4.2 การวนซ้ำและตัวแปรแถวลำดับ

การเขียนโปรแกรมแบบวนซ้ำ (Loop) เป็นการเขียนโปรแกรมที่มีลักษณะการทำงานซ้ำๆ เมื่อเขียนโปรแกรมแบบวนซ้ำจะช่วยให้การเขียนโปรแกรมสั้นลง ซึ่งการเขียนโปรแกรมแบบวนซ้ำมีรูปแบบการเขียนหลายแบบ ขึ้นอยู่กับความเหมาะสมและสถานการณ์ที่แตกต่างกัน ในการโปรแกรมแบบวนซ้ำนี้มักจะใช้ตัวแปรแบบแถวลำดับ (Array) มาช่วยในการเก็บข้อมูล เพราะตัวแปรแบบแถวลำดับจะมีการเก็บข้อมูลเป็นชุดและมีชนิดข้อมูลเดียวกันทั้งชุด ในหนึ่งชุดก็สามารถเก็บข้อมูลได้มากกว่าหนึ่งตัว เช่น ชุดข้อมูลวันที่ 31 วัน ตัวเลขจำนวน 50 ตัว เป็นต้น การเรียกใช้ตัวแปรแถวลำดับจำเป็นจะต้องอาศัยการเขียนโปรแกรมแบบวนซ้ำมาช่วยในการเข้าถึงชุดข้อมูลนั้นๆ

4.2.1 คำสั่ง for

เป็นคำสั่งวนซ้ำ ซึ่งจะใช้ในกรณีที่รู้จำนวนรอบการวนซ้ำที่แน่นอน

<u>รูปแบบการใช้คำสั่ง for</u>

```
                1                    2                    4
for ( ตัวแปรที่ใช้เป็นดัชนี = ค่าเริ่มต้น ; เงื่อนไขที่ทำให้โปรแกรมหยุดวนซ้ำ ; เปลี่ยนแปลงค่าดัชนี )
{
        คำสั่ง ;        3
}
```

จากรูปแบบการใช้คำสั่ง for มีขั้นตอนการทำงาน ดังนี้

หมายเลข 1 กำหนดค่าเริ่มต้นให้ตัวแปรที่ใช้เป็นดัชนี

หมายเลข 2 ตรวจสอบเงื่อนไขให้โปรแกรมหยุดวนซ้ำ

หมายเลข 3 ถ้าเงื่อนไขเป็นจริงทำงานคำสั่งที่อยู่ภายในคำสั่ง For

หมายเลข 4 เปลี่ยนแปลงค่าดัชนี จากนั้นจะวนซ้ำกลับไปที่ หมายเลข 2

ตัวอย่างการใช้คำสั่ง for

```
int i;
 for (i = 1; i <= 5; i++)
 {
     textBox1.Text += "LOVE YOU \r\n";
 }
```

อธิบาย

จากตัวอย่างการใช้คำสั่ง For มีขั้นตอนการทำงาน ดังนี้

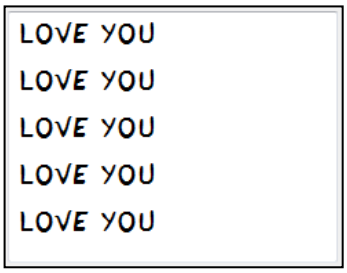
ลำดับที่ 1 กำหนดค่าเริ่มต้นให้ตัวแปร i มีค่าเท่ากับ 1

ลำดับที่ 2 ตรวจสอบเงื่อนไข i ต้องมีค่าน้อยกว่าหรือเท่ากับ 5

ลำดับที่ 3 ถ้าเงื่อนไขเป็นจริง แสดงคำว่า LOVE YOU แล้วขึ้นบรรทัดใหม่

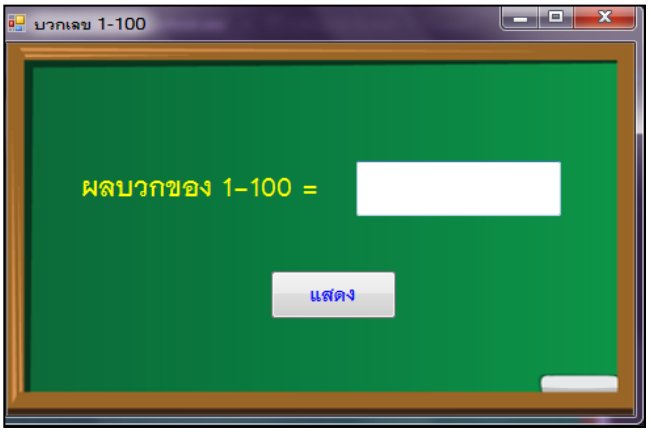
ลำดับที่ 4 เพิ่มค่าให้ i ทีละ 1 จากนั้นจะวนซ้ำกลับไปที่ ลำดับที่ 2

ลำดับที่ 5 เมื่อวนทำงานครบ 5 รอบ ก็จะหยุดการทำงาน และแสดงผลลัพธ์ ดังนี้



ตัวอย่างโปรแกรมที่ใช้คำสั่ง for

ตัวอย่างการใช้คำสั่ง for กับโปรแกรมบวกเลข 1-100 โดยผู้ใช้คลิกปุ่มแสดง โปรแกรมจะแสดงผลบวกของ 1-100 ในคอนโทรล textbox ซึ่งโปรแกรมมีการออกแบบหน้าจอ ดังนี้

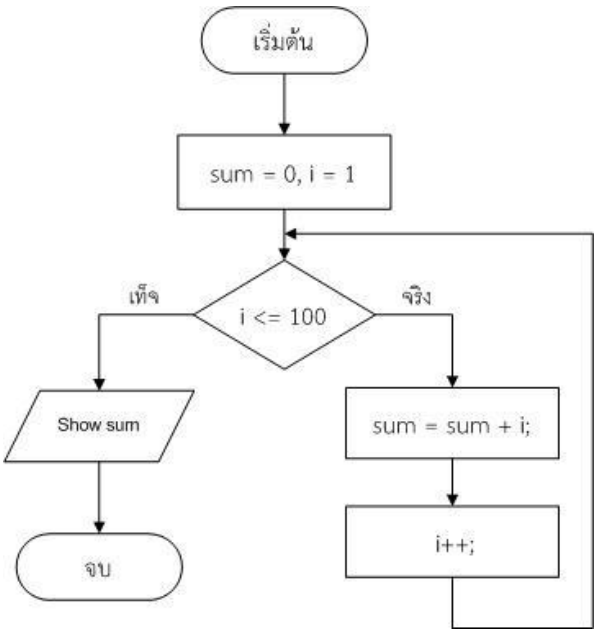


ดับเบิ้ลคลิกที่ปุ่มแสดง แล้วเขียนคำสั่งต่อไปนี้

```
int i, sum = 0;
 for (i = 1; i <= 100; i++)
 {
     sum = sum + i;
 }
          textBox1.Text = sum.ToString();
```

บันทึกงานแล้วกดรัน และคลิกที่ปุ่มแสดง จะได้ผลลัพธ์ดังหน้าจอ

แสดงแผนผังลำดับงานของโปรแกรมได้ดังนี้



4.2.2 คำสั่ง do...while

เป็นคำสั่งวนซ้ำ ซึ่งจะใช้ในกรณีที่ไม่รู้จำนวนรอบการวนซ้ำที่แน่นอน คำสั่งนี้จะทำงานก่อน 1 รอบ แล้วตรวจสอบเงื่อนไขทีหลัง

<u>รูปแบบการใช้คำสั่ง do...while</u>

```
do
{
        คำสั่ง ;

} while(เงื่อนไข);
```

จากรูปแบบการใช้คำสั่ง do...while มีขั้นตอนการทำงาน ดังนี้

หมายเลข 1 ทำงานคำสั่งที่อยู่ภายในคำสั่ง do

หมายเลข 2 ตรวจสอบเงื่อนไขในคำสั่ง while

- หากตรวจสอบเงื่อนไขแล้วเป็นเท็จโปรแกรมจะหยุดวนซ้ำ
- หากตรวจสอบเงื่อนไขแล้วเป็นจริงจะวนกลับไปทำงานที่หมายเลข 1 อีกครั้ง

<u>ตัวอย่างการใช้คำสั่ง do...while</u>

```
int i=1;
 do
 {
     textBox1.Text += "LOVE YOU \r\n";
     i++;
 } while ( i <= 5 ) ;
```

อธิบาย

จากตัวอย่างการใช้คำสั่ง do...while มีขั้นตอนการทำงาน ดังนี้

ลำดับที่ 1 ประกาศตัวแปร i ให้มีค่าเริ่มต้น เท่ากับ 1

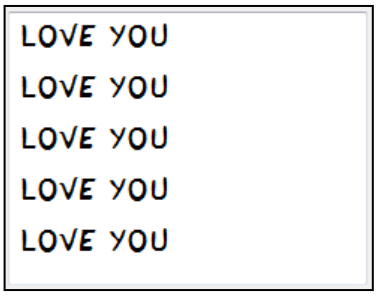
ลำดับที่ 2 แสดงคำว่า LOVE YOU ใน textbox1 แล้วขึ้นบรรทัดใหม่

ลำดับที่ 3 เพิ่มค่าให้ i ทีละ 1

ลำดับที่ 4 ตรวจสอบเงื่อนไขว่าตัวแปร i ต้องมีค่าน้อยกว่าหรือเท่ากับ 5 หรือไม่

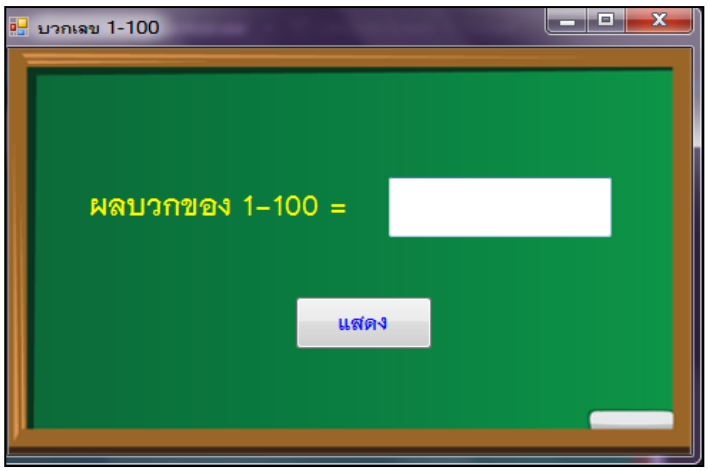
- หากตรวจสอบเงื่อนไขแล้วเป็นเท็จโปรแกรมจะหยุดวนซ้ำ
- หากตรวจสอบเงื่อนไขแล้วเป็นจริงจะวนกลับไปทำงานลำดับที่ 2 อีกครั้ง

ลำดับที่ 5 เมื่อวนทำงานครบ 5 รอบ ก็จะหยุดการทำงาน และแสดงผลลัพธ์ ดังนี้



ตัวอย่างโปรแกรมที่ใช้คำสั่ง do...while

ตัวอย่างการใช้คำสั่ง do...while กับโปรแกรมบวกเลข 1-100 โดยผู้ใช้คลิกปุ่มแสดง โปรแกรมจะแสดงผลบวกของ 1-100 ในคอนโทรล textbox ซึ่งโปรแกรมมีการออกแบบหน้าจอ ดังนี้



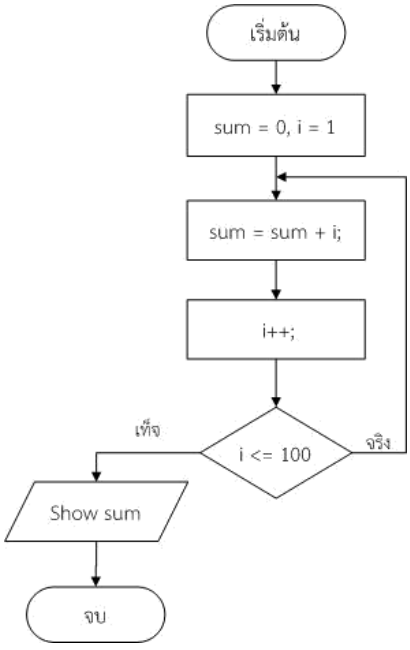
ดับเบิ้ลคลิกที่ปุ่มแสดง แล้วเขียนคำสั่งต่อไปนี้

```
int i=1, sum = 0;
do
{
    sum = sum + i;
    i++;
} while(i <= 100);

  textBox1.Text = sum.ToString();
```

บันทึกงานแล้วกดรัน และคลิกที่ปุ่มแสดง จะได้ผลลัพธ์ดังหน้าจอ

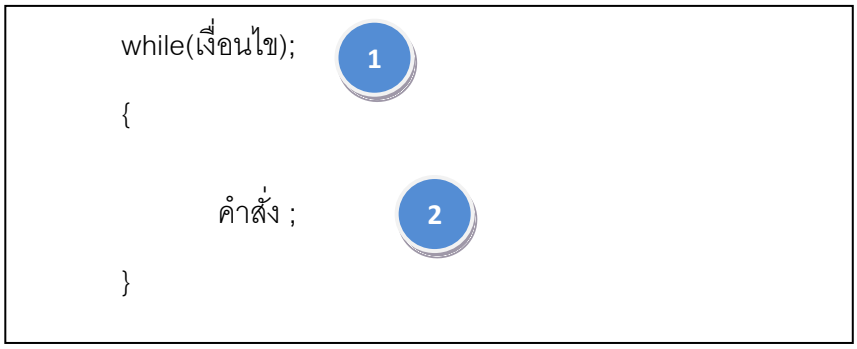
แสดงแผนผังลำดับงานของโปรแกรมได้ดังนี้



4.2.3 คำสั่ง while

เป็นคำสั่งวนซ้ำ ซึ่งจะใช้ในกรณีที่ไม่รู้จำนวนรอบการวนซ้ำที่แน่นอน คำสั่งนี้จะทำงานก็ต่อเมื่อตรวจสอบเงื่อนไขแล้วเป็นจริง

<u>รูปแบบการใช้คำสั่ง while</u>



จากรูปแบบการใช้คำสั่ง while มีขั้นตอนการทำงาน ดังนี้

หมายเลข 1 ตรวจสอบเงื่อนไขในคำสั่ง while

- หากตรวจสอบเงื่อนไขแล้วเป็นเท็จโปรแกรมไม่เข้าไปทำงานภายใต้คำสั่ง while
- หากตรวจสอบเงื่อนไขแล้วเป็นจริงจะทำงานคำสั่งตาม หมายเลข 2

<u>ตัวอย่างการใช้คำสั่ง do...while</u>

```
int i=1;
 while ( i <= 5 )
 {
     textBox1.Text += "LOVE YOU \r\n";
     i++;
 }
```

อธิบาย

จากตัวอย่างการใช้คำสั่ง while มีขั้นตอนการทำงาน ดังนี้

ลำดับที่ 1 ประกาศตัวแปร i ให้มีค่าเริ่มต้น เท่ากับ 1

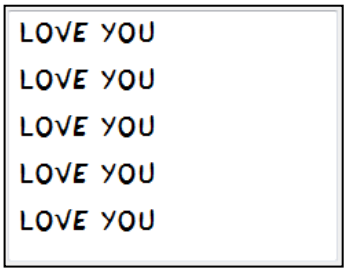
ลำดับที่ 2 ตรวจสอบเงื่อนไขว่าตัวแปร i ต้องมีค่าน้อยกว่าหรือเท่ากับ 5 หรือไม่

- หากตรวจสอบเงื่อนไขแล้วเป็นเท็จไม่เข้าไปทำงานภายใต้คำสั่ง while
- หากตรวจสอบเงื่อนไขแล้วเป็นจริงจะแสดงคำว่า LOVE YOU แล้วขึ้นบรรทัดใหม่

ลำดับที่ 3 เพิ่มค่าให้ i ทีละ 1

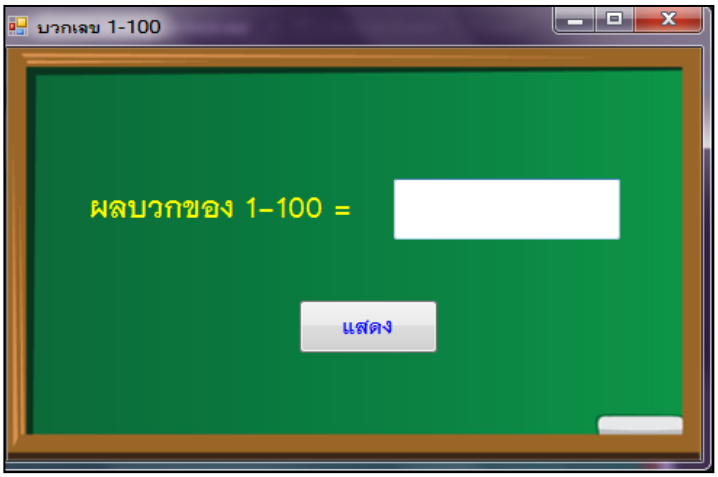
ลำดับที่ 4 วนซ้ำกลับไปแสดง LOVE YOU แล้วขึ้นบรรทัดใหม่ ในลำดับที่ 2 และ 3

ลำดับที่ 5 เมื่อวนทำงานครบ 5 รอบ ก็จะหยุดการทำงาน และแสดงผลลัพธ์ ดังนี้



ตัวอย่างโปรแกรมที่ใช้คำสั่ง while

ตัวอย่างการใช้ while กับโปรแกรมบวกเลข 1-100 โดยผู้ใช้คลิกปุ่มแสดง โปรแกรมจะแสดงผลบวกของ 1-100 ในคอนโทรล textbox ซึ่งโปรแกรมมีการออกแบบหน้าจอ ดังนี้



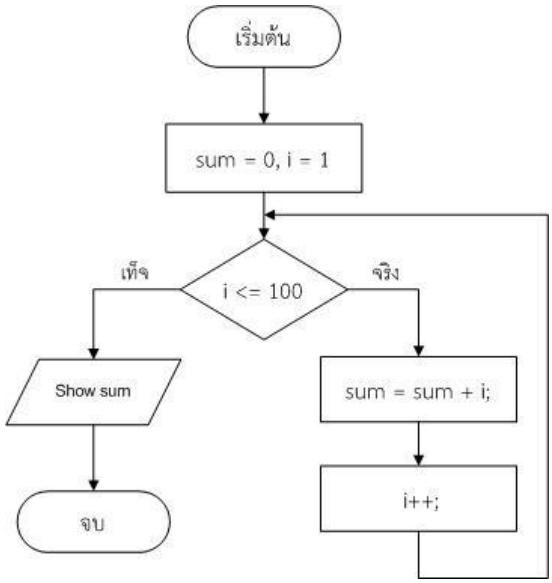
ดับเบิ้ลคลิกที่ปุ่มแสดง แล้วเขียนคำสั่งต่อไปนี้

```
int i=1, sum = 0;
while(i <= 100)
{
    sum = sum + i;
    i++;
}

  textBox1.Text = sum.ToString();
```

บันทึกงานแล้วกดรัน และคลิกที่ปุ่มแสดง จะได้ผลลัพธ์ดังหน้าจอ

แสดงแผนผังลำดับงานของโปรแกรมได้ดังนี้



4.2.4 ตัวแปรแถวลำดับ (Array)

แถวลำดับ(array) เป็นโครงสร้างข้อมูลที่ประกอบไปด้วยสมาชิกหรือตัวแปรที่มีชนิดเดียวกันเรียงต่อกันเป็นชุด โดยสมาชิกแต่ละตัวจะมีหมายเลขอ้างอิงบอกลำดับของสมาชิก ที่เรียกว่า ดัชนี (Index)

การเข้าถึงสมาชิกของแถวลำดับทำได้โดยการใช้ชื่อของแถวลำดับตามด้วยดัชนีเพื่อระบุสมาชิกที่ต้องการ เช่น x[1] ขอบเขตของดัชนีในภาษา C# สามารถกำหนดให้มีค่าได้เป็นช่วงตามความต้องการในรูปแบบ m...n เมื่อ m เป็นค่าลำดับแรกของสมาชิก และ n เป็นค่าสุดท้ายของสมาชิก เช่น ต้องการให้ x เป็นแถวลำดับชนิดจำนวนเต็ม ที่มีสมาชิก 10 ตัว จะประกาศได้ดังนี้

```
int [ ] x = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

แผนภาพแสดงโครงสร้างของแถวลำดับ

เมื่อประกาศตัวแปร x เป็นแถวลำดับชนิดจำนวนเต็ม ที่มีสมาชิก 10

```
int [ ] x = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

จะได้โครงสร้างการเก็บข้อมูลแถวลำดับ ดังนี้

แถวลำดับ x	1	2	3	4	5	6	7	8	9	10
	X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]	X[9]

1. แถวลำดับ 1 มิติ

แถวลำดับ 1 มิติ เป็นชุดของตัวแปรที่เรียงต่อกันเป็นแถวโดยที่สมาชิกแต่ละตัวของแถวลำดับจะมีดัชนีเพียง 1 ตัว เช่น

```
x[6] , num[0]
```

นั่นคือ - x และ num เป็นชื่อของตัวแปร

- เลขใน [ ] เป็นดัชนีบอกลำดับของสมาชิก

รูปแบบการประกาศตัวแปรแถวลำดับ 1 มิติ

```
ชนิดตัวแปร [ ] ชื่อตัวแปร = new ชนิดตัวแปร [จำนวนสมาชิก] { สมาชิก1, สมาชิก2,...สมาชิกตัวสุดท้าย}
```

ตัวอย่างการเขียนตัวแปรแถวลำดับ 1 มิติ ชนิดตัวเลขจำนวนเต็ม

```
int [ ] num1 = new int [4] { 2, 4, 6, 8}   //จำกัดสมาชิก 4 ตัว
int [ ] num2 = new int [ ] { 2, 4, 6, 8}   //ไม่จำกัดสมาชิก
int [ ] num3 = { 2, 4, 6, 8}   //ไม่จำกัดสมาชิก
```

อธิบาย

- int [ ] num1 = new int [4] { 2, 4, 6, 8} คือ การประกาศตัวแปร num1 เป็นตัวแปรแถวลำดับที่มีชนิดข้อมูลเป็นจำนวนเต็ม มีสมาชิก 4 ตัวเท่านั้น คือ num1 [0] = 2 , num1 [1] = 4 , num1 [2] = 6 , num1 [3] = 8

- int [ ] num2 = new int [ ] { 2, 4, 6, 8} คือ การประกาศตัวแปร num2 เป็นตัวแปรแถวลำดับที่มีชนิดข้อมูลเป็นจำนวนเต็ม มีสมาชิก 4 ตัวแบบไม่จำกัดจำนวนสมาชิก คือ num2 [0] = 2 , num2 [1] = 4 , num2 [2] = 6 , num2 [3] = 8

- int [ ] num3 = { 2, 4, 6, 8} คือ การประกาศตัวแปร num3 เป็นตัวแปรแถวลำดับที่มีชนิดข้อมูลเป็นจำนวนเต็ม มีสมาชิก 4 ตัวไม่จำกัดจำนวนสมาชิก คือ num3 [0] =2 , num3 [1] =4 , num3 [2] = 6 , num3 [3] =8

ตัวอย่างการเขียนตัวแปรแถวลำดับ 1 มิติ ชนิดข้อความ

```
string [ ] str1 = new string [3] { “John”, “Paul”, “Mary”}   //จำกัดสมาชิก 3 ตัว
string [ ] str2 = new string [ ] { “John”, “Paul”, “Mary”}  //ไม่จำกัดสมาชิก
string [ ] str3 = { “John”, “Paul”, “Mary”}   //ไม่จำกัดสมาชิก
```

อธิบาย

- string [ ] str1 = new string [3] { “John”, “Paul”, “Mary”} คือ การประกาศตัวแปร str1 เป็นตัวแปรแถวลำดับที่มีชนิดข้อมูลเป็นข้อความ มีสมาชิก 3 ตัวเท่านั้น คือ str1 [0] = John , str1[1] = Paul , str1 [2] = Mary

- string [ ] str2 = new string [ ] { “John”, “Paul”, “Mary”} คือ การประกาศตัวแปร str2 เป็นตัวแปรแถวลำดับที่มีชนิดข้อมูลเป็นข้อความ มีสมาชิก 3 ตัวแบบไม่จำกัดจำนวนสมาชิก คือ str2 [0]= John , str2 [1] = Paul , str2 [2] = Mary

- string [ ] str3 = { “John”, “Paul”, “Mary”} คือ การประกาศตัวแปร str3 เป็นตัวแปรแถวลำดับที่มีชนิดข้อมูลเป็นข้อความ มีสมาชิก 3 ตัวแบบไม่จำกัดจำนวนสมาชิก คือ str3 [0] = John , str3 [1] = Paul , str3 [2] = Mary

<u>ตัวอย่างโปรแกรมที่ใช้แถวลำดับ 1 มิติ</u>

ตัวอย่างการใช้ตัวแปรแถวลำดับกับโปรแกรมสุ่มตัวเลข โดยผู้ใช้ป้อนจำนวนชุดของตัวเลข และคลิกปุ่มสุ่ม โปรแกรมจะแสดงตัวเลขที่สุ่มไปเก็บไว้ในตัวแปรแถวลำดับออกมามาตามจำนวนชุดตัวเลขที่ผู้ใช้งานต้องการในคอนโทรล Richtextbox1 ซึ่งโปรแกรมมีการออกแบบหน้าจอ ดังนี้

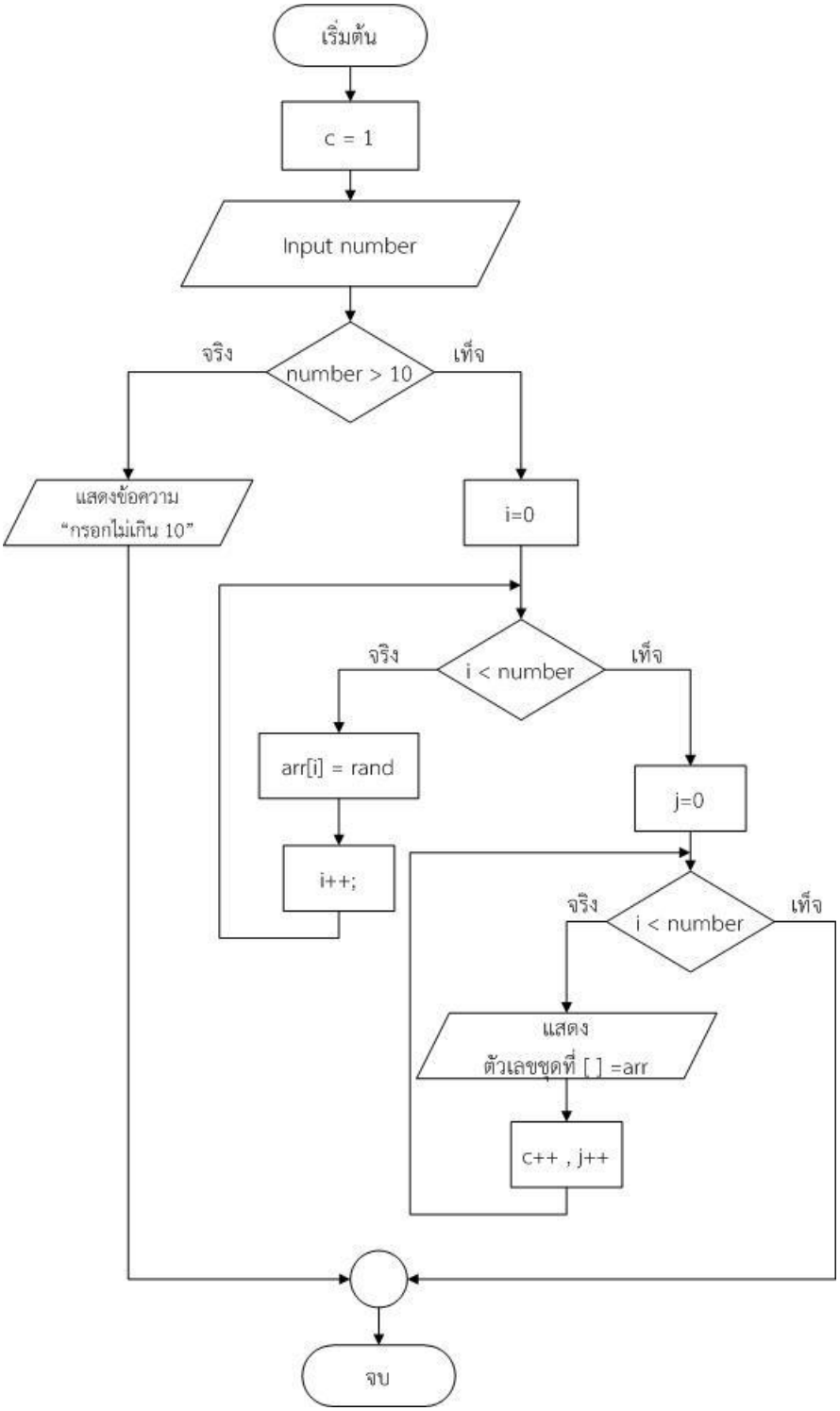
ดับเบิ้ลคลิกที่ปุ่ม**คลิกสุ่ม** แล้วเขียนคำสั่งต่อไปนี้

```
int c = 1;
int[ ] arr = new int[10];
richTextBox1.Clear();
Random rand = new Random();
int number = int.Parse(textBox1.Text);
if (number > 10)
{
        MessageBox.Show("กรอกตัวเลขไม่เกิน 10");
 }
else
{
        for (int i = 0; i < number; i++)
         {
                arr[i] = rand.Next(900)+100 ;
         }

         for(int j = 0; j < number; j++)
         {
                 richTextBox1.Text += "ชุดที่ ["+c.ToString()+"]"+"="+arr[j].ToString()+"\n" ;
                c++;
        }
}
```

บันทึกงานแล้วกดรัน และคลิกที่ปุ่ม **คลิกสุ่ม** จะได้ผลลัพธ์ดังหน้าจอ

แสดงแผนผังลำดับงานของโปรแกรมได้ดังนี้



2. แถวลำดับ 2 มิติ

แถวลำดับ 2 มิติ เป็นชุดของตัวแปรเหมือนการเก็บค่าในตารางที่มีแถวและคอลัมน์ ในการประกาศอาเรย์ 2 มิตินั้น จะคล้ายกับอาเรย์ 1 มิติ แต่มันจะใช้เครื่องหมาย [,] โดยแต่ละคู่นั้นแสดงแถวและคอลัมน์โดย ROW คือจำนวนของแถวของอาเรย์ และ COLUMN หรือขนาดคอลัมน์ของอาเรย์ ดังนั้นจำนวนสมาชิกของมันจึงเป็น ROW * COLUMN เช่น

```
x[ 3 , 2 ] ,  num[ 4 , 4 ]
```

นั่นคือ - x และ num เป็นชื่อของตัวแปร

- เลขใน [ ] บอกตำแหน่งของแถวตมด้วยตำแหน่งของคอลัมน์

<u>รูปแบบการประกาศตัวแปรแถวลำดับ 2 มิติ</u>

```
ชนิดตัวแปร [ , ] ชื่อตัวแปร = new ชนิดตัวแปร [จำนวนแถว , จำนวนคอลัมน์] ;
ชนิดตัวแปร [ , ] ชื่อตัวแปร = new ชนิดตัวแปร [ , ] { {‘สมาชิก’ , ‘สมาชิก’ } , {‘สมาชิก’ , ‘สมาชิก’ } } ;
```

<u>ตัวอย่างการเขียนตัวแปรแถวลำดับ 2 มิติ ชนิดตัวเลขจำนวนเต็ม</u>

```
int [ , ] number= new int [4,4] ;
int [ , ] arr = new int [ , ] { {‘2’ , ‘4’} , { ‘6’ , ‘8’ } } ;
```

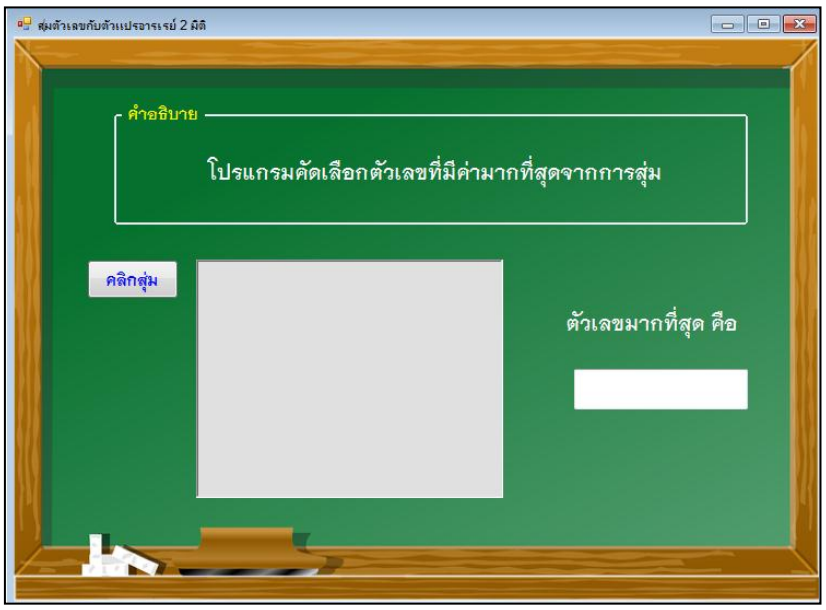
<u>อธิบาย</u>

- int [,] number= new int [4,4] คือ การประกาศตัวแปร number เป็นตัวแปรแถวลำดับ 2 มิติ ที่มีชนิดข้อมูลเป็นจำนวนเต็ม มีจำนวน 4 แถว และ 4 คอลัมน์ ไม่ได้กำหนดค่าเริ่มต้น

- int [,] arr = new int [,] { {‘2’ , ‘4’} , { ‘6’ , ‘8’ } } ; คือ การประกาศตัวแปร arr เป็นตัวแปรแถวลำดับที่มีชนิดข้อมูลเป็นจำนวนเต็ม ซึ่งมีการกำหนดค่าเริ่มต้น ดังนี้ arr[0][0] = 2 , arr[0][1] = 4 , arr[1][0] = 6 , arr[1][1] = 8

ตัวอย่างโปรแกรมที่ใช้แถวลำดับ 2 มิติ

ตัวอย่างการใช้ตัวแปรแถวลำดับ 2 มิติ กับโปรแกรมคัดเลือกตัวเลขที่มากที่สุดจากการโปรแกรม โดยผู้ใช้ป้อนจำนวนนักเรียน ป้อนคะแนน และคลิกปุ่มแสดง โปรแกรมจะแสดงคะแนนของนักเรียนแต่ละคนออกมา ในคอนโทรล Richtextbox1 ซึ่งโปรแกรมมีการออกแบบหน้าจอ ดังนี้

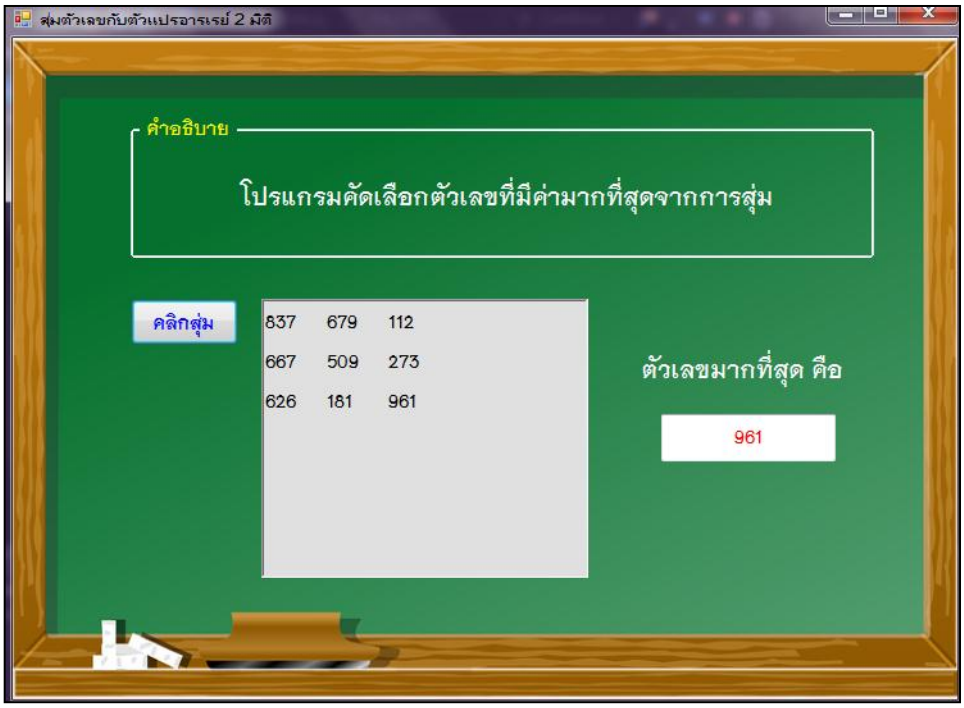


ดับเบิ้ลคลิกที่ปุ่ม**คลิกสุ่ม** แล้วเขียนคำสั่งต่อไปนี้

```
int tmp = 0;
int[,] arr = new int[3,3];
richTextBox1.Clear();
textBox1.Clear();
Random rand = new Random();
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        arr[i,j] = rand.Next(900)+100;
    }
}
```

```
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        richTextBox1.Text +=    arr[i,j].ToString() + "\t";
        if(arr[i ,j] > tmp)
        {
            tmp = arr[i, j];
        }
    }
    richTextBox1.Text += "\n";
}
textBox1.Text = tmp.ToString();
```

บันทึกงานแล้วกดรัน และคลิกที่ปุ่ม **คลิกสุ่ม** จะได้ผลลัพธ์ดังหน้าจอ



แสดงแผนผังลำดับงานของโปรแกรมได้ดังนี้