนเป็นกลุ่ม กลุ่มละ 2 คน

4.1.2 เตรียมใบงานที่ 10.1-10.3 ตามจำนวนกลุ่ม และใบความรู้ที่ 10.1-10.2 ตามจำนวนผู้เรียน

4.2 ขั้นตอนการดำเนินการ

4.2.1 ผู้สอนกล่าวถึงเรื่องการจัดกลุ่มข้อมูลแบบโครงสร้าง

4.2.2 ผู้เรียนแต่ละกลุ่มศึกษาใบความรู้ที่ 10.1 เรื่องชนิดข้อมูลแบบโครงสร้าง จากนั้นทำใบงานที่ 10.1 เรื่องรู้จักกับข้อมูลแบบโครงสร้าง และใบงานที่ 10.2 เรื่องเวกเตอร์

4.2.3 ผู้สอนสุ่มผู้เรียนออกมานำเสนอคำตอบในใบงานที่ 10.1 และ 10.2

4.2.4 ผู้เรียนแต่ละกลุ่มศึกษาใบความรู้ที่ 10.2 เรื่องการใช้งานโครงสร้างร่วมกับอาเรย์ จากนั้นทำใบงานที่ 10.3 เรื่องฐานข้อมูลนักเรียน

4.2.5 ผู้เรียนและผู้สอนร่วมกันอภิปรายและสรุป

5. การวัดและประเมินผล

5.1 แบบประเมินโครงงาน

6. แหล่งความรู้เพิ่มเติม

6.1 [http://msdn.microsoft.com/th-th/vcsharp/default\(en-us\).aspx](http://msdn.microsoft.com/th-th/vcsharp/default(en-us).aspx)

7. ข้อเสนอแนะ

-

ใบงานที่ 10.1

รู้จักกับข้อมูลแบบโครงสร้าง

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 10.1 แล้วตอบคำถามต่อไปนี้

1. เรียนรู้การใช้งานข้อมูลโครงสร้าง

1.1 นิยามโครงสร้างชื่อ *VehicleInfo* สำหรับเก็บข้อมูลรถยนต์แต่ละคัน ซึ่งประกอบด้วยสมาชิกดังนี้

- ยี่ห้อรถ เป็นข้อมูลชนิดข้อความ อ้างถึงโดยใช้ชื่อ *make*
- ทะเบียนรถ เป็นข้อมูลชนิดข้อความ อ้างถึงโดยใช้ชื่อ *plate*
- สีรถ เป็นข้อมูลชนิดข้อความ อ้างถึงโดยใช้ชื่อ *color*
- ปีที่ผลิต เป็นข้อมูลชนิดตัวเลขจำนวนเต็ม อ้างถึงโดยใช้ชื่อ *year*

```
struct _____ {  
    _____;  
    _____;  
    _____;  
    _____;  
}
```

1.2 เราจะนิยามเมทอดขึ้นมาสองเมทอดเพื่อจัดการข้อมูลเกี่ยวกับเวกเตอร์และนำไปใช้ได้ในงานหลัง ๆ เมทอดแรกคือ *ReadVector* ซึ่งรับข้อมูลเวกเตอร์สามมิติจากผู้ใช้งานและส่งค่าคืนกลับมาในรูปโครงสร้าง *Vector* อีกเมทอดหนึ่งคือ *PrintVector* ใช้สำหรับแสดงผลข้อมูลในโครงสร้าง *Vector* ออกทางหน้าจอ จงเติมคำสั่งลงในช่องว่างเพื่อให้โปรแกรมทำงานได้ตรงกับตัวอย่างผลลัพธ์ที่กำหนดให้

```
using System;  
class VectorEx {  
    struct Vector {  
        public double x,y,z;  
    }  
  
    static Vector ReadVector() {  
        Vector v;  
        Console.Write("X element: ");  
        _____ = double.Parse(Console.ReadLine());  
        Console.Write("Y element: ");
```



```

        _____;
        _____;
        _____;

    _____;
}

static void PrintVector(Vector v) {
    Console.Write("{0},{1},{2}",
        _____, _____, _____);
}

static void Main() {
    Console.WriteLine("Enter a vector");
    Vector a = ReadVector();
    Console.Write("You justed enter a vector ");
    PrintVector(a);
    Console.WriteLine();
}
}

```

ตัวอย่างผลการทำงาน

```

Enter a vector
X element: 6
Y element: 13
Z element: 1
You just entered a vector (6,13,1)

```

ใบงานที่ 10.2

เวกเตอร์

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 10.1 แล้วตอบคำถามต่อไปนี้

1. นิยามโครงสร้างและเมทอดเพื่อประมวลผลข้อมูลเวกเตอร์

เราจะนิยามเมทอดขึ้นมาสองเมทอดเพื่อจัดการข้อมูลเกี่ยวกับเวกเตอร์และนำไปใช้ในงานต่อไป เมทอดแรกคือ *ReadVector* ซึ่งรับข้อมูลเวกเตอร์สามมิติจากผู้ใช้และส่งค่าคืนกลับมาในรูปโครงสร้าง *Vector* อีกเมทอดหนึ่งคือ *PrintVector* ใช้สำหรับแสดงผลข้อมูลในโครงสร้าง *Vector* ออกทางหน้าจอ

จงเติมคำสั่งลงในช่องว่างเพื่อให้โปรแกรมทำงานได้ตรงกับตัวอย่างผลลัพธ์ที่กำหนดให้

```
using System;
class VectorEx {
    struct Vector {
        public double x,y,z;
    }

    static Vector ReadVector() {
        Vector v;
        Console.Write("X element: ");
        _____ = double.Parse(Console.ReadLine());
        Console.Write("Y element: ");
        _____;
        _____;
        _____;
        _____;
    }

    static void PrintVector(Vector v) {
        Console.Write("{0},{1},{2}",
            _____, _____, _____);
    }

    static void Main() {
        Console.WriteLine("Enter a vector");
        Vector a = ReadVector();
        Console.Write("You justed enter a vector ");
        PrintVector(a);
        Console.WriteLine();
    }
}
```



ตัวอย่างผลการทำงาน

```
Enter a vector
X element: 6
Y element: 13
Z element: 1
You just entered a vector (6,13,1)
```

2. สร้างเมทอดสำหรับคำนวณขนาดของเวกเตอร์

ให้ $\mathbf{v} = (v_x, v_y, v_z)$ เป็นเวกเตอร์ในปริภูมิสามมิติ ขนาดของเวกเตอร์ \mathbf{v} เขียนแทนด้วย $|\mathbf{v}|$ สามารถคำนวณได้จากสูตร

$$|\mathbf{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

โปรแกรมด้านล่างมีการนิยามเมทอดชื่อ *VectorSize* ขึ้นมาเพื่อคำนวณขนาดของเวกเตอร์ โปรแกรมนี้อาศัยเมทอด *ReadVector* และ *PrintVector* จากแบบฝึกหัดที่แล้วเพื่อรับข้อมูลเวกเตอร์จากผู้ใช้และแสดงผลข้อมูลในเวกเตอร์ (เมทอดเหล่านี้รวมถึงการนิยามโครงสร้าง *Vector* ถูกละไว้เพื่อสงวนพื้นที่) จงเติมคำสั่งที่เหมาะสมลงในช่องว่างเพื่อให้การนิยามเมทอด *VectorSize* เสร็จสมบูรณ์

```
using System;
class VectorEx {
    // คัดลอกนิยามโครงสร้าง Vector มาปะในตำแหน่งนี้

    // คัดลอกเมทอด PrintVector และ ReadVector
    // จากแบบฝึกหัดที่แล้วมาปะในตำแหน่งนี้

    static double VectorSize(Vector v) {
        return _____;
    }

    static void Main() {
        Console.WriteLine("Enter a vector");
        Vector v = ReadVector();
        Console.Write("The size of the vector ");
        PrintVector(v);
        Console.WriteLine(" is {0}", VectorSize(v));
    }
}
```

ตัวอย่างผลการทำงาน

```
Enter a vector
X element: 3
Y element: 7
Z element: -2
The size of the vector (3,7,-2) is 7.87400787401181
```



3. สร้างเมทอดเพื่อคำนวณผลคูณจุดของเวกเตอร์สองตัว

สำหรับเวกเตอร์ $\mathbf{u} = (u_x, u_y, u_z)$ และเวกเตอร์ $\mathbf{v} = (v_x, v_y, v_z)$ ใด ๆ ผลคูณจุด (dot product) ของเวกเตอร์ \mathbf{u} และ \mathbf{v} มีสัญลักษณ์เป็น $\mathbf{u} \cdot \mathbf{v}$ ให้ผลลัพธ์เป็นปริมาณสเกลาร์ที่มีค่าตามสูตร

$$\mathbf{u} \cdot \mathbf{v} = u_x v_x + u_y v_y + u_z v_z$$

จงทำโปรแกรมต่อไปนี้ให้สมบูรณ์เพื่อให้โปรแกรมรับเวกเตอร์สองจำนวนจากผู้ใช้งานและแสดงผลคูณจุดที่เกิดจากเวกเตอร์ทั้งคู่

```
using System;
class VectorEx {
    // คัดลอกนิยามโครงสร้าง Vector มาปะในตำแหน่งนี้

    // คัดลอกเมทอด ReadVector จากแบบฝึกหัดที่แล้วมาปะในตำแหน่งนี้

    static double DotVectors(Vector u, Vector v) {
        _____;
    }

    static void Main() {
        Console.WriteLine("Enter vector u");
        Vector u = ReadVector();
        _____;
        _____;
        Console.WriteLine("u * v = {0}", DotVectors(u,v));
    }
}
```

ตัวอย่างผลการทำงาน

```
Enter vector u
X element: -1.2
Y element: 3
Z element: 0.5
Enter vector v
X element: 4
Y element: 2
Z element: 1.8
u * v = 2.1
```



ใบงานที่ 10.3

ฐานข้อมูลนักเรียน

รายชื่อสมาชิกในกลุ่มที่.....

1. 2.
3. 4.

ให้ผู้เรียนศึกษาใบความรู้ที่ 10.2 แล้วเขียนโปรแกรมจากโจทย์ที่กำหนดให้ต่อไปนี้

เขียนโปรแกรมเพื่อจัดการระเบียนนักเรียนที่สามารถค้นหาข้อมูลนักเรียนตามรหัสประจำตัวได้ โดยข้อมูลของนักเรียนแต่ละคนประกอบด้วย

- รหัสประจำตัว มีค่าได้ตั้งแต่ 1 ถึง 999
- ชื่อ-นามสกุล
- เกรดเฉลี่ยสะสม

โปรแกรมจะอ่านข้อมูลนักเรียนทั้งหมดเข้ามาก่อนแล้วจึงเข้าสู่โหมดการค้นหาโดยเริ่มรับรหัสประจำตัวที่ต้องการค้นหาจากผู้ใช้ หากพบรหัสที่ตรงกันในฐานข้อมูลโปรแกรมจะรายงานชื่อและเกรดเฉลี่ยสะสมของนักเรียนผู้นั้นให้ทราบ แต่ถ้าหากไม่พบรหัสก็จะไม่มีการแสดงผลใดๆ โปรแกรมจะจบการทำงานเมื่อผู้ใช้ป้อนรหัสที่ต้องการค้นเป็น -1

ตัวอย่างผลการทำงาน

```
How many students? 3
Enter student#1's information
ID: 234
Name: Arthur
GPA: 4.00
Enter student#2's information
ID: 876
Name: Paula
GPA: 3.99
Enter student#3's information
ID: 379
Name: Ariya
GPA: 1.50
Enter ID to search (-1 to quit): 379
Name: Ariya
GPA: 1.50
Enter ID to search (-1 to quit): 111
Enter ID to search (-1 to quit): 234
Name: Arthur
GPA: 4.00
```


Enter ID to search (-1 to quit): -1

คัดลอกโปรแกรมลงในช่องว่าง



ใบความรู้ที่ 10.1

ชนิดข้อมูลแบบโครงสร้าง

ในบางแก้ปัญหาบางอย่างเราต้องการเขียนโปรแกรมเพื่อจัดการชุดข้อมูลที่ประกอบด้วยข้อมูลย่อย ๆ ทั้งที่เป็นชนิดเดียวกันและต่างชนิดกัน ทว่าข้อมูลเหล่านี้มักต้องถูกใช้งานร่วมกันอยู่เป็นประจำ ลองพิจารณาลักษณะงานเกี่ยวกับทะเบียนนักเรียนที่ข้อมูลของนักเรียนคนหนึ่ง ๆ ประกอบด้วยข้อมูลย่อยหลายส่วน อาทิ เช่น รหัสประจำตัว ชื่อ นามสกุล ภาควิชา อายุ อาจารย์ที่ปรึกษา ฯลฯ แม้ว่าเราจะสามารถเขียนโปรแกรมโดยกำหนดให้ข้อมูลเหล่านี้ถูกแยกเก็บไว้ในตัวแปรที่แตกต่างกันได้ก็ตาม โปรแกรมที่ได้จะเต็มไปด้วยตัวแปรทั้งที่ใช้เก็บข้อมูลนักเรียนปะปนอยู่กับตัวแปรอีกหลายตัวที่ใช้สำหรับจุดประสงค์อื่นภายในโปรแกรม อันมีผลทำให้โปรแกรมยากต่อการทำความเข้าใจและแก้ไขเพิ่มเติมในภายหลัง

เช่นเดียวกับอาร์เรย์ ภาษา C# รวมถึงภาษาโปรแกรมอีกหลายภาษารองรับการใช้งาน โครงสร้าง (structure) ที่อนุญาตให้เรานำข้อมูลย่อยต่าง ๆ ที่อาจประกอบด้วยข้อมูลต่างชนิดกันมารวมไว้เป็นกลุ่มเดียวกัน โดยที่การอ้างถึงก็จะกระทำผ่านตัวแปรตัวเดียวกันทั้งหมด การทำเช่นนี้นอกจากจะทำให้โปรแกรมดูเป็นระเบียบขึ้นแล้ว การใช้โครงสร้างยังมีประโยชน์อย่างมากในการรวมข้อมูลเป็นกลุ่มเพื่อส่งไปประมวลผลในเมทอดอื่น ๆ ผ่านทางพารามิเตอร์เพียงตัวเดียว

1. การนิยามโครงสร้าง

เนื่องจากข้อมูลที่เราจะรวมไว้เป็นกลุ่มเดียวกันนั้นอาจมีชนิดข้อมูลที่แตกต่างกันได้หลากหลายรูปแบบต่างจากอาร์เรย์ที่ข้อมูลซึ่งถูกนำมารวมกันจะต้องเป็นข้อมูลชนิดเดียวกันเท่านั้น จึงเป็นสิ่งจำเป็นที่เราต้องนิยามโครงสร้างข้อมูลที่ชัดเจนขึ้นมาเสียก่อนว่ากลุ่มข้อมูลจะประกอบด้วย สมาชิก (member) ที่มีแบบข้อมูลชนิดใดบ้าง และสมาชิกแต่ละตัวจะถูกอ้างถึงอย่างไร การนิยามโครงสร้างในภาษา C# นั้นอาศัยคีย์เวิร์ด **struct** (มาจากคำว่า structure) ซึ่งมีรูปแบบดังนี้

```
struct StructName {  
    public DataType1 var1;  
    public DataType2 var2;  
    :  
    public DataTypeN varN;  
}
```

รูปแบบข้างต้นเป็นการนิยามโครงสร้างชื่อ *StructName* ซึ่งประกอบด้วยสมาชิก *N* ตัว สมาชิกจำนวนแรกมีชนิดข้อมูลเป็น *DataType1* และถูกอ้างถึงผ่านชื่อ *var1* สมาชิกตัวถัดมามีชนิดข้อมูลเป็น *DataType2* และถูกอ้างถึงผ่านชื่อ *var2* เช่นนี้เรื่อยไป สำหรับคีย์เวิร์ด **public** ที่ปรากฏหน้าการนิยามสมาชิกแต่ละตัวเป็นการกำหนดระดับการปกป้องข้อมูล (protection level) ซึ่งเราจะยังไม่สนใจในที่นี้

การนิยามโครงสร้างต้องปรากฏอยู่ภายในคลาส แต่อยู่ภายนอกเมทอดใด ๆ เสมอ

ตัวอย่างที่ 10.1 นิยามโครงสร้างชื่อ *StdInfo* เพื่อใช้เก็บข้อมูลนักเรียนแต่ละคน ซึ่งประกอบด้วยสมาชิกดังนี้

- รหัสประจำตัว เป็นข้อมูลชนิดตัวเลขจำนวนเต็ม อ้างถึงโดยใช้ชื่อ *id*
- ชื่อนักเรียน เป็นข้อมูลชนิดข้อความ อ้างถึงโดยใช้ชื่อ *name*
- คะแนนนักเรียน เป็นข้อมูลชนิด **double** อ้างถึงโดยใช้ชื่อ *score*

```
struct StdInfo {  
    public int id;  
    public string name;  
    public double score;  
}
```

ตามที่กล่าวไว้ข้างต้น การนิยามโครงสร้างภายในโปรแกรมจะต้อง อยู่ภายในคลาสแต่อยู่นอกเมทอดเสมอ ตัวอย่างเช่น

```
class MyClass {  
    struct StdInfo {  
        public int id;  
        public string name;  
        public double score;  
    }  
  
    static void Main() {  
        :  
    }  
}
```

2. การประกาศและการใช้งานตัวแปรชนิดโครงสร้าง

การใช้คีย์เวิร์ด **struct** ข้างต้นนั้นเป็นเพียงการนิยามโครงสร้างขึ้นมาเท่านั้น ยังมิได้มีผลทำให้โปรแกรมสร้างเนื้อที่สำหรับเก็บข้อมูลขึ้นมาภายในหน่วยความจำแต่อย่างใด การนำโครงสร้างมาใช้เก็บข้อมูลจริง ๆ จะต้องมีการประกาศตัวแปรที่ระบุชนิดข้อมูลเป็นชื่อของโครงสร้างนั้น ๆ เสียก่อน ดังรูปแบบต่อไปนี้

```
StructName structVar;
```

โดย *StructName* คือชื่อของโครงสร้างที่ได้นิยามไปแล้ว และ *structVar* คือชื่อตัวแปรที่นำมาใช้อ้างอิงถึงข้อมูลภายในโครงสร้างนั้น เราจะเรียกใช้คำสั่งแบบนี้ได้ภายในเมทอดเท่านั้น

สังเกตว่าคำสั่งข้างต้นนั้นมีรูปแบบเดียวกันกับคำสั่งที่ใช้ในการประกาศตัวแปรทั่วไปทุกประการ ดังนั้นการใช้งานคีย์เวิร์ด **struct** จึงเปรียบเสมือนการสร้างชนิดข้อมูลชนิดใหม่ขึ้นมาั่นเอง ซึ่งก็หมายความว่านอกเหนือจากการประกาศตัวแปรแล้ว เรายังสามารถนำชื่อโครงสร้างไปใช้ในส่วนอื่นของ

โปรแกรมได้อีก อาทิเช่น ใช้ระบุนชนิดข้อมูลของพารามิเตอร์สำหรับเมทอด ระบุนชนิดข้อมูลที่เมทอดคืนค่า หรือแม้แต่การนำไปใช้นิยามสมาชิกในโครงสร้างอื่น ๆ

เนื่องจากตัวแปรชนิดโครงสร้างไม่ได้เป็นตัวแปรที่เก็บค่าเพียงค่าเดียว แต่เป็นเสมือนตัวแทนกลุ่มข้อมูลที่มีรูปแบบตามโครงสร้างนั้น ๆ การเข้าถึงข้อมูลภายในโครงสร้างจึงต้องมีการระบุชัดเจนว่าข้อมูลชิ้นใดที่ถูกอ้างถึง ซึ่งทำได้โดยการระบุชื่อของสมาชิกต่อท้ายชื่อตัวแปรแบบโครงสร้าง โดยค้นด้วยเครื่องหมายจุด (.) ดังแสดง

```
structVar.memberName
```

เช่นเดียวกับอาเรย์ การอ้างถึงสมาชิกในโครงสร้างเช่นนี้จะมีการใช้งานเสมือนเป็นตัวแปรโคดตัวหนึ่งที่มีชนิดข้อมูลตามที่กำหนดให้กับสมาชิกในระหว่างนิยามโครงสร้าง นั่นคือหากเราใช้การอ้างอิงนี้เป็นส่วนหนึ่งของนิพจน์ใด ๆ ค่าของสมาชิกจะถูกดึงออกมาใช้เพื่อประเมินค่าของนิพจน์นั้น ๆ ในทางตรงกันข้าม หากเราวางการอ้างอิงนี้ไว้ทางด้านซ้ายของเครื่องหมาย = ในคำสั่งให้ค่ากับตัวแปร ค่าสมาชิกตำแหน่งนี้ก็จะถูกเปลี่ยนค่าไปตามค่าที่กำหนดให้

ตัวอย่างที่ 10.2 โปรแกรมด้านล่างนิยามโครงสร้างชื่อ *Vector* เพื่อใช้แทนข้อมูลแบบเวกเตอร์ในปริภูมิสามมิติ ภายในโครงสร้างประกอบด้วยสมาชิกชื่อ *x*, *y* และ *z* ที่มีชนิดข้อมูลแบบ **double** ใช้สำหรับเก็บค่าในแต่ละแกนของเวกเตอร์ ภายในโปรแกรมหลักจะมีการสร้างตัวแปรแบบ *Vector* ขึ้นมาหนึ่งตัวและกำหนดให้มีค่าเป็น (3,4,5) จากนั้นจึงนำค่าเหล่านี้มาแสดงผลบนหน้าจอในรูปเวกเตอร์

```
1: using System;
2: class VectorEx {
3:     struct Vector {
4:         public double x;
5:         public double y;
6:         public double z;
7:     }
8:
9:     static void Main() {
10:         Vector v1;
11:         v1.x = 3; v1.y = 4; v1.z = 5;
12:         Console.WriteLine("Vector v1 = ({0},{1},{2})",
13:             v1.x, v1.y, v1.z);
14:     }
15: }
```

สังเกตว่าในการนิยามโครงสร้าง *Vector* นั้นมีสมาชิกที่มีชนิดของข้อมูลเป็น **double** เหมือนกันหมด ในกรณีนี้ภาษา C# อนุญาตให้เราประกาศสมาชิกไว้ภายใต้คำสั่งเดียวกันได้ ดังนั้นโปรแกรมในช่วงบรรทัดที่ 3-7 จึงเขียนให้สั้นลงได้เป็น

```
struct Vector {
    public double x,y,z;
}
```

ใบความรู้ที่ 10.2

การใช้งานโครงสร้างร่วมกับอาร์เรย์

ที่ผ่านมาเราได้เรียนรู้การประกาศตัวแปรแบบโครงสร้างสำหรับสิ่งของหรือวัตถุเพียงชิ้นเดียว เช่น นักเรียนหนึ่งคนหรือเวกเตอร์หนึ่งเวกเตอร์ ในบางครั้งเราจำเป็นต้องประมวลผลข้อมูลเกี่ยวกับวัตถุต่าง ๆ เหล่านี้มากกว่าหนึ่งชิ้น ดังเช่นตัวอย่างงานด้านระเบียบนักเรียนที่ได้กล่าวมาในตอนต้นที่โปรแกรมต้องสามารถประมวลผลข้อมูลนักเรียนเข้ากันหลาย ๆ คนได้ โดยที่ข้อมูลของนักเรียนแต่ละคนแม้จะแตกต่างกัน แต่ก็มีโครงสร้างข้อมูลเหมือนกัน ดังนั้นอาร์เรย์ของโครงสร้างจึงมีความเหมาะสมที่สุดสำหรับการจัดการข้อมูลลักษณะนี้

ตามที่ได้กล่าวมาแล้วว่าโครงสร้างที่ถูกนิยามไว้เรียบร้อยแล้วสามารถถูกนำมาใช้เหมือนกับเป็นชนิดข้อมูลชนิดหนึ่งได้ทันที ดังนั้นการประกาศตัวแปรแบบอาร์เรย์ของโครงสร้างจึงมีรูปแบบเหมือนกับการประกาศอาร์เรย์ทั่ว ๆ ไป เพียงแต่ระบุชนิดข้อมูลให้ตรงกับชื่อโครงสร้างนั้น ๆ เท่านั้น ตัวอย่างเช่นหากเราต้องการประกาศตัวแปรชื่อ *students* เพื่อเป็นอาร์เรย์หนึ่งมิติของโครงสร้าง *StdInfo* ที่นิยามไว้ในตัวอย่างที่ 10.1 เราจะใช้คำสั่งดังนี้

```
StdInfo[] students;
```

อย่าลืมว่าอาร์เรย์จริง ๆ ยังไม่ถูกสร้างจนกว่าจะใช้คำสั่ง **new** ตัวอย่างเช่น

```
students = new StdInfo[30];
```

จะเป็นการสร้างอาร์เรย์ที่บรรจุข้อมูลชนิด *StdInfo* ได้ 30 ช่อง อ้างอิงผ่านตัวแปรชื่อ *students*

การเข้าถึงข้อมูลในอาร์เรย์ของโครงสร้างจะมีรูปแบบเหมือนการเข้าถึง% ข้อมูลในอาร์เรย์ของข้อมูลพื้นฐานทั่ว ๆ ไป โดยระบุตำแหน่งในอาร์เรย์ด้วยดัชนี ต่างกันตรงที่รูปแบบการใช้งานนี้จะไม่เป็นเหมือนตัวแปรที่เก็บค่าใด ๆ อีกต่อไป แต่เป็นเหมือนตัวแปรแบบโครงสร้างนั้น ๆ เช่น

```
students[5]
```

จะทำงานเหมือนตัวแปรแบบโครงสร้างที่เก็บข้อมูลของนักเรียน ณ ดัชนี 5 (นักเรียนคนที่ 6 ในอาร์เรย์) ดังนั้นการเข้าถึงข้อมูลภายในโครงสร้างจึงทำได้โดยการระบุชื่อสมาชิกของโครงสร้างต่อท้ายรูปแบบข้างต้นเท่านั้น เช่นชื่อของนักเรียนคนนี้จะถูกอ้างอิงผ่าน

```
students[5].name
```



ซึ่งหากพิจารณาจากโครงสร้าง *StdInfo* แล้ว *students[5].name* จะเปรียบเสมือนตัวแปรแบบข้อความตัวหนึ่งที่จะถูกนำไปกำหนดค่าให้ก็ได้ หรือนำไปใช้ในรูปนิพจน์ก็ได้ เช่น

```
students[5].name = "Arthur";  
Console.WriteLine("The student's name is {0}", students[5].name);
```

คณะผู้พัฒนาเอกสาร การโปรแกรมภาษาซีชาร์ป

คณะผู้พัฒนาเอกสารฝึกอบรมครู

1. ดร.ชัยพร ใจแก้ว	มหาวิทยาลัยเกษตรศาสตร์	กรุงเทพฯ
2. ดร.จิตติวรณ ศรีนาค	มหาวิทยาลัยเกษตรศาสตร์	กรุงเทพฯ
3. รศ.ธีรวัฒน์ ประกอบผล	สถาบันเทคโนโลยีพระจอมเกล้าฯ ลาดกระบัง	กรุงเทพฯ
4. อาจารย์สายชล ใจเย็น	สถาบันเทคโนโลยีพระจอมเกล้าฯ ลาดกระบัง	กรุงเทพฯ
5. นายชยการ คีรีรัตน์	สาธิตจุฬาลงกรณ์มหาวิทยาลัย ฝ่ายมัธยม	กรุงเทพฯ
6. นายนิพนธ์ สุขศรี	สสวท.	กรุงเทพฯ
7. นางสาวจินดาพร หมวกหมื่นไวย	สสวท.	กรุงเทพฯ
8. นางสาวทัศนีย์ กรองทอง	สสวท.	กรุงเทพฯ
9. นางสาวพรพิมล ตั้งชัยสิน	สสวท.	กรุงเทพฯ
10. นางสาวจิระพร สังขเวทย์	สสวท.	กรุงเทพฯ
11. นายนิรมิช เพียรประเสริฐ	สสวท.	กรุงเทพฯ
12. นายพนมยงค์ แก้วประชุม	สสวท.	กรุงเทพฯ

